Detecting Weakly Simple Polygons[*]

Hsien-Chih Chang Jeff Erickson Chao Xu

Department of Computer Science
University of Illinois, Urbana-Champaign

Abstract

A closed curve in the plane is weakly simple if it is the limit (in the Fréchet metric) of a sequence of simple closed curves. We describe an algorithm to determine whether a closed walk of length n in a simple plane graph is weakly simple in $O(n \log n)$ time, improving an earlier $O(n^3)$-time algorithm of Cortese *et al.* [*Discrete Math.* 2009]. As an immediate corollary, we obtain the first efficient algorithm to determine whether an arbitrary n-vertex polygon is weakly simple; our algorithm runs in $O(n^2 \log n)$ time. We also describe algorithms that detect weak simplicity in $O(n \log n)$ time for two interesting classes of polygons. Finally, we discuss subtle errors in several previously published definitions of weak simplicity.

Dedicated with thanks to our colleague
Ferran Hurtado (1951–2014)

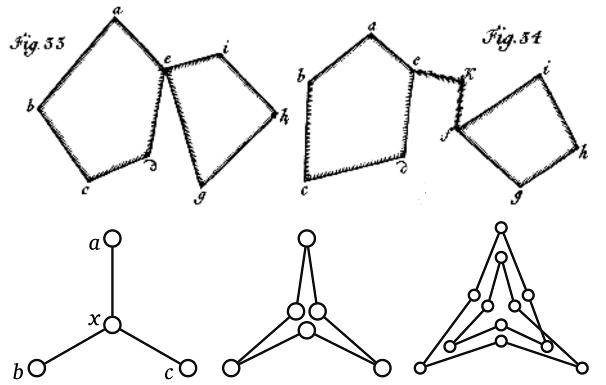

Figure 1.1. Top: Two weakly simple polygons from Meister's seminal 1770 treatise on polygons [37]. Bottom: The polygon (a, x, b, x, c, x) is weakly simple; the polygon $(a, x, b, x, c, x, a, x, b, x, c, x)$ is not.

1 Introduction

Simple polygons in the plane have been standard objects of study in computational geometry for decades, and in the broader mathematical community for centuries. Many algorithms designed for simple polygons continue to work with little or no modification in degenerate cases, where intuitively the polygon overlaps itself but does not cross itself. We offer the first complete and efficient algorithm to detect such degenerate polygons.

Formally, a *closed curve* in the plane is a continuous function $P\colon S^1 \to \mathbb{R}^2$. A closed curve is *simple* if it is injective, and *weakly simple* if for any $\varepsilon > 0$, there is a simple closed curve whose Fréchet distance from P is at most ε. A recent nontrivial result of Ribó Mor [40] implies that a *polygon* P with at least three vertices is weakly simple if and only if, for any $\varepsilon > 0$, we can obtain a simple polygon by perturbing each vertex of P within a ball of radius ε. See Figure 1.1 for some small examples.

Unfortunately, neither of these definitions imply efficient algorithms to determine whether a given polygon is weakly simple. Several authors have offered alternative characterizations of weakly simple polygons, all building on the intuition that a weakly simple polygon does not "cross itself". Unfortunately, *none* of these characterizations is entirely consistent with the formal definition, especially when the polygon contains *spurs*—vertices whose two incident edges overlap. We discuss these earlier definitions in detail in Section 3.

We describe an algorithm to determine whether a closed walk of length n in a planar straight-line graph is weakly simple in $O(n \log n)$ time. Our algorithm is essentially a more efficient implementation of an earlier $O(n^3)$-time algorithm of Cortese *et al.* [17] for the same problem. Since any n-vertex polygon can be subdivided into a walk of length $O(n^2)$ in union of the polygon's vertices and edges, we immediately obtain an algorithm to decide whether a polygon is weakly simple in $O(n^2 \log n)$ time. The quadratic blowup is caused by *forks* in the polygon: vertices that lie in the interior of many overlapping collinear edges. For polygons without forks, the running time is simply $O(n \log n)$. We also describe a simpler $O(n \log n)$-time algorithm for polygons without spurs (but possibly with forks).

Our paper is organized as follows. We review standard terminology and formally define weak simplicity in Section 2, and discuss problems with previously published definitions in Section 3. In Section 4, as a warm-up to our later results, we describe a simple algorithm to determine whether a polygon without spurs or forks is

[*]Work on this paper was partially supported by NSF grant CCF-0915519. See http://www.cs.illinois.edu/~jeffe/pubs/weak.html for the most recent version of this paper.

weakly simple in $O(n \log n)$ time. Section 5 describes our $O(n \log n)$-time algorithm for polygons without spurs, but possibly with forks. We give a complete description of the algorithm of Cortese *et al.* for closed walks in planar graphs, reformulated using our terminology, in Section 6 and then describe and analyze our faster implementation in Section 7. We conclude in Section 8 by describing several extensions of our results and some interesting open problems.

2 Background

2.1 Curves and Polygons

A ***path*** in the plane is formally defined as a continuous function $P: [0,1] \to \mathbb{R}^2$. A ***closed curve*** in the plane is any continuous function $P: S^1 \to \mathbb{R}^2$. A path or closed curve is ***simple*** if it is injective.

A ***polygon*** is a piecewise-linear closed curve in the plane. Every polygon is defined by a cyclic sequence of points, called ***vertices***, connected by line segments, called ***edges***. We often describe polygons by listing their vertices in parentheses; for example, $\boldsymbol{(p_0, p_1, \ldots, p_{n-1})}$ denotes the polygon with vertices p_i and edges $p_i p_{i+1 \bmod n}$ for every index i. A polygon is simple if and only if its vertices are distinct and its edges intersect only at common endpoints. We emphasize that the vertices of a non-simple polygon need not be distinct, and two edges of a non-simple polygon may overlap or even coincide [26, 37]. However, we do assume without (significant) loss of generality that every edge of a polygon has positive length.

Similarly, a ***polygonal chain*** is a piecewise-linear path, which consists of a linear sequence of points (vertices) connected by line segments (edges). We often describe polygonal chains by listing their vertices in square brackets, like $\boldsymbol{[p_0, p_1, \ldots, p_{n-1}]}$, to distinguish them from polygons. A ***corner*** of a polygon or polygonal chain P is a subpath $[p_{i-1}, p_i, p_{i+1}]$ consisting of two consecutive edges of P.

Three local features of polygons play important roles in our results. A ***spur*** of a polygon P is a vertex of P whose two incident edges overlap. A ***fork*** of a polygon P is a vertex of P that lies in the interior of an edge of P. Finally, a ***simple crossing*** between two paths is a point of *transverse intersection*: within any sufficiently small neighborhood of the intersection point, the paths are homeomorphic to two intersecting straight lines. If the two paths are polygonal chains, this intersection point could be a vertex of one or both paths.

2.2 Distances

Let $\boldsymbol{d(p,q)}$ denote the Euclidean distance between two points p and q in the plane. The ***Fréchet distance*** $\boldsymbol{d_{\mathcal{F}}(P,Q)}$ between two closed curves P and Q is defined as

$$d_{\mathcal{F}}(P,Q) := \inf_{\rho : S^1 \to S^1} \max_{t \in S^1} d(P(\rho(t)), Q(t)),$$

where the infimum is taken over all orientation-preserving homeomorphisms of S^1. The function ρ is often called a ***reparametrization*** of S^1. Fréchet distance is a complete metric over the space of all (unparametrized) closed curves in the plane. The Fréchet distance between paths is defined similarly.

The ***vertex distance*** $\boldsymbol{d_V(P,Q)}$ between two polygons $P = (p_0, p_1, \ldots, p_{n-1})$ and $Q = (q_0, q_1, \ldots, q_{n-1})$ with the same number of vertices is defined as

$$d_V(P,Q) := \min_s \max_i d(p_i, q_{i+s \bmod n}).$$

For each integer n, vertex distance is a complete metric over the space of all n-vertex polygons (where two polygons that differ only by a cyclic shift of indices are identified).

2.3 Planar Graphs

A ***planar embedding*** of a graph represents the vertices by distinct points in the plane, and the edges by interior-disjoint simple paths between their endpoints. Any planar graph embedding partitions the plane into several regions, called the ***faces*** of the embedding. Euler's formula states that any planar embedding of a connected graph with V vertices and E edges has exactly $2 - V + E$ faces. Any planar graph embedding can be represented abstractly by a ***rotation system***, which records for each vertex the cyclic sequence of incident edges [38].

A ***planar straight-line graph*** is a planar graph embedding in which every edge is represented by a single line segment. Planar straight-line graphs can be represented by several data structures, each of which allows fast access to the abstract graph, the coordinates of its vertices, and the rotation system of its embedding. Two popular examples are the quad-edge data structure [27] and the doubly-connected edge list [5].

2.4 Weak Simplicity

Intuitively, a closed curve or polygon is *weakly simple* if it can be made simple by an arbitrarily small perturbation, or equivalently, if it is the limit of a sequence of simple closed curves or polygons. Our two different metrics for curve similarity give us two different formal definitions, one for arbitrary closed curves and the other only for polygons:

- A closed curve P is ***weakly simple*** if, for any $\varepsilon > 0$, there is a simple closed curve Q such that $d_{\mathcal{F}}(P,Q) < \varepsilon$. In other words, a closed curve is weakly simple if it can be made simple by an arbitrarily small perturbation of the entire curve.

- A polygon P is ***rigidly weakly simple*** if, for any $\varepsilon > 0$, there is a simple polygon Q with the same number of vertices such that $d_V(P,Q) < \varepsilon$. In other words, a polygon is rigidly weakly simple if it can be made simple by an arbitrarily small perturbation of its vertices.

For any two polygons P and Q with the same number of vertices, we have $d_{\mathcal{F}}(P,Q) \leq d_V(P,Q)$; thus, every rigidly weakly simple polygon is also weakly simple. The following theorem, whose proof we defer to the full version of our paper, implies that the two definitions are *almost* equivalent for polygons.

Theorem 2.1. *Every weakly simple polygon with more than two vertices is rigidly weakly simple.*

Our proof relies on a nontrivial result of Ribó Mor [40, Theorem 3.1] (previously conjectured by Connelly *et al.* [16, Conjecture 4.1]) about "self-touching" linkage configurations. The restriction to polygons with more than two vertices is necessary; every polygon with at most two vertices is weakly simple, because it is a degenerate ellipse, but not rigidly weakly simple, because every simple polygon has at least three vertices.

3 Problems with Earlier Definitions

Many authors have offered the intuition that a polygon is weakly simple if and only if it "does not cross itself"; indeed, this is the complete definition offered in a few papers [1, 31]. However, correctly formalizing this intuition involves several subtleties that do not appear to have been previously recognized in the literature. Perhaps because of these subtleties, we are unaware of any previous algorithm to determine whether an arbitrary given polygon is weakly simple.

3.1 Crossing

We believe the following topological definitions of "crossing" and "self-crossing" are the most natural and general. Two paths P and Q are ***weakly disjoint*** if, for all sufficiently small $\varepsilon > 0$, there are disjoint paths $\tilde{P}$ and $\tilde{Q}$ such that $d_{\mathcal{F}}(P,\tilde{P}) < \varepsilon$ and $d_{\mathcal{F}}(Q,\tilde{Q}) < \varepsilon$; in other words, two paths are weakly disjoint if they can be made disjoint by arbitrarily small perturbations. Two paths ***cross*** if they are not weakly disjoint. Finally, a closed curve is ***self-crossing*** if it contains two crossing subpaths.[1]

[1]One can similarly define *rigidly self-crossing* polygons in terms of vertex distance; Ribó Mor's theorem [40] implies that these definitions are equivalent for polygons.

However, this is *not* the definition of "crossing" that most often appears in the computational geometry literature; variants of the following combinatorial definition are much more common [10, 19, 35, 48]. Two polygonal chains $P = [p_0, p_1, \ldots, p_\ell]$ and $Q = [q_0, q_1, \ldots, q_\ell]$ with length $\ell \geq 3$ have a ***forward crossing*** if they satisfy two conditions:

- $p_i = q_i$ for all $1 \leq i \leq \ell - 1$, and
- the cyclic order of p_0, q_0, p_2 around p_1 is equal to the cyclic order of $p_\ell, q_\ell, p_{\ell-2}$ around $p_{\ell-1}$.

Polygonal chains P and Q have a ***backward crossing*** if P and the reversal of Q have a forward crossing. (See Figure 3.1.) Finally, two polygonal chains P and Q ***cross*** if some subpaths of P and Q have a simple crossing, a forward crossing, or a backward crossing.

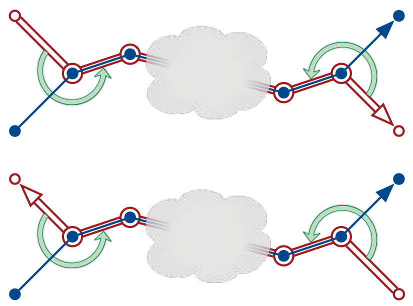

Figure 3.1. A forward crossing and a backward crossing. The paths coincide within the clouds.

These two definitions of "crossing" appear to be equivalent for polygonal chains *without spurs or forks*, but they are not equivalent in general; Figure 3.2 shows three simple examples where the definitions differ. In fact, we know of no combinatorial definition of "crossing" that agrees with our topological definition for arbitrary polygonal chains with spurs.

Even in contexts where this combinatorial definition of "crossing" agrees with our topological definition, it is unclear how to use it to quickly determine whether a polygon is weakly simple. Given a polygon P without spurs, there is a natural algorithm to determine whether P is (combinatorially) self-crossing in $O(n^3)$ time—Find all maximal coincident subpaths via dynamic programming, and then check whether each one is a forward or backward crossing—but this is considerably slower than the algorithms we derive in this paper using the topological definition of "weakly simple".

Several other authors (including the second author of this paper) have proposed the following more intuitive definition [22, 45, 50]: Two paths P and Q "cross" if, for arbitrarily small neighborhoods U of some common subpath, path P contains a point in more than one

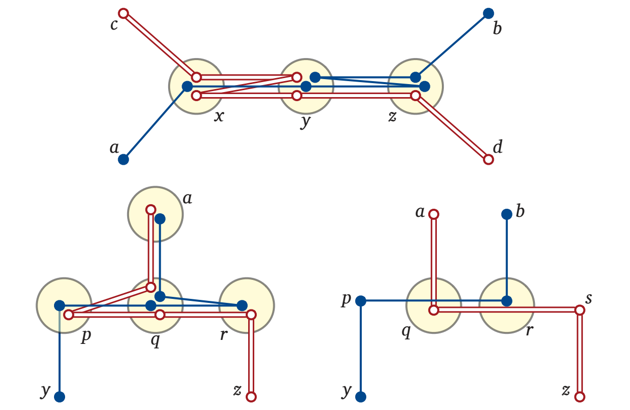

Figure 3.2. Three pairs of crossing paths that do not satisfy the combinatorial definition. Vertices in each small circle coincide; in the bottom right example, vertices p, q, r, s are collinear.

component of $\partial U \setminus Q$; see Figure 3.3(a). This definition is correct when P and Q are *simple*, but it yields both false positives and false negatives for non-simple paths, even without spurs. For example, the paths $[w, x, y, z, x, y]$ and $[z, x, y, z, x, w]$ in Figure 3.3(b) cross but do not satisfy this definition (because P has *no* points in $\partial U \setminus Q$), and the paths $[a, x, y, z, b]$ and $[c, x, y, z, d]$ in Figure 3.3(c) do not cross but satisfy this definition (because small neighborhoods of the common subpath are not simply connected).

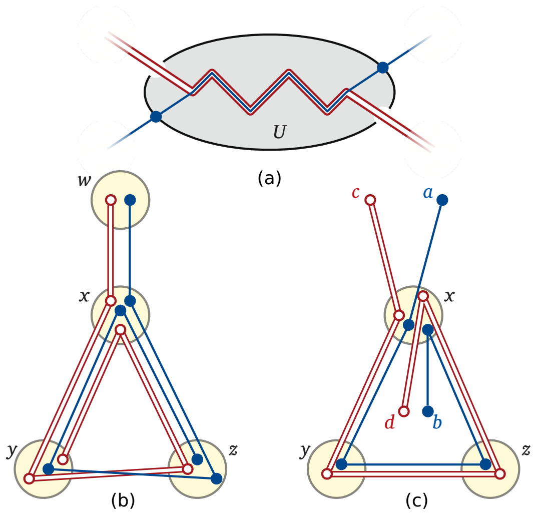

Figure 3.3. (a) A common intuitive definition of "crossing". (b) Two crossing paths that do not satisfy this definition. (c) Two non-crossing paths that do satisfy this definition. Vertices in each small circle coincide.

3.2 Weak Simplicity

Avoiding self-crossings is a *necessary* condition for a polygon to be weakly simple, but it is not sufficient. For example, polygon that wraps 3 times around triangle (as considered by Meister [37]; see Figure 3.4) and the polygon $(a, x, b, x, c, x, a, x, b, x, c, x)$ in Figure 1.1 are neither self-crossing nor weakly simple.

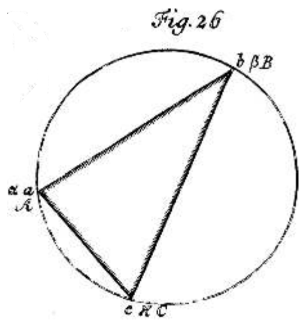

Figure 3.4. Meister's enneagon $[a, b, c, \alpha, \beta, \kappa, A, B, C]$ is neither weakly simple nor self-crossing.

Toussaint [10, 45, 48, 49] defines a polygon to be weakly simple if its *rotation number* is either $+1$ or -1 and any pair of points splits the polygon into two polygonal chains that do not cross. Here, the ***rotation number*** of a polygon is the sum of the signed external angles at the vertices of the polygon, divided by 2π, where each external angle is normalized to the interval $(-\pi, \pi)$. Unfortunately, the rotation number of a polygon with spurs is undefined, since there is no way to determine locally whether the external angle at a spur should be π or $-\pi$. As a result, Toussaint's definition can only be applied to polygons without spurs.

Even for polygons without spurs, there is a more subtle problem with Toussaint's definition. Consider the 14-vertex polygon $(a, b, c, a, b, c, a, x, y, z, x, y, z, x)$ shown in Figure 3.5. This polygon contains exactly two crossings: one between subpaths $[x, a, b, c, a, b]$ and $[c, a, b, c, a, x]$, and the other between subpaths $[a, x, y, z, x, y]$ and $[z, x, y, z, x, a]$. However, both pairs of crossing subpaths overlap not just geometrically, but combinatorially, as substrings of the polygon's vertex sequence. In short, a polygon (or polygonal chain) can cross itself without being divisible into two paths that cross each other! Demaine and O'Rourke's definition of a self-crossing linkage configuration [19] has the same problem. (Their definition also does not consider the possibility of backward crossings.)

Several other papers offer definitions of "weakly simple" that are overly restrictive [7, 8, 29, 36, 42]. For example, the LEDA software library [36] defines a polygon to be weakly simple if it has coincident vertices but otherwise disjoint edges; Bose and Morin [7,8] define a polygon to be weakly simple if the graph G defined by its vertices and edges is plane, the outer face of G is a cycle, and one bounded face of G is adjacent to all vertices. It

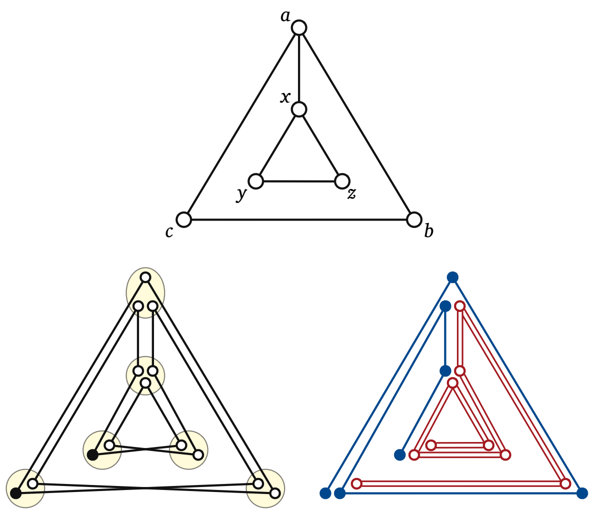

Figure 3.5. A polygon $(a, b, c, a, b, c, a, x, y, z, x, y, z, x)$ that is not weakly simple, even though its rotation number is 1 and every pair of vertices splits the polygon into two paths that do not cross.

is straightforward to construct weakly simple polygons that contradict these definitions.

Finally, several recent papers [18, 21, 30, 33] define weak simplicity directly in terms of vertex perturbation—what we call *rigid* weak simplicity—without mentioning any connection to the more general (and arguably more natural) definition in terms of Fréchet distance.

3.3 All Is Not Lost

We emphasize that the *algorithms* in all the papers discussed in this section appear to be correct. The inputs to these algorithms are simple polygons, polygons with holes, or planar straight-line graphs, composed of line segments that intersect only at common endpoints. The algorithms construct weakly simple polygons only as intermediate results, and always with additional structure that is consistent with the definitions in the corresponding papers. More importantly, in each case, the perturbations required to make those polygons simple are implicit in their construction.

4 No Spurs or Forks

In this section, we describe an algorithm to determine whether a polygon without spurs or forks is weakly simple in $O(n \log n)$ time. Although we have not seen a complete description of this algorithm in the literature, it is essentially folklore. Similar techniques were previously used by Reinhart [39], Zieschang [51], and Chillingworth [14] to determine whether a given closed curve in an arbitrary 2-manifold with boundary is homotopic to a simple closed curve, and more recently by Cortese *et al.* [17] to determine clustered planarity of closed walks in plane graphs.

Let $P = (p_0, \ldots, p_{n-1})$ be an arbitrary polygon without spurs or forks. The algorithm consists of three phases. First, we construct the image graph of P, to identify all coincident vertices and edges and to rule out simple crossings between edges of P. Second, we look for simple crossings at the vertices of P. Finally, if there are no simple crossings, we "inflate" P into a nearby 2-regular plane graph in the only consistent manner possible, and then check whether the expansion is consistent with P.

The description and analysis of our algorithm use the following *multiplicity functions*: $\boldsymbol{n(u)}$ denotes the number of times the point u occurs as a vertex of P; $\boldsymbol{n(uv)}$ denotes the number of times the line segment uv occurs (in either orientation) as an edge of P; and $\boldsymbol{n(uvw)}$ denote the number of times the corner $[u, v, w]$ or its reversal $[w, v, u]$ occurs in P.

4.1 Constructing the Image Graph

In the first phase, we determine in $O(n \log n)$ time whether any two edges of P cross, using the classical sweep-line algorithm of Shamos and Hoey [44]; if so, we immediately halt and report that P is not weakly simple. Otherwise, the image of P is a planar straight-line graph G, whose vertices we call ***nodes*** and whose edges we call ***segments***, to distinguish them from the vertices and edges of P. Specifically, the nodes of G are obtained from the vertices of P by removing duplicates, and the segments of G are obtained from the edges of P by removing duplicates. The image graph G can be constructed in $O(n \log n)$ time using, for example, a straightforward modification of Shamos and Hoey's sweep-line algorithm. This is the most time-consuming portion of our algorithm; the other two phases each require only $O(n)$ time.[2]

4.2 Node Expansion

In the second phase, since we have already ruled out simple crossings in the interiors of edges, any simple crossing in P must consist of two corners $[u, v, w]$ and $[x, v, y]$ whose endpoints have the cyclic order u, x, w, y around their common middle vertex v. We can detect all such crossings in $O(n)$ time using an operation we call ***node expansion***, defined geometrically as follows. (Cortese *et al.* [17] call this operation a *cluster expansion*.)

Let u be an arbitrary node in G. Consider a disk D_u of radius δ centered at u, with δ chosen sufficiently small that D_u intersects only the segments of G incident to u.

[2]We conjecture that the image graph of a weakly simple polygon can actually be constructed in $O(n \log^* n)$ or even $O(n)$ time, by a suitable modification of algorithms for triangulating simple polygons [2, 13, 15, 20, 43].

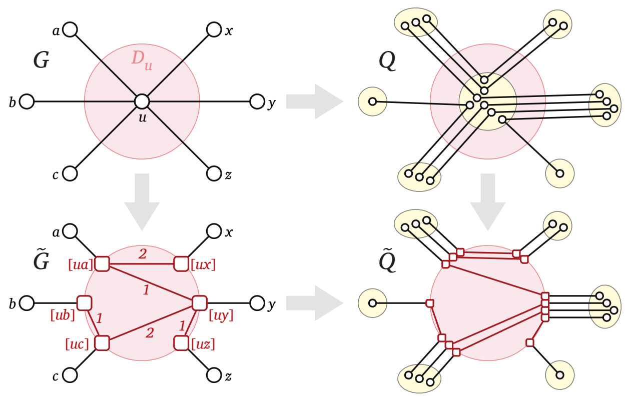

Figure 4.1. Left column: Expanding a node in G. Top row: If P is weakly simple, there is a simple polygon Q arbitrarily close to P. Right column: Modifying Q to mirror the node expansion. Bottom row: Inflating $\tilde{G}$ into a simple 2-regular plane graph $\tilde{Q}$.

Because segments of G are straight line segments, the circle ∂D_u intersects each segment at most once. On each segment ux incident to u, we introduce a new node $[ux]$ at the intersection point $ux \cap \partial D_u$. Then we modify P by replacing each maximal subpath in D_u with a straight line segment. Thus, if P contains the subpath $[x,u,y]$, the modified polygon contains the edge $[ux][uy]$. (We emphasize here that because P has no spurs, points x and y must be distinct and therefore $[ux]$ and $[uy]$ are distinct as well.) Let $\tilde{P}$ denote the modified polygon, and let $\tilde{G}$ denote the image of $\tilde{P}$ in the plane; see the left column of Figure 4.1.

In Section 4.4, we prove that the original polygon P is weakly simple if and only if the expanded polygon $\tilde{P}$ is weakly simple. If P has a simple crossing at u, then some pair of edges of $\tilde{P}$ cross inside D_u. We describe how to quickly check for such crossings in Section 4.5.

Altogether, expanding node u and checking for simple crossings requires $O(n(u))$ time; thus, we can expand *every* node of G in $O(n)$ time overall. If any node expansion creates a simple crossing, we halt immediately and report that P is not weakly simple. Otherwise by induction, $\tilde{G}$ is a plane graph, and P is weakly simple if and only if $\tilde{P}$ is weakly simple.

4.3 Global Inflation

In the final phase of the algorithm, we "inflate" the image graph $\tilde{G}$ into a 2-regular plane graph $\tilde{Q}$ by replacing each segment s of $\tilde{G}$ with several parallel segments, one for each edge of $\tilde{P}$ that traverses s; see the bottom row of Figure 4.1. Then $\tilde{P}$ (and therefore P) is weakly simple if and only if $\tilde{Q}$ is connected and consistent with $\tilde{P}$.

Fix an arbitrarily small positive real number $\varepsilon \ll \delta$. To construct $\tilde{Q}$ from $\tilde{G}$, we replace each node $[uv]$ with a sequence of $n(uv)$ closely spaced points $[uv]_1, [uv]_2, \ldots, [uv]_{n(uv)}$ on ∂D_u, all within ε of the original node $[uv]$. Then we replace each segment $[uv][vu]$ of $\tilde{G}$ outside the disks with $n(uv)$ parallel segments. Finally, within each disk D_u, we replace each corner segment $[uv][uw]$ with $n(uvw)$ parallel segments $[uv]_i[uw]_j$, so that all resulting segments in D_u are disjoint.

Crucially, once the points $[uv]$ on the boundary of D_u are fixed, there is exactly one way to perform this final replacement within D_u so that the resulting segments are consistent with $\tilde{P}$. Moreover, we can find these replacement segments in $O(n(u))$ time as follows. Let $\tilde{G}_u$ be the restriction of $\tilde{G}$ to the disk D_u, together with arcs of ∂D_u between successive nodes. Because $\tilde{G}_u$ is outerplanar, its dual is an apex graph over a tree T, where the apex node is dual to the outer face. Subdivide the edges of T, replacing the edge dual to $[uv][uw]$ with a path of length $n(vuw)$, to obtain a new tree T'. The replacement segments in $\tilde{Q}$ are dual to the subdivided edges of T'. See [6, 11, 12] for related constructions.

The uniqueness of $\tilde{Q}$ implies $\tilde{P}$ is weakly simple if and only if (1) the resulting 2-regular graph $\tilde{Q}$ is connected and thus a simple polygon, and (2) we have $d_V(\tilde{P},\tilde{Q}) < \varepsilon$. We can check whether $\tilde{Q}$ is connected in $O(n)$ time by depth-first search. Finally, if $\tilde{Q}$ is a simple polygon, we can determine whether $d_V(\tilde{P},\tilde{Q}) < \varepsilon$ in $O(n)$ time using any fast string-matching algorithm.

Theorem 4.1. *Given an arbitrary n-vertex polygon P without spurs or forks, we can determine in $O(n \log n)$ time whether P is weakly simple.*

4.4 Expansion Proofs

As promised, we now prove that node expansions preserve weak simplicity.

Lemma 4.2 (Cortese *et al.* [17, Lemma 2]). *Let P be an arbitrary polygon, and let $\tilde{P}$ be the result of a single node expansion. If $\tilde{P}$ is weakly simple then P is also weakly simple.*

Proof: Suppose $\tilde{P}$ is weakly simple. Fix a sufficiently small positive real number $\varepsilon < \delta$. By Theorem 2.1, there is a simple polygon $\tilde{Q}$ such that $d_{\mathcal{F}}(\tilde{P},\tilde{Q}) \le d_V(\tilde{P},\tilde{Q}) < \varepsilon$. (See Figure 4.1.) We easily observe that $d_{\mathcal{F}}(P,\tilde{P}) < \delta$. Thus, the triangle inequality implies

$$d_{\mathcal{F}}(P,\tilde{Q}) \;<\; d_{\mathcal{F}}(P,\tilde{P}) + d_{\mathcal{F}}(\tilde{P},\tilde{Q}) \;<\; \delta + \varepsilon \;<\; 2\delta.$$

Because δ is arbitrarily small, we conclude that P is weakly simple. □

To prove the converse of Lemma 4.2, we actually consider a more general ***expansion*** operation, defined as follows. Let $P = (p_0, p_1, \ldots, p_{n-1})$ be an arbitrary polygon, and let D be an arbitrary elliptical disk that intersects P *transversely*; that is, the boundary ellipse ∂D intersects at least one edge of P, is not tangent to any edge of P, and does not contain any vertex of P. To ***expand P inside D***, we subdivide the edges of P that intersect ∂D by introducing new vertices at the intersection points, and then replace each maximal subpath of P inside D with a straight line segment between its endpoints on ∂D. Finally, we remove any boundary-to-boundary segments with length zero by merging their coincident endpoints.

Lemma 4.3. *Let P be an arbitrary polygon; let D be an elliptical disk whose boundary intersects P transversely; and let $\tilde{P}$ be the result of expanding P inside D. If P is weakly simple, then $\tilde{P}$ is also weakly simple.*

Proof: Suppose polygon $P = (p_0, p_1, \ldots, p_{n-1})$ is weakly simple. By Theorem 2.1, for any real number $\varepsilon > 0$, there is a simple polygon $Q = (q_0, q_1, \ldots, q_{n-1})$ such that $d_V(P, Q) < \varepsilon$. If ε is sufficiently small, ∂D also intersects Q transversely; moreover, ∂D intersects an edge of Q if and only if it intersects the corresponding edge of P in the same number of points. Let $\tilde{Q}$ be the polygon obtained by expanding Q inside D. See Figure 4.2.

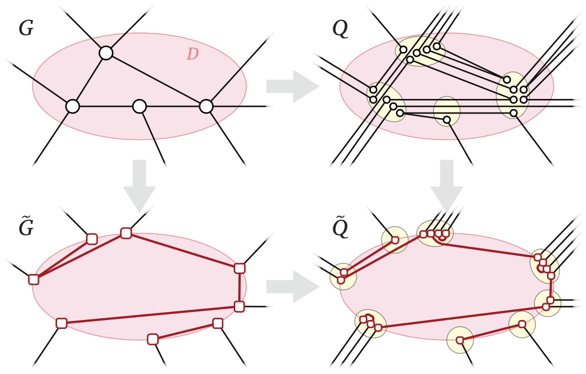

Figure 4.2. Left column: Expanding the image graph G. Top row: If P is weakly simple, there is a nearby simple polygon Q. Right column: Expanding Q in the same ellipse yields a simple polygon $\tilde{Q}$. Bottom row: $\tilde{Q}$ is close to $\tilde{P}$. Compare with Figure 4.1.

Because Q is simple, the intersection $Q \cap D$ consists of disjoint simple subpaths. For any two such subpaths α and β, the endpoints of α must lie in one component of $\partial D \setminus \beta$, and thus the corresponding line segments $\tilde{\alpha}$ and $\tilde{\beta}$ of $\tilde{Q}$ are also disjoint. It follows that $\tilde{Q}$ is also simple.

Consider a vertex $\tilde{p}$ of $\tilde{P}$, located at the intersection of an edge $p_i p_{i+1}$ of P and the ellipse ∂D. Let θ denote the angle between $p_i p_{i+1}$ and (the line tangent to) ∂D at $\tilde{p}$. If ε is sufficiently small, the corresponding edge $q_i q_{i+1}$ of Q intersects ∂D at a point $\tilde{q}$ such that $d(\tilde{p}, \tilde{q}) < 2\varepsilon / \sin\theta$. It follows that $d_V(\tilde{P}, \tilde{Q}) < 2\varepsilon / \sin\theta^*$, where θ^* is the minimum angle of intersection between P and ∂D.

Thus, for any $\delta > 0$, we obtain a simple polygon $\tilde{Q}$ such that $d_{\mathcal{F}}(\tilde{P}, \tilde{Q}) < \delta$ by setting $\varepsilon < (\delta/2)\sin\theta^*$. We conclude that $\tilde{P}$ is weakly simple. □

Unfortunately, the converse of Lemma 4.3 is not true in general. For example, the polygon (w, x, u, y, z, u, v, u) in Figure 4.3 is not weakly simple, because it contains two subpaths $[w, u, v]$ and $[x, u, z]$ that cross transversely at u, but expanding it inside a disk D around segment uv produces the weakly simple polygon $(w, x, [ux], [uy], y, z, [uz], [uw])$, where for example $[ux]$ denotes the point $ux \cap \partial D$.

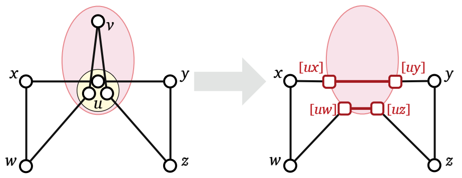

Figure 4.3. Careless expansion of a polygon that is not weakly simple can produce a weakly simple polygon. (The three vertices in the small circle actually coincide.)

4.5 Implementation and Planarity Checking

Given a polygon P and an ellipse D, it is straightforward to compute the polygon $\tilde{P}$ resulting from expanding P inside D in $O(n)$ time by brute force. For *arbitrary* expansions, computing and sorting the coordinates of the intersection points with the ellipse ∂D requires some care. In our algorithms, however, all expansion operations can be performed combinatorially, *with no numerical computation whatsoever*. The actual size and shape of the disk D is completely immaterial to our algorithms; our geometric description in terms of ellipses is intended only to provide intuition and simplify our proofs of correctness. In particular, we never need to choose explicit values for the parameters δ and ε.

Our algorithms maintain a representation of the polygon P that allows us to compute the sequence of points where P crosses ∂D, in cyclic order around D, in constant time per intersection point. Specifically, for the node expansions applied in this section, this sequence can be extracted directly from the rotation system of the image graph G, which records the cyclic order of segments incident to each node. (For our later applications of expansion, extracting the sorted sequence of points around ∂D is only slightly more complex.) Given this sequence of points, which become the new vertices of $\tilde{P}$, we can perform the rest of the expansion in $O(m)$ time, where m is the number of edges of P that intersect D.

If P is *not* weakly simple, the expanded polygon $\tilde{P}$ may include pairs of edges that cross transversely. Let G_D

denote the graph whose nodes are the intersection points $\operatorname{im} P \cap \partial D$ and whose segments are distinct edges of $\tilde{P}$ inside D and arcs of ∂D between vertices in cyclic order. Polygon $\tilde{P}$ contains crossing edges if and only if G_D is not a plane graph.

We can determine the planarity of G_D as follows. First, add a new "apex" node and connect it by segments to each of the nodes of G_D. The resulting abstract graph G_D^+ is 3-connected, and therefore has at most one planar embedding. Moreover, G_D^+ is planar if and only if there is a planar embedding of G_D with all nodes on a single face (which we take to be the outer face) in the correct cyclic order. Thus, G_D is a *plane* graph if and only if G_D^+ is a *planar* graph; there are several linear-time algorithms to determine whether a graph is planar [9, 25, 32, 46]. Crudely, G_D has at most $O(m)$ nodes and segments, so the planarity check takes at most $O(m)$ time.

5 Forks But No Spurs

Recall that a ***fork*** in a polygon is a vertex that lies in the interior of an edge. With only minor modifications, the algorithm in the previous section can also be applied to polygons with forks, but still without spurs. Specifically, in a preprocessing phase, we find all incident vertex-edge pairs in $O(n \log n + k)$ time, where $k = O(n^2)$ is the number of such pairs, using a simple modification of the standard Bentley-Ottmann sweep-line algorithm [4]. Whenever we discover an incident vertex-edge pair, we subdivide that edge at the incident vertex in $O(1)$ additional time. The remainder of the weak simplicity algorithm is unchanged. In the worst case, subdividing the polygon to eliminate forks increases its complexity from n to $\Theta(n^2)$, which in turn increases the overall running time of our algorithm from $O(n \log n)$ to $O(n^2 \log n)$. With more care, however, we can avoid this quadratic blowup.

We define a coarser decomposition of the image of P into points and line segments called a ***bar decomposition***, as follows. A ***bar*** of P is a component of the union of the interiors of all edges of P that lie on a common line. Every fork lies in exactly one bar; we call any vertex that is not in a bar ***sober***. The bar decomposition consists of all bars of P and all distinct sober vertices of P; the image of P is equal to the union of these bars and points. If P has no forks, the bar decomposition consists of the nodes and segments of the image graph G.

We can compute the bar decomposition of P in $O(n \log n)$ time as follows. First, cluster the edges of P into collinear subsets, by sorting them by the slopes and y-intercepts of their supporting lines. Then for each collinear subset, sort the endpoints by x-coordinate, breaking ties by sorting all right endpoints before all left endpoints. Finally, a linear-time scan along each sorted

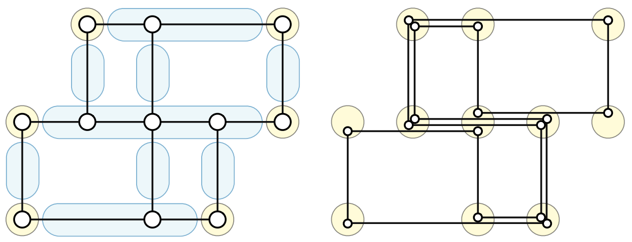
Figure 5.1. A bar decomposition of a weakly simple polygon without spurs, and a nearby simple polygon.

collinear subset of edges yields all bars that lie on that line. We also identify all distinct sober vertices, compute (the indices of) all vertices and edges of P that lie in each bar, and compute the bars incident to each fork in cyclic order. Finally, we verify that no pair of bars crosses, using a standard sweep-line algorithm [44].

Next we perform a node expansion at each sober node, as described in the previous section. We also perform a ***bar expansion*** around each bar, defined as follows. For any bar b, let b° denote the subset of b obtained by removing all points within distance 2δ of endpoints of b, and let D_b denote an elliptical disk whose major axis is b° and whose minor axis has length 2δ, where the parameter δ is chosen so that D_b intersects only the edges of P that also intersect the bar. We emphasize that the endpoints of the bar b are *outside* D_b. Then bar expansion is just expansion inside the disk D_b, as described in Section 4.4. That is, we subdivide P at the intersection points $\operatorname{im} P \cap \partial D_b$, replace each maximal subpath of P inside D_b with a line segment, and finally verify that these new segments do not cross, as described in Section 4.5.

Again, there is no need to choose a specific value of δ; each bar expansion is performed entirely combinatorially. In particular, the sorted sequence of new nodes around ∂D_b can be extracted in $O(1)$ time each from the order of forks along b and the cyclic order of bars ending at each fork in b; both of these orders are computed as part of the bar decomposition. Since every edge of P touches at most three bars or sober nodes, we can expand every bar and every sober node in $O(n)$ time. The resulting polygon $\tilde{P}$ has no simple crossings, no forks, and no spurs. Lemmas 4.2, 4.3, and 5.1 (proved below) imply inductively that $\tilde{P}$ is weakly simple if and only if the original polygon P is weakly simple. Finally, we can determine whether $\tilde{P}$ is weakly simple in $O(n)$ time using the global inflation algorithm described in Section 4.3.

It remains only to prove that bar expansion properly preserves weak simplicity.

Lemma 5.1. *Let P be an arbitrary polygon* ***without spurs****, and let $\tilde{P}$ be the result of a single bar expansion. P is weakly simple if and only if $\tilde{P}$ is weakly simple.*

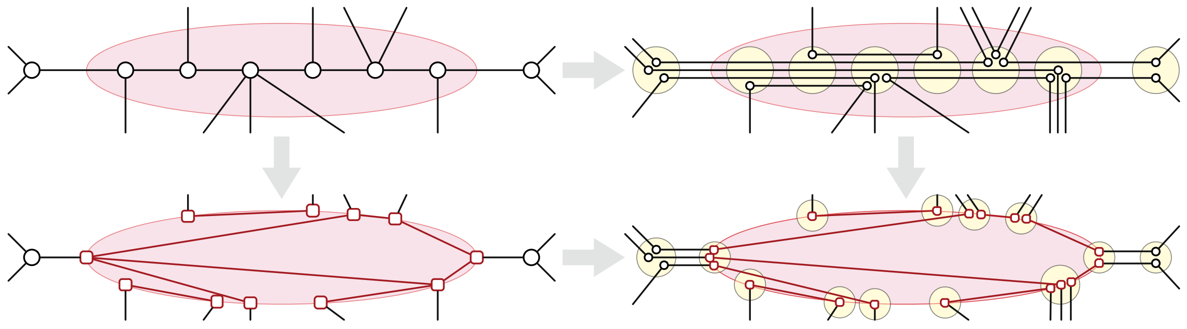

Figure 5.2. Bar expansion. Compare with Figure 4.1.

Proof: Let b be the bar around which we are expanding, and let $\theta > 0$ denote the minimum positive angle between b and any edge incident to but not contained in b. Consider a subpath $[u, v, w, x]$ of P such that vw lies in the interior of D_b and therefore in the bar b; vertices u and x must lie outside D_b. Let $\tilde{v} = uv \cap \partial D_b$ and $\tilde{w} = wx \cap \partial D_b$; the line segment $\tilde{v}\tilde{w}$ is an edge of the expanded polygon $\tilde{P}$. We easily observe that $d(v, \tilde{v}) \le \delta / \sin \angle uvw \le \delta / \sin \theta$ and $d(w, \tilde{w}) \le \delta / \sin \angle vwx \le \delta / \sin \theta$, which implies that $d_{\mathcal{F}}([u, v, w, x], [u, \tilde{v}, \tilde{w}, x]) \le \delta / \sin \theta$. By similar arguments for edges that share one or both endpoints of b, we conclude that $d_{\mathcal{F}}(P, \tilde{P}) < \delta / \sin \theta$.

If $\tilde{P}$ is weakly simple, there is a simple polygon $\tilde{Q}$ such that $d_{\mathcal{F}}(\tilde{P}, \tilde{Q}) \le d_V(\tilde{P}, \tilde{Q}) < \delta$, which implies

$$d_{\mathcal{F}}(P, \tilde{Q}) \; < \; d_{\mathcal{F}}(P, \tilde{P}) + d_{\mathcal{F}}(\tilde{P}, \tilde{Q}) \; < \; \delta(1 + 1/\sin \theta)$$

by the triangle inequality. Since this Fréchet distance can be made arbitrarily small by shrinking δ, we conclude that P is weakly simple.

Lemma 4.3 immediately implies that if P is weakly simple, then $\tilde{P}$ is weakly simple. □

We emphasize that if the polygon P has spurs, then bar expansions are no longer safe; the resulting polygon $\tilde{P}$ could be weakly simple even though P is not.

Theorem 5.2. *Given an arbitrary n-vertex polygon P without spurs, but possibly with forks, we can determine in $O(n \log n)$ time whether P is weakly simple.*

6 Polygons with Spurs

Neither of the previous algorithms is correct when the input polygon has spurs. To handle these polygons, we apply a recent algorithm of Cortese *et al.* [17] to determine whether a walk in a plane graph is weakly simple (in their terminology, whether a rigid clustered cycle is c-planar). We include a complete description of the algorithm here, reformulated in our terminology, not only to keep this paper self-contained, but also so that we can improve its running time in Section 7.

At a high level, the algorithm proceeds as follows. In an initial preprocessing phase, we construct the image graph G of the input polygon P and then expand P at every node in G. (We must make one small change to the node expansion operation: After expanding around any spur, we remove any zero-length edges. Thus, expanding around a spur u replaces any subpath of the form $[x, u, x]$ with $x[ux]x$ instead of $x[ux][ux]x$.) Then, following Cortese *et al.* [17], we repeatedly modify the polygon using an operation we call ***segment expansion***, defined in Section 6.1, until either we detect a simple crossing (in which case P is not weakly simple), the image graph is reduced to a single line segment (in which case P is weakly simple), or there are no more "useful" segments to expand.

If the input polygon has no forks, we construct the image graph G in $O(n \log n)$ time and then expand every node in $O(n)$ time, exactly as described in Section 4. A simple potential argument, given in Section 6.3, implies that the main loop terminates after at most $O(n)$ segment expansions. (Cortese *et al.* [17] prove an upper bound of $O(n^2)$ expansions using a different potential function.) A naïve implementation executes each segment expansion by brute force in $O(n)$ time, giving us an overall running time of $O(n^2)$.

Theorem 6.1. *Given an arbitrary n-vertex polygon P without forks but possibly with spurs, we can determine in $O(n^2)$ time whether P is weakly simple.*

Any polygon with n vertices can be subdivided at the forks into a polygon with $O(n^2)$ vertices and without forks in $O(n^2)$ time using a standard sweep-line algorithm, as described in Section 5. Thus, Theorem 6.1 has the following immediate corollary.

Corollary 6.2. *Given an arbitrary n-vertex polygon P, we can determine in $O(n^4)$ time whether P is weakly simple.*

For the remainder of the paper, we assume that the input polygon P has no forks.

6.1 Segment Expansion

We now describe the main loop of our algorithm in more detail. The main operation, ***segment expansion***, is nearly identical to our earlier expansion operations. Let s be an arbitrary segment of the current image graph G. Let D_s be an elliptical disk whose boundary intersects precisely the segments that share one endpoint with s. To expand segment s, we subdivide P at its intersection with ∂D_s, replace each maximal subpath of P that lies in D_s with a line segment, and verify that the new segments do not cross.

Unfortunately, as we have already seen in Figure 4.3, *arbitrary* segment expansions could change a polygon that is not weakly simple into a polygon that is weakly simple. To avoid this behavior, our algorithm expands only segments with the following property. A segment uv in the image graph is a ***base*** of one of its endpoints u if every occurrence of u in the polygon P is immediately preceded or followed by the other endpoint v. A segment is ***safe*** if it is a base of both of its endpoints. Our algorithm only performs segment expansions around safe segments.

In Section 6.4 below, we prove that safe segment expansions properly preserve weak simplicity. That is, if $\tilde{P}$ is the result of a safe segment expansion on P, then P is weakly simple if and only if $\tilde{P}$ is weakly simple.

After a node expansion around any node u (as defined in Section 4.2), each new node $[ux]$ on the boundary of D_u has a base, namely the segment $[ux]x$ [17, Property 5]. Thus, expanding every node in the original image graph guarantees that every node in the image graph has a base. Similarly, after any segment expansion, every newly created node has a base. Thus, our algorithm inductively maintains the invariant that every node in the image graph has a base.

It follows that among all the segments incident to any node u, any segment uv with maximum multiplicity $n(uv)$ is a base of u. (If u has two bases uv and uw, those are the only segments incident to u, every corner at u has the form $[v,u,w]$ or $[w,u,v]$, and thus $n(uv) = n(uw) = n(vuw) = n(u)$. No node can have more than two bases.) Therefore, any segment in G with *globally* maximum multiplicity is a base of both of its endpoints. We conclude that at every iteration of our algorithm, the image graph G contains at least one safe segment, and so our algorithm never gets stuck.

6.2 Useful Segments

However, under some circumstances, expanding a safe segment does not actually make progress. Specifically, if both endpoints of the segment have degree 2 in G, and no spur in P includes that segment, the segment expansion does not change the combinatorial structure of P or G at all. We call any such segment ***useless***. Equivalently, a safe segment is ***useful*** if it is the only base of one of its endpoints, and useless otherwise. Our algorithm repeatedly expands only *useful* segments until either the image graph consists of a single segment (in which case P is weakly simple) or there are no more useful segments to expand.

Lemma 6.3. *Let G be the image graph of a polygon P without simple crossings. If every node of G has a base but G has no useful segments, then G is a simple polygon and P has no spurs.*

Proof: Let $m = \max_{uv} n(uv)$ be the maximum multiplicity among all segments of G, and let H be the subgraph of G induced by all segments with multiplicity m, together with their endpoints. (Recall that every segment in H is safe, although there may be safe segments not in H.) The degree of any node in H is at most the number of bases of that node in G. Since no node in G can have more than two bases, H is the union of disjoint cycles and nontrivial paths. If some component of H is a path, the first (or last) segment in that path is useful. Otherwise, every node in G has two bases, and therefore has degree 2; because G is a connected planar straight-line graph, it must be a simple polygon. Moreover, because every node in G has two bases, P has no spurs. □

Lemma 6.3 implies that when our main loop ends, we can invoke our algorithm for spur-free polygons in Section 4. But this is overkill. Because P has no spurs, P must traverse every segment of G the same number of times. It follows that the 2-regular plane graph $\tilde{Q}$ constructed in the inflation phase of the algorithm in Section 4 will consist of $n(uv)$ parallel copies of G. Thus, when the main loop ends, P is weakly simple if and only if $n(uv) = 1$, for any single segment uv.

6.3 Termination and Analysis

Like Cortese *et al.*, we prove that our algorithm halts using a potential argument. Let $|P|$ and $|G|$ respectively denote the number of vertices in polygon P and the number of nodes in its image graph G; our potential function is $\Phi(P,G) := 2|P| - |G|$. Clearly $\Phi(P,G)$ is always non-negative. Performing the initial node expansions at most doubles $|P|$, so at the beginning of the main loop we have $\Phi(P,G) \le 4n$. (Cortese *et al.* used the potential $\sum_s n(s)^2 = O(n^2)$, where the sum is over all segments of G; otherwise, our analysis is nearly identical.)

Lemma 6.4. *Let P be any polygon whose image graph G has more than one segment, let $\tilde{P}$ be the result of expanding a useful segment of G, and let $\tilde{G}$ be the image graph of $\tilde{P}$. Then $\Phi(\tilde{P},\tilde{G}) < \Phi(P,G)$.*

Proof: Let uv be a useful segment of G, where without loss of generality we have $\deg(u) \leq \deg(v)$. Because uv is safe, every maximal subpath of P that lies in the ellipse D_{uv} contains at least 2 vertices, so $|\tilde{P}| \leq |P|$. We easily observe that $|\tilde{G}| = |G| + \deg(u) + \deg(v) - 4$, so if $\deg(u) + \deg(v) > 4$, the proof is complete. There are three other cases to consider:

- If $\deg(u) = \deg(v) = 1$, then uv is the only segment of G, which is impossible.
- Otherwise, if $\deg(u) = 1$, then P must contain the spur $[v, u, v]$, which implies $|\tilde{P}| \leq |P| - 1$ and therefore $\Phi(\tilde{P}, \tilde{G}) \leq \Phi(P, G) - (\deg(v) - 1) \leq \Phi(P, G) - 1$.
- Finally, suppose $\deg(u) = \deg(v) = 2$. Because uv is useful, P must contain one of the spurs $[v, u, v]$ or $[u, v, u]$, which implies $|\tilde{P}| \leq |P| - 1$ and therefore $\Phi(\tilde{P}, \tilde{G}) \leq \Phi(P, G) - 2$. □

Lemma 6.4 immediately implies the main loop of the algorithm ends after at most $4n$ segment expansions. Because the potential Φ decreases at every iteration, the polygon never has more than $4n$ vertices. We can find a useful segment in the image graph (if one exists) and perform a segment expansion in $O(n)$ time by brute force. Thus, a naïve implementation of our algorithm runs in $O(n^2)$ time.

6.4 Safe Segment Expansion

As promised, we now prove that expanding around a safe segment properly preserves weak simplicity. This key lemma was was previously proved by Cortese *et al.* [17, Lemma 3] using different terminology and techniques; to keep this paper relatively self-contained, we provide a new geometric proof.

To simplify our argument, we imagine that each segment expansion is preceded by a ***spur reduction***, which replaces any subpath $[u, v, \ldots, u, v]$ that alternates between nodes u and v at least twice with the single edge $[u, v]$. For example, the subpath $[a, u, v, u, v, u, b]$ would become a single spur $[a, u, v, u, b]$, and the subpath $[a, u, v, u, v, u, v, u, v, z]$ would become a simple subpath $[a, u, v, z]$.

Lemma 6.5. *Let P be any polygon, and let $\bar{P}$ be the result of a spur reduction on some segment uv of its image graph. If $\bar{P}$ is weakly simple, then P is weakly simple.*

Proof: It suffices to consider the case where $\bar{P}$ is obtained from P by replacing exactly one subpath $[a, u, v, u, v, z]$ with the simpler subpath $[a, u, v, z]$, for some nodes $a \neq v$ and $z \neq u$. The argument for paths of odd length is similar, and the general case then follows by induction.

Suppose $\bar{P}$ is weakly simple. Then by Theorem 2.1 for any $\varepsilon > 0$, there is a polygon $\bar{Q}$ with $d_V(\bar{P}, \bar{Q}) < \varepsilon$. Let $[a', u', v', z']$ be the subpath of $\bar{Q}$ corresponding to the replacement subpath $[a, u, v, z]$ of $\bar{P}$. If ε is sufficiently small, we can find four points u_1, u_2, v_1, v_2 with the following properties:

- $d(u_1, u') = d(u_2, u') = d(v_1, v') = d(v_2, v') = \varepsilon$,
- $d(u_1, u'v') = d(u_2, u'v') = \varepsilon^2 = d(v_1, u'v') = d(v_2, u'v') = \varepsilon^2$,
- the path $[u', u_1, v_1, u_2, v_2, v']$ is simple and intersects $\bar{Q}$ only at the edge $u'v'$.

Let Q be the simple polygon obtained by replacing the edge $[u', v']$ with $[u', u_1, v_1, u_2, v_2, v']$. Then we have $d_{\mathcal{F}}(P, Q) < 2\varepsilon$ by the triangle inequality, which implies that P is weakly simple. □

Lemma 6.6. *Let P be a spur-reduced polygon with more than two distinct vertices, and let $\tilde{P}$ be the result of a safe segment expansion. P is weakly simple if and only if $\tilde{P}$ is weakly simple.*

Proof: Let uv be the safe segment around which we are expanding, and let θ be the smallest positive angle between uv and any other segment incident to u or v. By the previous lemma, we can assume without loss of generality that P does not contain the subpath $[u, v, u, v]$.

Suppose $\tilde{P}$ is weakly simple. Fix a real number $\varepsilon \ll \delta/2n$, and let $\tilde{Q}$ be a simple polygon such that $d_V(\tilde{P}, \tilde{Q}) < \varepsilon$, as guaranteed by Theorem 2.1. If P has no spurs at uv, the proof of Lemma 5.1 implies that $d_{\mathcal{F}}(P, \tilde{Q}) < \delta(1 + 1/\sin\theta)$ and we are done. However, if P has a spur at uv, the Fréchet distance between P and $\tilde{Q}$ is approximately the length of uv. In this case, we iteratively modify $\tilde{Q}$ into a new simple polygon Q such that $d_{\mathcal{F}}(P, Q) < \delta/\sin\theta + \varepsilon < \delta(1 + 1/\sin\theta)$.

Suppose P has k distinct spurs at uv. For each integer i from 0 to k, let D_i be the elliptical disk concentric with D_{uv} but whose axes are shorter by $2(i+1)\varepsilon$. For example, the major axis of D_0 has length $|uv| + 2\delta - 2\varepsilon$ and the minor axis has length $2\delta - 2\varepsilon$. Every vertex of $\tilde{Q}$ lies outside each disk D_i, and if ε is sufficiently small, every edge of $\tilde{Q}$ that intersects D also intersects each disk D_i.

We iteratively define a sequence of polygons $\tilde{Q} = Q_0, Q_1, \ldots, Q_k$ as follows. Fix an index $i \geq 0$. Let U_i and V_i denote the subsets of ∂D_i within distance $\delta/\sin\theta$ of u and v, respectively. If δ is sufficiently small, the elliptical arcs U_i and V_i are disjoint. Every segment in $\tilde{Q}_i \cap D_i$ has endpoints in $U_i \cup V_i$. We call a segment in $\tilde{Q}_i \cap D_i$ a ***left segment*** if both endpoints are in U_i, or a ***right segment*** if both endpoints are in V_i. Every left segment corresponds to a subpath of P of the form $[a, u, v, u, b]$, and every right segment corresponds to a subpath of P of the form $[y, v, u, v, z]$.

Suppose $\tilde{Q}_i \cap D_i$ includes at least one left segment; right segments are handled symmetrically. Then some component R_i of $D_i \setminus \tilde{Q}_i$ has both a left segment and an arc of V_i on its boundary. Suppose the left segment corresponds to the subpath $[a, u, v, u, b]$ of P. Then $\tilde{P}$ contains the subpath $[a, [ua], [ub], b]$, where $[ua] = ua \cap \partial D_{uv}$ and $[ub] = ub \cap \partial D_{uv}$, polygon $\tilde{Q}$ contains an edge $a'b'$ such that $d(a', [ua]) < \varepsilon$ and $d(b', [ub]) < \varepsilon$, and finally $a_i b_i = a'b' \cap D_i$ is the left segment in question. Fix a point $z_i \in R \cap V_i$, and let $\tilde{Q}_{i+1}$ be the simple polygon obtained by replacing the line segment $a_i b_i$ with the path $[a_i, z_i, b_i]$.

Arguments in the proof of Lemma 5.1 imply that $d(u, a') < \delta / \sin\theta + \varepsilon$ and $d(u, b') < \delta / \sin\theta + \varepsilon$, which in turn imply that $d(u, a_i) < \delta / \sin\theta + \varepsilon$ and $d(u, b_i) < \delta / \sin\theta + \varepsilon$. It now follows that

$$d_{\mathcal{F}}\big([[ua], u, v, u, [ub]],\ [a', a_i, z_i, a_i, b']\big) < \delta / \sin\theta + \varepsilon.$$

The final polygon Q_k has no left or right segments in D_k; thus, every spur in P is within Fréchet distance $\delta / \sin\theta + \varepsilon$ of the corresponding path in Q_k. We conclude that $d_{\mathcal{F}}(P, Q_k) < \delta / \sin\theta + \varepsilon$, which implies that P is weakly simple.

Finally, Lemma 4.3 implies that if P is weakly simple, then $\tilde{P}$ is weakly simple, completing the proof. □

7 Fast Implementation

Finally, we describe a more careful implementation of the previous algorithm that runs in $O(n \log n)$ time, again assuming that the input polygon has no forks. At a high level, the algorithm is identical to the previous section. We build the image graph G and perform the initial node expansions in $O(n \log n)$ time, and then repeatedly expand useful segments until either we discover a simple crossing or there are no more useful segments. The only change is that we maintain a collection of simple data structures that allow us to perform each segment expansion more quickly, as described in Sections 7.1 and 7.2. The final $O(n \log n)$ running time follows from a heavy-path decomposition argument [28, 47] in Section 7.3.

Theorem 7.1. *Given an arbitrary n-vertex polygon P without forks but possibly with spurs, we can determine in $O(n \log n)$ time whether P is weakly simple.*

Corollary 7.2. *Given an arbitrary n-vertex polygon P, we can determine in $O(n^2 \log n)$ time whether P is weakly simple.*

7.1 Data Structures

Our improved algorithm uses the following data structures. We maintain the image graph G in a standard data structure for planar straight-line graphs, such as a quad-edge data structure [27] or a doubly-connected edge list [5]. The polygon P is represented by a circular doubly-linked list that alternates between vertex records and edge records; each vertex in P points to the corresponding node in G.

Call an edge of P ***simple*** if neither of its endpoints is a spur and ***complex*** otherwise. For each segment uv, we separately maintain a doubly-linked list $\mathit{Simple}(uv)$ of all simple edges of P that coincide with uv, and a doubly-linked list $\mathit{Complex}(uv)$ of all complex edges of P that coincide with uv. Every record in these lists has pointers both to and from the corresponding edge record in the circular list representing P. Each of these lists also maintains its size.

Each node u in G also maintains its multiplicity $n(u)$ and pointers to its base(s). Finally, we maintain a global queue of all useful segments in G. All of these data structures can be constructed in $O(n)$ time after the preprocessing phase.

7.2 Fast Segment Expansion

To simplify notation, let $D = D_{uv}$. Our segment expansion algorithm divides each segment expansion into four phases: (1) remove all spurs at the endpoints u and v; (2) compute the nodes $[ua]$ and $[vz]$ in cyclic order around the ellipse ∂D; (3) straighten the remaining paths through u and through v; (4) build the graph G_D, check for simple crossings, and update the image graph G. Our analysis uses the following functions of u (and similar functions of v), all defined just before the segment expansion begins:

- $\mathbf{deg}(\boldsymbol{u})$ is the number of segments incident to u, including uv.
- $\boldsymbol{n(u)}$ is the number of vertices of P that coincide with u.
- $\boldsymbol{\sigma(u)}$ is the number of spurs of P that coincide with u.

Phase 1: Remove spurs. Because uv is a safe segment, all spurs at u and v occur in subpaths of the form $[u, v, u]$ or $[v, u, v]$. We remove all spurs at u and v by brute force, by traversing uv's list of complex edges. Each time we remove a spur σ, we check the edges of P immediately before and after σ, and if necessary, move them from $\mathit{Simple}(uv)$ to $\mathit{Complex}(uv)$ or vice versa. We also update the multiplicities $n(u)$ and $n(v)$. After this phase, every maximal subpath of P inside the disk D_{uv} has length at most 3. The total time for this phase is $O(\sigma(u) + \sigma(v) + 1)$.

Phase 2: Compute sequence of new nodes. Next, we compute the intersection points $G \cap \partial D$ in cyclic order by considering the segments incident to u in cyclic order, starting just after uv, and then the segments incident to v in cyclic order, starting just after uv. These two cyclic orders are part of the rotation system of G, which is directly accessible from the graph data structure. For each intersection point $[ua]$ or $[vz]$, we initialize a new node record with a pointer to its base $[ua]a$ or $[vz]z$. At this point, we can update the queue of useful segments.

Finally, we find a segment ua^* with maximum multiplicity $n([ua^*])$ and a segment vz^* with maximum multiplicity $n([vz^*])$, such that $a^* \neq v$ and $z^* \neq u$. To simplify notation, let u' denote the point $[ua^*] = ua^* \cap \partial D$ and let v' denote the point $[vz^*] = vz^* \cap \partial D$. The total time for this phase is $O(\deg(u) + \deg(v))$.

Phase 3: Expansion. The third phase straightens the remaining constant-length paths through u and v *except* for subpaths that start with a^* or end with z^*. Specifically, for each segment ua with $a \neq a^*$, we replace subpaths of P containing segment ua with corresponding paths through the new node $[ua]$ as follows:

- $[a, u, v]$ becomes $[a, [ua], v]$,
- $[a, u, a]$ becomes $[a, [ua], a]$,
- $[a, u, b]$ becomes $[a, [ua], [ub], b]$ for any other node b.

Then for each segment vz with $z \neq z^*$, we similarly replace subpaths of P that contain vz with paths through the new node $[vz]$. Each path replacement takes $O(1)$ time, including the time to update the relevant simple- and complex-edge lists.

At the end of these two loops, the only remaining subpaths through u and v have the forms $[a^*, u, v, y]$ or $[a^*, u, b]$ or $[b, u, v, z^*]$ or $[y, v, z^*]$. To update these subpaths, we intuitively *move* node u to u' and move node v to v'. In fact, "moving" these two nodes has no effect on our data structures at all; we only change their *names* for purposes of analysis.

The total time for this phase is at most $O(n(u) - n(u') + n(v) - n(v') + 1)$, where $n(u')$ and $n(v')$ denote the number of vertices at u' and v' *after* the segment expansion.

Phase 4: Check planarity and update *G*. We discover all the segments in the graph G_D in the previous phase. If there are more than $2(\deg(u) + \deg(v))$ such segments, then G_D cannot be planar, so we immediately halt and report that P is not weakly simple. Otherwise, we compute the rotation system of G_D and check its planarity in $O(\deg(u) + \deg(v))$ time, as described in Section 4.5. If G_D is a plane graph, we splice it into the image graph G, again in $O(\deg(u) + \deg(v))$ time.

7.3 Time Analysis via Heavy-Path Decomposition

The analysis in the previous section implies that the total time for a single edge expansion is at most

$$O(\sigma(u) + \sigma(v) + 1) + O(\deg(u) + \deg(v)) \\ + O(n(u) - n(u') + n(v) - n(v') + 1).$$

We can charge the $+1$s in the first and third terms to the second term. Expanding any segment uv decreases the number of vertices of P by at least $\sigma(u) + \sigma(v)$. It follows that $\sum_{uv}(\sigma(u) + \sigma(v)) = O(n)$, where the sum is taken over all expanded segments uv.

For any segment uv, we have $\deg(u) \leq 2(n(u) - n(u')) + 2$ and $\deg(v) \leq 2(n(v) - n(v')) + 2$. Moreover, if uv is a useful segment, we also have $n(u) + n(v) > n(u') + n(v')$, which implies

$$\deg(u) + \deg(v) \leq 6(n(u) - n(u') + n(v) - n(v')).$$

Thus, to bound the overall running time of our algorithm, it remains only to bound the sum

$$\sum_{uv}(n(u) - n(u') + n(v) - n(v')).$$

For purposes of analysis, we define a ***family tree T*** of all nodes that our algorithm ever creates. The root of T is a special node r, whose children are the nodes in the initial image graph (after node expansion). The children in T of any other node u are the new nodes $[ua]$ created during the segment expansion that destroys u, together with u'. (Likewise, the children of v are the new nodes $[vz]$ together with v'.) Each node u in T has weight $n(u)$, which is the number of vertices located at u just before the segment expansion that destroys u, or equivalently, just after the segment expansion that creates u. To simplify analysis, we set $n(r)$ to be the number of vertices in P immediately after the initial node expansion. Let $C(u)$ denote the children of node u in T. Recall that u' denotes some maximum-weight child of node u. Finally, let N denote the set of all nodes in T, and let $N' = \{u' \mid u \in N\}$.

Expanding segment uv distributes the $n(u)$ vertices at u as follows:

- All $\sigma(u)$ spurs at u vanish.
- In any subpath of the form $[a, u, v, \dots, u, a]$, where the intermediate vertices alternate between u and v, the first and last occurrences of u merge into a single spur at the child $[ua]$.
- Every other vertex at u moves to one of the children of u.

It follows that for any node u, we have

$$n(u) \ = \sigma(u) + \sum_{x \in C(u)} (n(x) + \sigma(x)),$$

which implies

$$n(u) - n(u') = \sum_{x \in C(u) \setminus \{u'\}} n(x) + \sum_{x \in C(u) \cup \{u\}} \sigma(x)$$

and therefore

$$\begin{aligned}\sum_{u \in N} (n(u) - n(u')) &\leq \sum_{x \in N \setminus N'} n(x) + 2 \sum_{x \in N} \sigma(x) \\ &= \sum_{x \in N \setminus N'} n(x) + O(n).\end{aligned}$$

The following standard heavy-path decomposition argument [28, 47] implies that $\sum_{x \in N \setminus N'} n(x) = O(n \log n)$. If we remove the vertices in N' from T by contracting each node in N' to its parent, the resulting tree has height $O(\log n)$, and the total weight of the nodes at each level is $O(n)$. We conclude that the total time spent expanding segments is $O(n \log n)$, which completes the proof of Theorem 7.1.

8 Generalizations and Open Problems

Finally, we conclude by describing some extensions of our results and a few interesting open problems.

8.1 Polygonal Chains

A straightforward generalization of our algorithm can determine whether a given *polygonal chain* is weakly simple in $O(n^2 \log n)$ time, by checking a polygon that traverses the chain twice.

Lemma 8.1. *Let $P = [p_0, p_1, \ldots, p_{n-1}, p_n]$ be an arbitrary polygonal chain. P is weakly simple if and only if the polygon $\hat{P} = (p_0, p_1, \ldots, p_{n-1}, p_n, p_{n-1}, \ldots, p_1)$ is weakly simple.*

Proof: Suppose P is weakly simple. Then for any $\delta > 0$, there is a simple curve Q with $d_{\mathcal{F}}(P, Q) < \delta$. In fact, we can assume Q is a simple polygonal chain, as Ribó Mor's results [40] extend to polygonal chains. Fix a positive real number $\varepsilon \ll \delta$ such that the intersection of Q with any closed disk of radius ε centered on a point of Q is connected; it suffices for ε be less than the *minimum local feature size* of Q [41]. Let $\hat{Q}$ denote the boundary of the ε-neighborhood of Q. Then $\hat{Q}$ is a simple closed curve where $d_{\mathcal{F}}(\hat{P}, \hat{Q}) < \delta + 2\varepsilon$, which implies that $\hat{P}$ is weakly simple.

On the other hand, suppose the polygon $\hat{P}$ is a weakly simple. Then for any $\delta > 0$ there is a simple polygon $\hat{Q}$ such that $d_V(\hat{P}, \hat{Q}) < \delta$. Let Q be either subpath of $\hat{Q}$ between the vertex corresponding to p_0 to the vertex corresponding to p_n. Then Q is a simple polygonal chain with $d_V(P, Q) < \delta$, which implies that P is weakly simple. □

8.2 Graph Drawings

There is a natural generalization of weak simplicity to arbitrary graph drawings. Any graph can be regarded as a topological space, specifically, a branched 1-manifold. A ***planar drawing*** of a graph H is just a continuous map from H to the plane; a drawing is ***simple*** or an ***embedding*** if it is injective. The Fréchet distance between two planar drawings P and Q of the same graph H is naturally defined as

$$d_{\mathcal{F}}(P, Q) = \inf_{\phi : H \to H} \max_{x \in H} d(P(\phi(x)), Q(x))$$

where the infimum is taken over all automorphisms of H (homeomorphisms from H to itself). We can define a planar drawing P to be ***weakly simple*** (or a ***weak embedding***) if, for any $\varepsilon > 0$, there is a planar embedding Q of H with $d_{\mathcal{F}}(P, Q) < \varepsilon$. This definition is consistent with our existing definitions of weakly simple closed curves in the plane. It is a natural open question whether one can decide in polynomial time whether a given straight-line drawing of a planar graph is a weak embedding; Cortese *et al.* [17] observe that it is not sufficient to check whether every cycle in the drawing is weakly simple.

However, a simple generalization of our algorithm actually solves this problem when the graph H being drawn is a disjoint union of cycles. In fact, the algorithm is unchanged except for the termination condition; when the main loop terminates, the image graph is the union of disjoint cycles and single segments, and P is weakly simple if and only if each component of P either traverses some cycle component of the image graph exactly once or maps to an isolated segment. Moreover, using the doubling trick described above in Section 8.1, our algorithm can also be applied to disjoint unions of cycles *and paths*. Details will appear in the full version of the paper.

Any algorithm to determine whether a graph drawing is a weak embedding must handle the special case where the image of the drawing is a simple path. This special case is equivalent to the *strip planarity* problem recently introduced by Angelini *et al.* [3] as a variant of clustered planarity [17, 23] and level planarity [34]. The strip planarity problem is open even when the graph H is a tree.

8.3 Surface Graphs

Our algorithm can also be generalized to surfaces of higher genus. A closed curve P in an arbitrary surface Σ is weakly simple if for any $\varepsilon > 0$ there is a simple (injective) closed curve Q in the same surface, such that $d_{\mathcal{F}}(P, Q) < \varepsilon$, where Fréchet distance is defined with respect to an arbitrary metric on Σ.

Theorem 8.2. *Given a closed walk P of length n in an arbitrary surface-embedded graph, we can determine whether P is weakly simple in $O(n \log n)$ time.*

Our algorithms for detecting weakly simple polygons use the geometry of the plane only in the preprocessing phase, where we apply a sweep-line algorithm to remove forks and to construct the image graph. Cortese *et al.* already describe topological versions of our expansion operations, which use topological disks instead of ellipses, as well as their proofs of correctness [17]. The only minor subtlety is the termination condition when the underlying surface is non-orientable. For any integer $k > 0$, let γ^k be the cycle that wraps around some simple cycle γ exactly k times. Then γ^k is weakly simple if and only if $k = 1$, or $k = 2$ and γ is orientation-reversing [14, 39, 51]. Again, details will appear in the full version of the paper.

8.4 Uniqueness

Another interesting open problem, suggested by Günter Rote, concerns the uniqueness of the simplifying perturbation. For any weakly simple polygon P without spurs, the perturbation required to make P simple appear to be essentially unique, in the following sense. Fix a sufficiently small parameter $\varepsilon > 0$. If Q and Q' are simple polygons within Fréchet distance ε of P, then Q can be morphed into Q' through a continuous sequence of simple closed curves within Fréchet distance ε of P. However, weakly simple polygons with spurs can be simplified by exponentially many essentially distinct perturbations; consider polygons of the form $(u, v, u, v, \ldots, u, v)$ for any points u and v. How quickly can we determine whether a given polygon is *uniquely* weakly simple? Can we efficiently count the number of essentially different simplifications?

8.5 Further Improvements

Finally, perhaps the most immediate open question is how to improve the $O(n^2 \log n)$ running time of our algorithm for arbitrary polygons with both spurs and forks. A direct generalization of bar expansion seems unlikely; Lemma 5.1 does not generalize to polygons with spurs. Nevertheless, we conjecture that the quadratic blowup from subdividing edges at forks can be avoided, and that the running time can be improved to $O(n \log n)$.

Acknowledgments. We thank to Sergio Cabello for pointing us toward the paper by Cortese *et al.* [17]. We are also grateful to Günter Rote and the anonymous SODA reviewers for extraordinarily detailed and helpful comments on earlier versions of the paper.

References

[1] Manuel Abellanas, Alfredo García, Ferran Hurtado, Javier Tejel, and Jorge Urrutia. Augmenting the connectivity of geometric graphs. *Comput. Geom. Theory Appl.* 40(3):220–230, 2008.

[2] Nancy M. Amato, Michael T. Goodrich, and Edgar Ramos. A randomized algorithm for triangulating a simple polygon in linear time. *Discrete Comput. Geom.* 26:245–265, 2001.

[3] Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, and Fabrizio Frati. Strip planarity testing. *Proc. 21st International Symposium on Graph Drawing*, 37–48, 2013. arXiv:1309.0683.

[4] Jon Louis Bentley and Thomas A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.* C-28(9):643–647, 1979.

[5] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*, 3rd edition. Springer-Verlag, 2008.

[6] Joan S. Birman and Caroline Series. Geodesics with bounded intersection number on surfaces are sparsely distributed. *Topology* 24(2):217–225, 1985.

[7] Prosenjit Bose, Jean-Lou De Carufel, Stephane Durocher, and Perouz Taslakian. Competitive online routing on Delaunay graphs. *Proc. 14th Scand. Workshop Algorithm Theory*, 98–104, 2014. Lecture Notes Comput. Sci. 8503, Springer.

[8] Prosenjit Bose and Pat Morin. Competitive online routing in geometric graphs. *Theoretical Computer Science* 324(2–3):273–288, 2004.

[9] John M. Boyer and Wendy J. Myrvold. On the cutting edge: Simplified $O(n)$ planarity by edge addition. *J. Graph Algorithms Appl.* 8(3):241–273, 2004.

[10] David Dylan Bremner. Point visibility graphs and restricted-orientation polygon covering. Master's thesis, Simon Fraser University, 1993.

[11] Erin W. Chambers, Éric Colin de Verdière, Jeff Erickson, Francis Lazarus, and Kim Whittlesey. Splitting (complicated) surfaces is hard. *Comput. Geom. Theory Appl.* 41(1–2):94–110, 2008.

[12] Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Minimum cuts and shortest homologous cycles. *Proc. 25th Ann. Symp. Comput. Geom.*, 377–385, 2009.

[13] Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.* 6(5):485–524, 1991.

[14] David R. J. Chillingworth. Winding numbers on surfaces, II. *Math. Ann.* 199:131–152, 1972.

[15] Kenneth L. Clarkson, Robert E. Tarjan, and Christopher J. Van Wyk. A fast Las Vegas algorithm for triangulating a simple polygon. *Discrete Comput. Geom.* 4:423–432, 1989.

[16] Robert Connelly, Erik D. Demaine, and Günter Rote. Infinitesimally locked self-touching linkages with applications to locked trees. *Physical Knots: Knotting, Linking, and Folding of Geometric Objects in* $\mathbb{R}^3$, 287–311, 2002. American Mathematical Society.

[17] Pier Francesco Cortese, Giuseppe Di Battista, Maurizio Patrignani, and Maurizio Pizzonia. On embedding a cycle in a plane graph. *Discrete Mathematics* 309(7):1856–1869, 2009.

[18] Mirela Damian, Robin Flatland, Joseph O'Rourke, and Suneeta Ramaswami. Connecting polygonizations via stretches and twangs. *Theory of Computing Systems* 47(3):674–695, 2010.

[19] Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge Univ. Press, 2007.

[20] Olivier Devillers. Randomization yields simple $O(n \log^* n)$ algorithms for difficult $\Omega(n)$ problems. *Int. J. Comput. Geom. Appl.* 2(1):621–635, 1992.

[21] Adrian Dumitrescu and Csaba D. Tóth. Light orthogonal networks with constant geometric dilation. *Journal of Discrete Algorithms* 7(1):112–129, 2009.

[22] Jeff Erickson and Amir Nayyeri. Shortest noncrossing walks in the plane. *Proc. 22nd Ann. ACM-SIAM Symp. Discrete Algorithms*, 297–308, 2011.

[23] Qing-Wen Feng, Robert F. Cohen, and Peter Eades. Planarity for clustered graphs. *Proc. 3rd Europ. Symp. Algorithms*, 1995. Lecture Notes Comput. Sci. 979, Springer.

[24] Qingwen Feng. Recognizing compound planarity of graphs. *Proc. 7th Australasian Workshop Combin. Algorithms*, 101–107, 1996. Technical Report 508, Basser Dept. Comput. Sci., Univ. Sydney. ⟨http://www.it.usyd.edu.au/research/tr/tr508.pdf⟩.

[25] Hubert de Fraysseix and Patrice Ossona de Mendez. Trémaux trees and planarity. *Europ. J.Combin.* 33(3):279–293, 2012.

[26] Branko Grünbaum. Polygons: Meister was right and Poinsot was wrong but prevailed. *Beitr. Algebra Geom.* 53(1):57–71, 2012.

[27] Leonidas J. Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graphics* 4(2):75–123, 1985.

[28] Dov Harel and Robert Endre Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM J. Comput.* 13(2):338–355, 1984.

[29] Michael Hoffmann, Bettina Speckmann, and Csaba D. Tóth. Pointed binary encompassing trees. *Proc. 9th Scand. Workshop Algorithm Theory*, 442–454, 2004. Lecture Notes in Computer Science 3111, Springer. Preliminary version of [30].

[30] Michael Hoffmann, Bettina Speckmann, and Csaba D. Tóth. Pointed binary encompassing trees: Simple and optimal. *Comput. Geom. Theory Appl.* 43(1):35–41, 2010. Full version of [29].

[31] Michael Hoffmann and Csaba D. Tóth. Pointed and colored binary encompassing trees. *Proc. 21st Ann. Symp. Comput. Geom.*, 81–90, 2005.

[32] John Hopcroft and Robert E. Tarjan. Efficient planarity testing. *J. Assoc. Comput. Mach.* 21(4):549–569, 1974.

[33] Mashhood Ishaque, Diane L. Souvaine, and Csaba D. Tóth. Disjoint compatible geometric matchings. *Discrete and Computational Geometry* 49(1):89–131, 2013.

[34] Michael Jünger, Sebastian Leipert, and Petra Mutzel. Level planarity testing in linear time. *Proc. 6th Int. Symp. Graw Drawing*, 224–237, 1998. Lecture Notes Comput. Sci. 1547, Springer.

[35] Yoshiyuki Kusakari, Hitoshi Suzuki, and Takao Nishizeki. A shortest pair of paths on the plane with obstacles and crossing areas. *Int. J. Comput. Geom. Appl.* 9(2):151–170, 1999.

[36] Kurt Mehlhorn and Stefan Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge Univ. Press, 1999.

[37] Albrecht Ludwig Friedrich Meister. Generalia de genesi figurarum planarum, et inde pendentibus earum affectionibus. *Novi Commentarii Soc. Reg. Scient. Gott.* 1:144–180 + 9 plates, 1769/1770. Presented January 6, 1770.

[38] Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces*. Johns Hopkins Univ. Press, 2001.

[39] Bruce L. Reinhart. Algorithms for Jordan curves on compact surfaces. *Ann. Math.* 75:271–283, 1962.

[40] Ares Ribó Mor. *Realization and Counting Problems for Planar Structures: Trees and Linkages, Polytopes and Polyominoes*. Ph.D. thesis, Freie Universität Berlin, 2006.

[41] Jim Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. *J. Algorithms* 18(3):548–585, 1995.

[42] Ignaz Rutter and Alexander Wolff. Augmenting the connectivity of planar and geometric graphs. *J. Graph Algorithms Appl.* 16(2):599–628, 2012.

[43] Raimund Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom. Theory Appl.* 1(1):51–64, 1991.

[44] Michael I. Shamos and Dan Hoey. Geometric intersection problems. *Proc. 17th Annu. IEEE Sympos. Found. Comput. Sci.*, 208–215, 1976.

[45] Thomas Shermer and Godfried Toussaint. Characterizations of star-shaped polygons. Tech. Rep. 92–11, School of Computing Science, Simon Fraser University, December 1992. ⟨ftp://fas.sfu.ca/pub/cs/TR/1992/CMPT92-11.ps.gz⟩.

[46] Wei-Kuan Shih and Wen-Lian Hsu. A new planarity test. *Theoret. Comput. Sci.* 223(1–2):179–191, 1999.

[47] Daniel D. Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.* 26(3):362–391, 1983.

[48] Godfried T. Toussaint. Computing geodesic properties inside a simple polygon. *Revue d'Intelligence Artificielle* 3(2):9–42, 1989.

[49] Godfried T. Toussaint. On separating two simple polygons by a single translation. *Discrete Comput. Geom.* 4(1):265–278, 1989.

[50] Chung-Do Yang, Der-Tsai Lee, and Chak-Kuen Wong. The smallest pair of noncrossing paths in a rectilinear polygon. *IEEE Trans. Comput.* 46(8):930–941, 1997.

[51] Heiner Zieschang. Algorithmen für einfache Kurven auf Flächen. *Math. Scand.* 17:17–40, 1965.