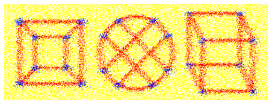

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 15, no. 1, pp. 7–32 (2011)

Planar Drawings of Higher-Genus Graphs

Christian A. Duncan[1] *Michael T. Goodrich*[2]
Stephen G. Kobourov[3]

[1]Dept. of Computer Science, Louisiana Tech Univ.
[2]Dept. of Computer Science, Univ. of California, Irvine
[3]Dept. of Computer Science, University of Arizona

Abstract

In this paper, we give polynomial-time algorithms that can take a graph G with a given combinatorial embedding on an orientable surface $\mathcal{S}$ of genus g and produce a planar drawing of G in $\mathbf{R}^2$, with a bounding face defined by a polygonal schema $\mathcal{P}$ for $\mathcal{S}$. Our drawings are planar, but they allow for multiple copies of vertices and edges on $\mathcal{P}$'s boundary, which is a common way of visualizing higher-genus graphs in the plane. However, unlike traditional approaches the copies of the vertices might not be in perfect alignment but we guarantee that their order along the boundary is still preserved. Our drawings can be defined with respect to either a canonical polygonal schema or a polygonal cutset schema, which provides an interesting tradeoff, since canonical schemas have fewer sides, and have a nice topological structure, but they can have many more repeated vertices and edges than general polygonal cutsets. As a side note, we show that it is NP-complete to determine whether a given graph embedded in a genus-g surface has a set of $2g$ fundamental cycles with vertex-disjoint interiors, which would be desirable from a graph-drawing perspective.

Submitted:	Reviewed:	Revised:	Accepted:
December 2009	August 2010	August 2010	August 2010
	Final: December 2010	Published: February 2011	
Article type: Regular paper		Communicated by: D. Eppstein and E. R. Gansner	

Research supported by NSF Career Grant CCF-0545743, NSF Grant CCR-0830403, and ONR MURI Award, number N00014-08-1-1015.

E-mail addresses: christian.duncan@acm.org (Christian A. Duncan) goodrich@ics.uci.edu (Michael T. Goodrich) kobourov@cs.arizona.edu (Stephen G. Kobourov)

1 Introduction

The *classic* way of drawing a graph $G = (V, E)$ in $\mathbf{R}^2$ involves associating each vertex v in V with a unique point (x_v, y_v) and associating with each edge $(v, w) \in E$ an open Jordan curve that has (x_v, y_v) and (x_w, y_w) as its endpoints. If the curves associated with the edges in a classic drawing of G intersect only at their endpoints, then (the embedding of) G is a *plane graph*. Graphs that admit plane graph representations are *planar graphs*, and there has been a voluminous amount of work on algorithms on classic drawings of planar graphs (e.g., see [15]). Most notably, planar graphs can be drawn with vertices assigned to integer coordinates in an $O(n) \times O(n)$ grid, which is often a desired type of classic drawing known as a *grid drawing*. Moreover, there are planar graph drawings that use only straight line segments for edges (e.g., see [3, 7, 9, 16]).

The beauty of plane graph drawings is that, by avoiding edge crossings, confusion and clutter in the drawing is minimized. Likewise, straight-line drawings further improve graph visualization by allowing the eye to easily follow connections between adjacent vertices. In addition, grid drawings enforce a natural separation between vertices, which further improves readability. Thus, a "gold standard" in classic drawings is to produce planar straight-line grid drawings and, when that is not easily done, to produce planar grid drawings with edges drawn as simple polygonal chains (e.g., see [4, 15]).

Unfortunately, not all graphs are planar. So drawing them in the classic way requires some compromise in the gold standard for plane drawings. In particular, any classic drawing of a non-planar graph must necessarily have edge crossings, and minimizing the number of crossings is NP-hard [8]. One point of hope for improved drawings of non-planar graphs is to draw them crossing-free on surfaces of higher genus, such as toruses, double toruses, or, in general, a surface topologically equivalent to a sphere with g handles, that is, a *genus-g* surface. Such drawings are called *cellular* embeddings or 2-*cell* embeddings, since they partition the genus-g surface into a collection of cells that are topologically equivalent to disks. As in classic drawings of planar graphs, these cells are called *faces*, and it is easy to see that such a drawing would avoid edge crossings.

In a fashion analogous to the case with planar graphs, cellular embeddings of graphs in a genus-g surface can be characterized combinatorially. In particular, it is enough if we just have a rotational order of the edges incident on each vertex in a graph G to determine a combinatorial embedding of G on a surface (which has that ordering of associated curves listed counterclockwise around each vertex). Such a set of orderings is called a *rotation system* and, since it gives us a combinatorial description of the set of faces, F, in the embedding, it gives us a way to determine the genus of the (orientable) surface that G is embedded into by using the *Euler characteristic*, $|V| - |E| + |F| = 2 - 2g$, which also implies that $|E|$ is $O(|V| + g)$ [14].

Unfortunately, given a graph G, it is NP-hard to find the smallest g such that G has a combinatorial cellular embedding on a genus-g surface [17]. This challenge need not be a deal-breaker in practice, however, for there are heuristic

algorithms for producing such combinatorial embeddings (that is, consistent rotation systems) [2]. Moreover, higher-genus graphs often come together with combinatorial embeddings in practice, as in many computer graphics and mesh generation applications.

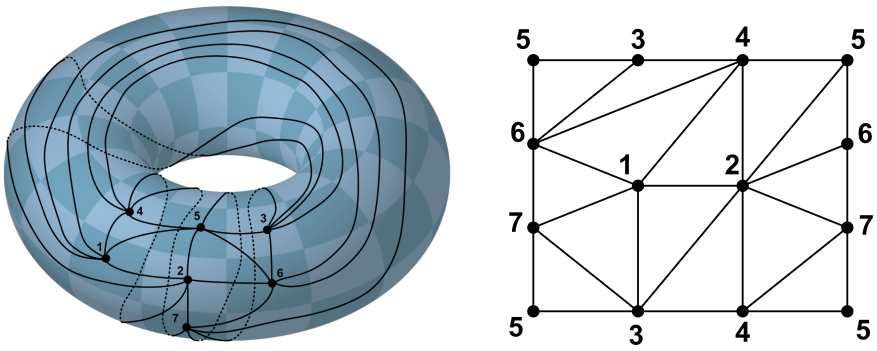
In this paper, we assume that we are given a combinatorial embedding of a graph G on an orientable genus-g surface, $\mathcal{S}$, and are asked to produce a geometric drawing of G that respects the given rotation system. Motivated by the gold standard for planar graph drawing and by the fact that computer screens and physical printouts are still primarily two-dimensional display surfaces, the approach we take is to draw G in the plane rather than on some embedding of $\mathcal{S}$ in $\mathbf{R}^3$.

Making this choice of drawing paradigm, of course, requires that we "cut up" the genus-g surface, $\mathcal{S}$, and "unfold" it so that the resulting sheet is topologically equivalent to a disk. Fortunately, there are several well-studied ways of doing this, at least in a topological sense. The traditional method for performing such a cutting is with a polygonal schema formed from fundamental cycles. A *fundamental cycle* on $\mathcal{S}$ is a continuous closed curve on $\mathcal{S}$ that cannot be retracted continuously to a point. If cutting $\mathcal{S}$ along a fundamental cycle results in a topological disk, the resulting boundary composed of the cut edges is called a *polygonal schema*, $\mathcal{P}$. Essentially, in the plane, the cycle can be represented as a polygon of at least $4g$ sides such that identifying edges with each other (gluing the cut edges back together) results in the original surface. If the schema is formed by $2g$ fundamental cycles all containing a common point, p, it is referred to as a *canonical polygonal schema*, for the corresponding polygon can be represented with exactly $4g$ sides each corner of which is the point p. Observe that the union of the $2g$ cycles is itself a fundamental cycle, and hence the cycles still form a valid polygonal schema. Moreover, these $2g$ fundamental cycles can be paired up into complementary sets of cycles, (a_i, b_i), one for each handle, so that if we orient the sides of $\mathcal{P}$, then a counterclockwise ordering of the sides of $\mathcal{P}$ can be listed as

$$a_1 b_1 a_1^{-1} b_1^{-1} a_2 b_2 a_2^{-1} b_2^{-1} \ \ldots \ a_g b_g a_g^{-1} b_g^{-1},$$

where a_i^{-1} (b_i^{-1}) is a reversely-oriented copy of a_i (b_i), so that these two sides of $\mathcal{P}$ are matched in orientation on $\mathcal{S}$. That is, they correspond to the two sides of a cycle that we cut on $\mathcal{S}$ to form a polygonal schema $\mathcal{P}$. Thus, the canonical polygonal schema for a genus-g surface $\mathcal{S}$ has $4g$ sides that are pairwise identified; see Fig. 1.

Because we are interested in drawing the graph G and not just the topology of $\mathcal{S}$, it would be preferable if the fundamental cycles are also cycles in G in the graph-theoretical sense. It would be ideal if these cycles form a canonical polygonal schema with no repeated vertices other than the common one. This is not always possible [11] and furthermore, as we show in Section 2.2, the problem of finding a set of $2g$ fundamental cycles with vertex-disjoint interiors in a combinatorially embedded genus-g graph is NP-complete. There are two natural choices, both of which we explore in this paper:



(a) (b)

Figure 1: Drawings of K_7 on the torus. (a) A crossing-free drawing of K_7 using the classic approach; (b) A polygonal-schema drawing of K_7, with fundamental cycles $a_1 = (5, 3, 4, 5)$ and $b_1 = (5, 6, 7, 5)$.

- Draw G in a polygon P corresponding to a canonical polygonal schema, $\mathcal{P}$, possibly with repeated vertices and edges on its boundary.
- Draw G in a polygon P corresponding to a polygonal schema, $\mathcal{P}$, that is not canonical.

In either case, the edges and vertices on the boundary of P are repeated (since we "cut" $\mathcal{S}$ along these edges and vertices). Thus, we need labels in our drawing of G to identify the correspondences. Such planar drawings of G inside a polygonal schema $\mathcal{P}$ are called *polygonal-schema* drawings of G. There are three natural aesthetic criteria such drawings should satisfy:

1. *Straight-line edges:* All the edges in a polygonal-schema drawing should be rendered as polygonal chains, or straight-line edges, when possible.
2. *Straight frame:* Each side of the polygonal schema, which in a canonical polygonal schema corresponds to a fundamental cycle that was cut along its boundary, should be rendered as a straight line segment, with the vertices and edges of the corresponding fundamental cycle placed along this segment, and such that the resulting polygon is convex. We refer to such a polygonal-schema drawing as having a *straight frame*.
3. *Polynomial area:* Drawings should have polynomial area when they are normalized to an integer grid.

It is also possible to avoid repeated vertices and instead use a classic graph drawing paradigm, by transforming the fundamental polygon rendering using polygonal-chain edges that run through "overpasses" and "underpasses" as in road networks, so as to illustrate the topological structure of G; see Fig. 2.

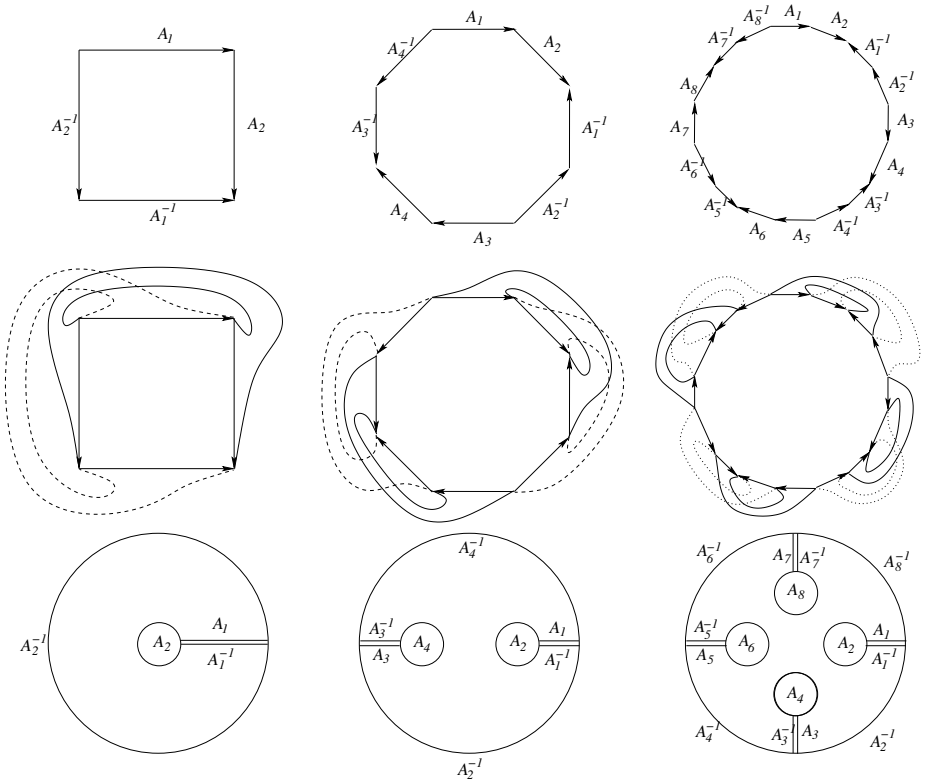


Figure 2: First row: Canonical polygonal schemas for graphs of genus one, two and four. Second row: Unrolling the high genus graphs with the aid of the overpasses and underpasses. Third row: Folding high genus graphs by compressing all the overpasses.

Our Contributions. We provide several methods for producing planar polygonal-schema drawings of higher-genus graphs. In particular, we provide four algorithms, one for toroidal ($g = 1$) graphs and three for non-toroidal ($g > 1$) graphs. Our algorithm for toroidal graphs simultaneously achieves the three aesthetic criteria for polygonal schema drawings: it uses straight-line edges, a straight frame, and polynomial area. The three algorithms for non-toroidal graphs, *Peel-and-Bend*, *Peel-and-Stretch*, and *Peel-and-Place*, provide various tradeoffs between these three desired criteria, which are identified in Table 1. Thus, we can achieve any two of the three desired aesthetic criteria for non-toroidal graphs. A recent result by Chambers *et al.* [1] has extended our work to achieve all three simultaneously for non-toroidal graphs.

Our algorithms also can be differentiated on whether we use a canonical polygonal schema or not. Having the ability to use either a canonical polygonal schema or a general cutset schema allows for more flexibility for visualizing graphs with combinatorial embeddings in genus-g surfaces. Specifically, canoni-

Algorithm	straight-line edges	straight frame	poly. area
Peel-and-Bend		•	•
Peel-and-Stretch	•	•	
Peel-and-Place	•		•

Table 1: Aesthetic criteria achieved by each of our algorithhms for non-toroidal graphs.

cal polygonal schemas tend to have fewer sides than general polygonal schemas, which tend to have fewer boundary vertices than canonical polygonal schemas. Moreover, as we show in Section 2.2, optimizing both of these competing criteria is NP-complete.

2 Finding Polygonal Schemas

Suppose we are given a graph G together with its cellular embedding on an orientable genus-g surface, $\mathcal{S}$. An important first step in all of our algorithms involves our finding a polygonal schema, $\mathcal{P}$, for G, that is, a set of cycles in G such that cutting $\mathcal{S}$ along these cycles results in a topological disk. We refer to this as the *Peel* step, since it involves cutting the surface $\mathcal{S}$ until it becomes topologically equivalent to a disk. Since these cycles form the sides of the fundamental polygon we will be using as the outer face in our drawing of G, it is desirable that these cycles be as "nice" as possible with respect to drawing aesthetics.

2.1 Trade-offs for Finding Polygonal Schemas

Unfortunately, some desirable properties are not effectively achievable. As Lazarus *et al.* [11] show, it is not always possible to have a canonical polygonal schema $\mathcal{P}$ such that each fundamental cycle in $\mathcal{P}$ has a distinct set of vertices in its interior. The *interior* of a fundamental cycle is the set of vertices distinct from the common vertex shared with its complementary fundamental cycle—with this vertex forming a corner of a canonical polygonal schema. In addition, we show in Section 2.2 that finding such a vertex-disjoint set of fundamental cycles is NP-complete. So, from a practical point of view, we have two choices with respect to methods for finding polygonal schemas.

Finding a Canonical Polygonal Schema. As mentioned previously, a canonical polygon schema of a graph G 2-cell embedded on a surface of genus g consists of $4g$ sides, which correspond to $2g$ fundamental cycles all containing a common vertex. Lazarus *et al.* [11] show that one can find such a schema for G in $O(gn)$ time and with total size $O(gn)$, and they show that this bound is within a constant factor of optimal in the worst case, where n is the total combinatorial complexity of G (vertices, edges, and faces), which is $O(|V| + g)$.

Minimizing the Number of Boundary Vertices in a Polygonal Schema. Another optimization would be to minimize the number of vertices in the boundary of a polygonal schema. Erikson and Har-Peled [6] show that this problem is NP-hard, but they provide an $O(\log^2 g)$-approximation algorithm that runs in $O(g^2 n \log n)$ time, and they give an exact algorithm that runs in $O(n^{O(g)})$ time.

In our *Peel* step, we assume that we use one of these two optimization criteria to find a polygonal schema, which either optimizes its number of sides to be $4g$, as in the canonical case, or optimizes the number of vertices on its boundary, which will be $O(gn)$ in the worst case either way. Nevertheless, for the sake of concreteness, we often describe our algorithms assuming we are given a canonical polygonal schema. It is straight-forward to adapt these algorithms for non-canonical schemas. We elaborate on some of these details throughout.

2.2 Finding Simple Fundamental Cycles is NP-Complete

We now address the difficulty in finding fundamental cycles with vertex-disjoint interiors.

Theorem 1 *Given a graph G combinatorially embedded on a genus-g surface, it is NP-complete to find a set of $2g$ fundamental cycles with vertex-disjoint interiors.*

Proof: It is easy to see that this problem is NP—just guess a set of $2g$ cycles and test that they are fundamental cycles with vertex-disjoint interiors.

To complete the proof, we observe that there is a simple reduction from the following NP-complete problem [12]:

- Disjoint Paths for Planar Graphs with Maximum Degree 3 (DPP3): Given a planar graph G with maximum degree 3 and a set S of disjoint edges in G, determine if there is a set of vertex-disjoint cycles in G such that each cycle contains exactly one edge of S and each edge of S is in exactly one cycle.

Suppose, then, that we are given an instance of the DPP3 problem. We begin by removing all the edges in S from G, giving the subgraph G', and checking, for each edge $e = (v, w)$ in S, if w is reachable from v. If any such test fails, then the answer to the DPP3 problem is "no." So, let us suppose each such endvertex is reachable from its partner. Then, for each $e = (v, w)$ in S, we connect v and w with a subgraph, G_e, which is a copy of K_5 minus one edge, oriented as shown in Fig. 3. Let $\hat{G}$ denote G' with the subgraph G_e replaced for each edge e in S, and note that $\hat{G}$ is embedded on a surface of genus $|S|$.

Suppose that for a given G and S, the answer to the DPP3 problem is "yes." Then in $\hat{G}$ we can form a set of $2|S|$ fundamental cycles with vertex-disjoint interiors, by taking, for each edge $e = (v, w)$ in S, the path from v to w in G' plus a path from w to v in G_e, and we can form its complementary fundamental cycle, say, by taking the cycle formed by the vertices in G_e adjacent to v.

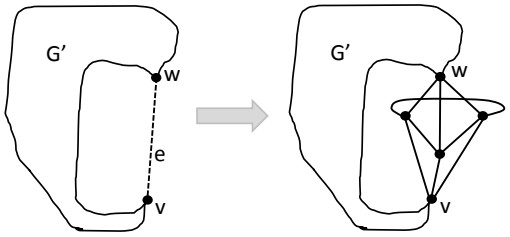


Figure 3: Replacing each edge in S with G_e, which is K_5 minus one edge.

Suppose, instead, that we can form a set of $2|S|$ vertex-disjoint fundamental cycles in $\hat{G}$. Note that, since w is reachable from v in G', for each $e = (v, w)$ in S, each G_e requires a handle in the embedding of $\hat{G}$ on a surface of genus $|S|$. That is, G' is essentially providing the missing edge in K_5 and the edges of G_e are oriented to require a handle here. Moreover, by the construction, one fundamental cycle must include only vertices in G_e, since G_e forms a biconnected component in $\hat{G}$. In addition, since G_e does not itself contain two complementary fundamental cycles, the complementary fundamental cycle to the one in G_e must be formed by a path from v to w in G'. Therefore, if all the fundamental cycles in $\hat{G}$ have vertex-disjoint interiors, then the paths in G' connecting the ends of each edge in S must be vertex disjoint. Thus, replacing each subgraph G_e with the associated edge e in S gives us a set of $|S|$ vertex-disjoint cycles in G containing each edge in S. □

2.3 Constructing Chord-Free Polygonal Schemas

In all of our algorithms the first step, *Peel*, constructs a polygonal schema of the input graph G. In fact, we need a polygonal schema, $\mathcal{P}$, in which there is no chord connecting two vertices on the same side of $\mathcal{P}$. Here we show how to transform any polygonal schema into a chord-free polygonal schema.

In the *Peel* step, we cut the graph G along a canonical set of $2g$ fundamental cycles getting two copies of each cycle in G^*, the resulting planar graph. For each of the two pairs of every fundamental cycle there may be chords. If the chord connects two vertices that are in different copies of the cycle in G^* then this is a chord that *can* be drawn with a straight-line edge and hence does not create a problem. However, if the chord connects two vertices in the same copy of the cycle in G^*, then we will not be able to place all the vertices of that cycle on a straight-line segment; see Fig. 4(a). We show next that a new chord-free polygonal schema can be efficiently determined from the original schema.

Theorem 2 *Given a graph G combinatorially embedded on a genus-g surface and a canonical polygonal schema $\mathcal{P}$ on G with a common vertex p, a* chord-free polygonal schema $\mathcal{P}^*$ *can be found in $O(gn)$ time.*

Proof: We first use the polygonal schema to cut the embedding of G into a

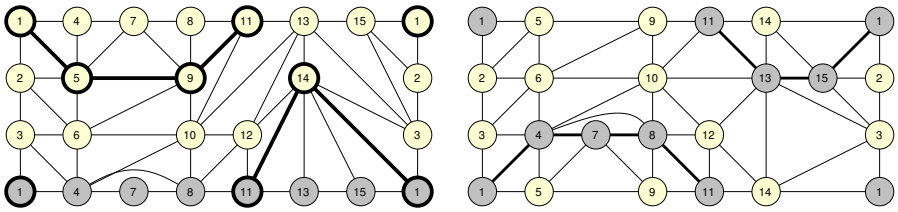


Figure 4: (a) A graph embedded on the torus that has been cut into a topological disk using the cycles $1, 2, 3$ and $1, 4, 7, 8, 11, 13, 15$ with chord $(4, 8)$. The grey nodes correspond to the identical vertices above. The highlighted path represents a shortest path between the two copies of the common vertex 1. (b) The topological disk *after* cutting along this new fundamental cycle. The grey nodes show the old fundamental cycle.

topological disk; see Fig. 4(a). Notice this cutting causes certain vertices to be split into multiple vertices. For each fundamental cycle $c_i \in \mathcal{P}$ starting with $i = 1$, we temporarily stitch the disk graph back together along this cycle forming a topological cylinder. The outer edges (left and right) of the cylinder along this stitch have (at least) two copies of the common vertex p, say p_1 and p_2. We perform a shortest path search from p_1 to p_2. This path becomes our new fundamental cycle c_i^*, (since p_1 and p_2 are the same vertex in G). Observe that this cycle must be chord-free or else the path chosen was not the shortest path; see Fig. 4(b). We then cut the cylinder along c_i^* and proceed to c_{i+1}. The resulting set, $\mathcal{P}^* = \{c_1^*, c_2^*, \ldots, c_{2g}^*\}$, is therefore a collection of chord-free fundamental cycles all sharing the common vertex p. □

It should be noted that, although each cycle c_i^* is at the time of its creation a shortest path from the two copies of p, these cycles are *not* the shortest fundamental cycles possible. For example, a change in the cycle of c_{i+1} could introduce a shorter possible path for c_i^* but not additional chords. These cycles are in a sense minimal but they are not the optimal (minimum) set; recall that computing the optimal set of fundamental cycles is NP-hard [6].

If the polygonal schema is not canonical, we no longer require that each side be a complete fundamental cycle with a common shared point, but we still want to maintain the overall structure of the polygon by retaining the vertices at the corners of the polygon. Fortunately, the same technique can be used to ensure each side of the polygon is chord-free: create a topological disk by cutting along the entire cycle, stitch each side together one by one creating topological cylinders, perform a shortest path search from one side at p_1 to the other at p_2, and cut along that path to get a disk again. The resulting polygonal schema will have the same vertex corners as the original but each side will be chord-free.

3 Straight Frame and Polynomial Area

In this section, we describe our algorithms that construct a drawing of G in a straight frame using polynomial area. Here we are given an embedded genus-g graph $G = (V, E)$ along with a chord-free polygonal schema, $\mathcal{P}$, for G from the *Peel* step. For the drawing, we rely on a modified version of the algorithm of de Fraysseix, Pach and Pollack [7], of which a brief review is given in Section 3.1. Sections 3.2 and 3.3 describe the details for $g = 1$ and for $g > 1$, respectively. In the latter case we introduce up to $O(k)$ edges with single bends where k is the number of vertices on the fundamental cycles. Thus, we refer to the algorithm for non-toroidal graphs as the *Peel-and-Bend* algorithm.

3.1 Canonical Labeling

We start with a brief overview of the algorithm of de Fraysseix, Pach and Pollack for drawing planar graphs with straight lines on an $O(n) \times O(n)$ grid [7]. The dPP algorithm inserts vertices of a fully triangulated graph incrementally according to an ordering created by a *canonical labeling* defined below, which assigns to each of the n vertices a unique label from the set $\{1, 2, \ldots, n\}$. We refer to the subgraph of G induced by the first i inserted vertices as G_i. The algorithm maintains a key invariant that at each iteration every edge on the boundary C_i of the external face of G_i is drawn with a slope either $+1$ or -1 except for the initial two vertices v_1 and v_2 that are connected by a horizontal line (at the bottom of the drawing). The face C_i, minus the edge from v_1 to v_2, is also drawn monotonically in the x-direction so that its shape is akin to a chain of mountain peaks. The initial graph G_3 is drawn with v_1, v_2 and v_3 at positions $(0, 0)$, $(2, 0)$ and $(1, 1)$ respectively. The vertices are ordered such that for any vertex v_i the set of its neighbors in G_{i-1} forms a consecutive chain on C_i. This condition and the shape of the drawing of C_i guarantees that each successive vertex v_i can "easily see" all of its neighbors in G_{i-1}, although some horizontal shifting of the vertices might be necessary to maintain the key invariant.

The canonical labeling is created essentially by reversing the process. For each graph G_i, starting with $G = G_{n-1}$, there exists at least one vertex v_i, not including v_1 and v_2, on C_i with exactly two neighbors on C_i, one v_ℓ immediately left and the other v_r immediately right of v_i. Since the graph is fully triangulated, the removal of v_i from G_i yields a new subgraph with an external face that is the same as C_i except that v_i has been replaced with a chain of its neighbors from v_ℓ to v_r.

Formally, we define the labeling as follows. Given a fully triangulated planar graph G embedded in the plane with exterior face v_1, v_2, v_n, a canonical labeling of the vertices $v_1, v_2, v_3, \ldots, v_{n-1}, v_n$ satisfies the following requirements for $n > i > 2$:

1. G_i, the subgraph of G induced by the first v_i vertices, is bi-connected, and the boundary C_i of its outer face contains the edge (v_1, v_2);

2. v_{i+1} lies in the outer face and its neighbors in G_i form a connected subpath of $C_i - (v_1, v_2)$ with length at least two.

The canonical ordering and the subsequent embedding can be done in linear time [3] and produces an embedding on an $O(n) \times O(n)$ grid [7]. Although the standard dPP algorithm does not directly solve all our needs, it can be modified so that we can achieve our stated goals.

3.2 Grid Embedding of Toroidal Graphs

For toroidal graphs we are able to achieve all three aesthetic criteria: straight-line edges, straight frame, and polynomial area.

Theorem 3 *Let G^* be an embedded planar graph and $\mathcal{P} = \{P_1, P_2, P_3, P_4\}$ in G^* be a collection of four paths such that each path $P_i = \{p_{i,1}, p_{i,2}, \ldots, p_{i,k_i}\}$ is chord-free, the last vertex of each path matches the first vertex of the next path, $(p_{i,k_i} = p_{i+1,1})$, including P_4 and P_1, and when treated as a single cycle, $\mathcal{P}$ forms the external face of G^*. We can in linear time draw G^* on an $O(n) \times O(n^2)$ grid with straight-line edges and no crossings in such a way that, for each path P_i on the external face, the vertices on that path form a straight line. Moreover, the external face forms a rectangle.*

We refer to $\mathcal{P}$ as a collection of paths, but they originated as fundamental cycles that were cut in a previous step. For our purposes, it is easier to think of each split vertex as a separate vertex and hence the cycles become paths after the cutting process. Reconnecting the split vertices, if desired, is a separate step using bridge edges (underpasses and overpasses). The remainder of this section deals with the proof of this theorem.

Proof: For simplicity, we assume that every face is a triangle, except for the outer face (extra edges can be added and later removed). The algorithm of de Fraysseix, Pach and Pollack (dPP) [7] does not directly solve our problem because of the additional requirement for the drawing of the external face. In this case, the additional requirement is that the graph must be drawn so that the external face forms a rectangle, with P_1 and P_3 as the top and bottom horizontal boundaries and P_2 and P_4 as the right and left boundaries.

As mentioned in Section 3.1, the dPP algorithm computes a canonical labeling of the vertices of the input graph and inserts them one at a time in that order while ensuring that when a new vertex is introduced it can "see" all of its already inserted neighbors. In our problem we need to modify the algorithm to ensure that all vertices belonging to the same fundamental cycle are placed along a straight-line segment. One technical difficulty lies in the proper placement of the top row of vertices. Due to the nature of the canonical order, we cannot force the top row of vertices to all be the last set of vertices inserted, unlike the bottom row which can be the first set inserted; see Fig. 5. Consequently, we propose an approach similar to that of Miura, Nakano, and Nishizeki [13]. First, we split the graph into two parts (not necessarily of equal size), perform

a modified embedding on both pieces, invert one of the two pieces, and stitch the two pieces together.

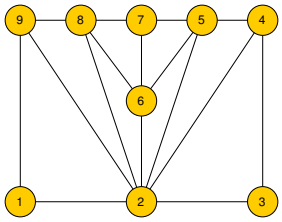


Figure 5: Placing vertices $9, 8, 7, 5, 4$ along the top row; labels correspond to the canonical ordering. The dPP algorithm would insert vertex 6 after 5 but before 7. As dPP places the next vertex in the canonical order *above* their neighboring predecessors, 6 would be placed above 5, thus preventing 5 from being in the top row.

Lemma 1 *Given an embedded plane graph G that is fully triangulated except for the external face and two edges e_ℓ and e_r on that external face, it is possible in linear time to partition $V(G)$ into two subsets V_1 and V_2 such that*

1. *the subgraphs of G induced by V_1 and V_2, called G_1 and G_2, are both connected subgraphs;*
2. *for edges $e_\ell = (u_\ell, v_\ell)$ and $e_r = (u_r, v_r)$, we have $u_\ell, u_r \in V_1$ and $v_\ell, v_r \in V_2$;*
3. *the union U of the set of internal faces in G that are not in G_1 or G_2 forms an outerplane graph with the property that the external face of U is a cycle with no repeated vertices.*

Proof: First, we compute the dual D of G, where each face in (the primal graph) G is a node in D and there is an arc between two nodes in D if their corresponding primal faces share an edge in common. We delete the node corresponding to the external face from D, along with its incident arcs. For clarity we shall refer to vertices and edges in the primal and nodes and arcs in the dual; see Fig. 6(a). We further augment the dual by adding an arc between two nodes in D if they also share a vertex in common. Call this augmented dual graph D^*. We refer to the arcs in D as *regular* arcs and to the arcs added to form D^* as *augmented* arcs.

Let the source node s be the node corresponding to the internal face containing the edge e_ℓ and the sink node t be the node corresponding to the internal face with edge e_r. We then perform a breadth-first shortest-path traversal from s to t on D^*; see Fig. 6(b). Let p^* be a shortest (augmented) path in D^* obtained by this search. We now create an (unaugmented) path p in D by expanding the augmented arcs added. That is, if there is an arc $(u, v) \in p^*$ such that u and v

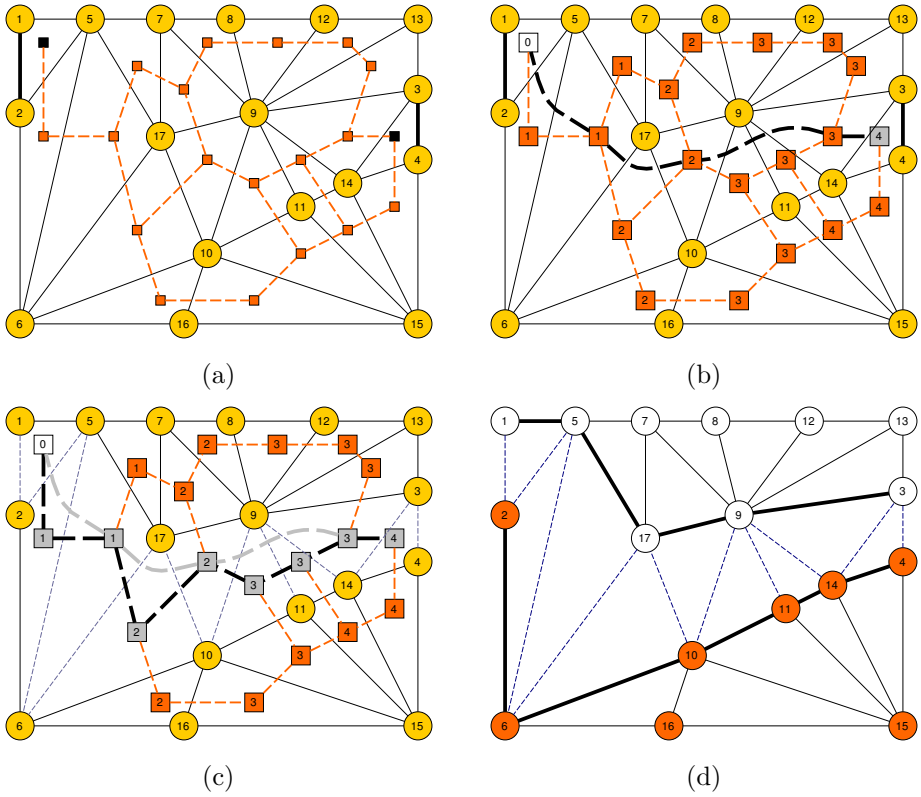


Figure 6: (a) A graph G and its dual D. Note that the augmented arcs for D^* are not shown. For example, there are arcs between all faces (nodes) incident to vertex 9. The dark edges/nodes represent the sink and source nodes. (b) Each dual node is labeled with its distance (in D^*) from the start node 0. Note the sink is at distance 4. A shortest path p^* is drawn with thick dark arcs. This path includes the augmented arcs of D^*. (c) The path p formed after expanding the augmented arcs. The edges from the primal that are cut by this path are shown faded. (d) The two sets V_1 (light vertices) and V_2 (darker vertices) formed by the removal of path p. The external face of U is defined by the thick edges along with the edges $(1, 2)$ and $(3, 4)$.

share a common vertex in G but *not* a common edge in G, i.e. they are part of a fan around the common vertex, we add back the regular arcs from u to v adjacent to this common vertex. The choice of going clockwise or counter-clockwise around the common vertex depends on the previous visited arc. That is, if the previous arc was a result of going around vertex a and we now wish to go around vertex b, the current (dual) node corresponds to the (primal) face abc for some shared vertex c and we go around b by crossing edge bc, rather than edge ba. For example, in Fig. 6(c), if at node (triangle) $5, 6, 17$ we went around vertex 17 clockwise rather than counterclockwise, we would have used node $9, 7, 17$ before reaching node $9, 17, 10$. At this point, going clockwise around vertex 9 would mean repeating the node $9, 7, 17$.

Although p is no longer the shortest possible path in D, it is still necessarily acyclic, and so no node in D is visited more than once. Additionally, no (primal) vertex is visited by more than one consecutive sequence of nodes in the path. That is, if two nodes u and v in p share a common (primal) vertex then all nodes between u and v in p also share a common vertex. To see why, assume not. Then in p^* there would have been two nodes u and v that shared a common vertex that visited at least one other node w between them. But since the distance between u and v in D^* is one, p^* could not be a shortest path.

Each arc in p corresponds to an edge in G. Since p is a path from one edge on the external face to another edge on the external face, the removal of these edges in G along with the two edges e_ℓ and e_r clearly separates the graph into two subgraphs G_1 and G_2. This proves the first condition. The second condition holds directly by construction, since we started with and removed edges e_ℓ and e_r; see Fig. 6(d).

We prove the final condition inductively by considering the nodes along the path p incrementally. First note that after the start node s, we have a single triangular face which clearly is outerplanar with a simple cycle for an external face. Therefore, let us assume that we have added the first $i-1$ nodes of p creating the outerplane graph U_{i-1} with the external face as a simple cycle. We now add the node u_i to create U_i. Since U_{i-1} is outerplanar, all vertices are represented on the external cycle, which is simple. The addition of node u_i replaces one edge on the external cycle, say (a_{i-1}, b_{i-1}), with two edges (a_{i-1}, a_i) and (a_i, b_{i-1}). Since no (primal) vertex on the path p is visited more than once non-consecutively, a_i must be a new vertex not already on the external face. In addition, notice that both a_{i-1} and b_{i-1} remain on the external face of U_i. Therefore, both U_i and its external face increase by exactly one new vertex a_i. Consequently, all vertices are still represented on the external face which implies that U_i is outerplanar. Since a_i is a new vertex, the external face remains a simple cycle.

All of the steps described above can be easily implemented in linear time. However, the augmented arcs could incur quadratic time and space costs (e.g., with a vertex of degree $O(n)$ in G). We can solve the problem in linear time and space if we instead find the path using the vertex-face incidence graph VF of G (see, [14]), where every vertex and face in G corresponds to a node in VF and two nodes a and b in VF are connected by an arc if and only if a corresponds to a

vertex that is on the boundary of the face b. Any path in VF is an alternating sequence of face and vertex nodes, and bypassing the vertex nodes in such a path yields an equivalent path in the augmented dual graph. Similarly, any path in the augmented dual graph can be mapped to an equivalent path in VF, by including a shared vertex between face nodes. Therefore, a shortest path in VF can be used to compute a valid shortest augmented path p^*. □

Fig. 6(d) illustrates the result of one such partition as well as an issue that can arise in the splitting of the planar graph G. In particular, there will be vertices on the external cycle of G that are also part of the union U. This means that either G_1 or G_2 might have a cut vertex. For vertices that are to be placed along the bottom row this is not a problem, but it can be problematic for the sides. We can avoid this issue by choosing a start and end edge from among a set of edges rather than just the two edges e_ℓ and e_r. The following extension of Lemma 1 addresses this issue.

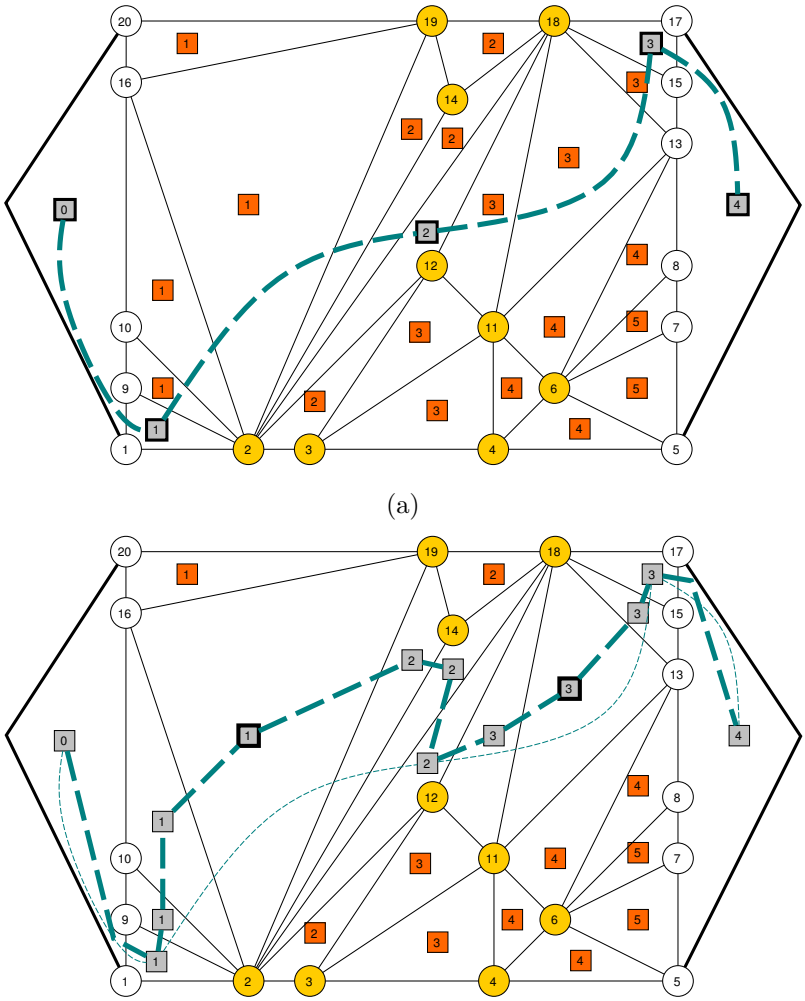
Lemma 2 *Given an embedded plane graph G that is fully triangulated, except for the external face, and given two vertex-disjoint chord-free paths L and R on that external face, it is possible in linear time to partition $V(G)$ into two subsets V_1 and V_2 such that*

1. *the subgraphs of G induced by V_1 and V_2, called G_1 and G_2, are both connected subgraphs;*

2. *there exists exactly one vertex $v \in V(L)$ (say $v \in V_1$) with neighbors in $V_2 \setminus V(L)$ (the opposite vertex set that are not part of $V(L)$), the same holds for $V(R)$; and*

3. *the union U of the set of internal faces in G that are not in G_1 or G_2 forms an outerplane graph with the property that the external face of U is a cycle with no repeated vertices.*

Proof: The proof is a simple modification of the graph and proof of Lemma 1. First, let $V(L) = v_1, v_2, \ldots, v_k$ be the ordered set of vertices in the path of L. We add an edge $e_\ell = (v_1, v_k)$ on the external face; see Fig. 7(a). We do the same for the set R creating an edge e_r. We then proceed as with Lemma 1 with one additional caveat. After expanding p^* to yield the path p in the dual D, let a be the last node visited on p that is incident to a vertex $v \in L$; that is, the face corresponding to a is incident to v. Replace the portion of the path of p from s to a with a path from s to a that visits consecutive nodes incident to v. For clarity, we shall refer to this path as p'; see Figs. 7 and 8. Again, the choice of going clockwise or counterclockwise around v will depend now on the succeeding arcs from a. If the arcs leave counterclockwise from a, then the path from s to a also proceeds counterclockwise around v. Otherwise, it proceeds clockwise around v. Though not always necessary, Fig. 8(c) illustrates an example where this is essential. The same is done with the first node visited on p that is incident to a vertex in R.



(a)

(b)

Figure 7: (a) A graph G with the added edges $e_\ell = (1, 20)$ and $e_r = (5, 17)$. The light vertices correspond to those in L or R. The nodes of the augmented dual D^* are shown labeled by their distance from the start node. One shortest (augmented) path p^* is shown highlighting the nodes visited. (b) The path p^* has been expanded revealing path p. Observe the two highlighted nodes from D correspond to the last, respectively first, node incident to a vertex in L, respectively R. Cutting along this path would place vertices 9, 10 and 16 in the upper graph but with edges extending to the lower graph and would place vertices 13 and 15 in the lower graph but with edges extending to the upper graph.

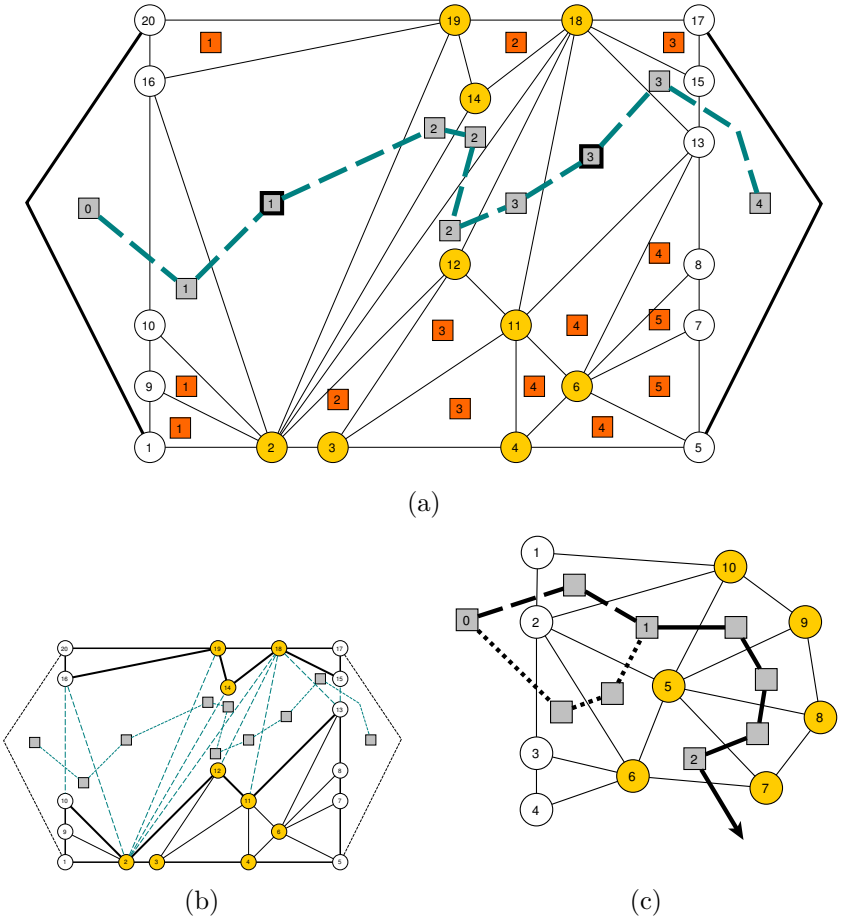


Figure 8: (a) Replacing the path p by pivoting around vertex 16 and vertex 13 to form p'. In this example, the choice of traversing CW or CCW did not matter. Observe that no nodes between the two highlighted nodes can be incident to a vertex in L or R and all other nodes are incident to either 16 or 13. (b) The result after splitting along the selected path p'. (c) Part of a graph depicting an example where the orientation matters. In this case, node 1 is the last visited node on the path p incident to a vertex (2) on L. The path from node 1 to 2 (and further) remains unchanged. Choosing the (dotted) counterclockwise path around vertex 2 causes 5 to be connected by only one edge once the cut is made; the resulting lower graph would have an external cycle that visited 6 twice. Choosing the (dashed) clockwise path avoids the issue.

The construction ensures that conditions (1) and (3) are met as they were with Lemma 1. To see that the new condition (2) is met, we claim that at most two (neighboring) vertices in L are involved in any cut along the path p'. By our construction, the path p' visits a sequence of nodes that are all incident to one vertex $v_i \in L$ starting from node s until node a; see Fig. 8(a). Since we let node a be the last such incident node on the original path p and hence on p', the only nodes visited that are incident to vertices in L must belong to this first sequence of nodes in p'. The first node visited after node s must cross an edge of L, namely, v_i and one of its immediate neighbors in L, which without loss of generality we call v_{i+1}. By construction of the path p', all subsequent nodes until a are also incident to v_i. Let v be any other vertex incident to one of these subsequent nodes. Clearly, v cannot be v_{i+1}. If $v = v_{i-1}$, the other immediate neighbor of v_i, then the path would have returned to the source node and hence p^* was not a shortest path. Finally, v cannot be any other vertex in L because L is chord-free and the edge (v, v_i) would necessarily be a chord.

Therefore, there are at most two vertices v_i and v_{i+1} from L involved in any of the cuts along the path p. Since no vertex is repeated, only one of these two vertices will have other neighboring vertices on the opposite vertex set; see Figs. 8(a-b). The same argument holds for R. □

We can now discuss the steps for the grid drawing of the genus-1 graph G^* with an external face formed by $\mathcal{P}$. Using Lemma 2, with $L = P_4$ and $R = P_2$, divide G^* into two subgraphs G_1 and G_2. We show how to embed G_1, as G_2 is processed in the same fashion. Assume without loss of generality that G_1 contains the bottom path, P_3. Compute a canonical order of G_1 so that the vertices of P_3 are the last vertices removed. This is possible with a simple modification. Introduce three new vertices a, b, c forming a new external face; connect all vertices of P_3 to b and the last vertex of P_3 to c; connect the first and last vertices of P_3 as well as all other vertices on the external cycle to a. The added vertices are ignored from the canonical ordering afterwards; see Figs. 9(a-b).

Place all of the vertices of P_3 consecutively on the horizontal line $y = 0$; $p_{3,k_3}, p_{3,k_3-1}, \ldots, p_{3,1}$; see Fig. 9(c). This is possible since the path is chord-free. As discussed in Section 3.1, the standard dPP algorithm [7] maintains the invariant that at the start of each iteration, the current external face consists of the original horizontal line and a set of line segments of slope ± 1 between consecutive vertices. The algorithm also maintains a "shifting set" for each vertex. We modify this condition by requiring that the vertices on the right and left boundary that are part of P_2 and P_4 be aligned vertically and that the current external face might have horizontal slopes corresponding to vertices from P_3; see Figs. 9(d) and 10(a). Upon insertion of a new vertex v, the vertex will have consecutive neighboring vertices on the external face. We label the left and rightmost neighbors x_ℓ and x_r. To achieve our modified invariant, we insert a vertex v into the current drawing depending on its type, 0, 1, or 2, as follows:

Type 0: Vertices not belonging to a path in $\mathcal{P}$ are inserted as with the tra-

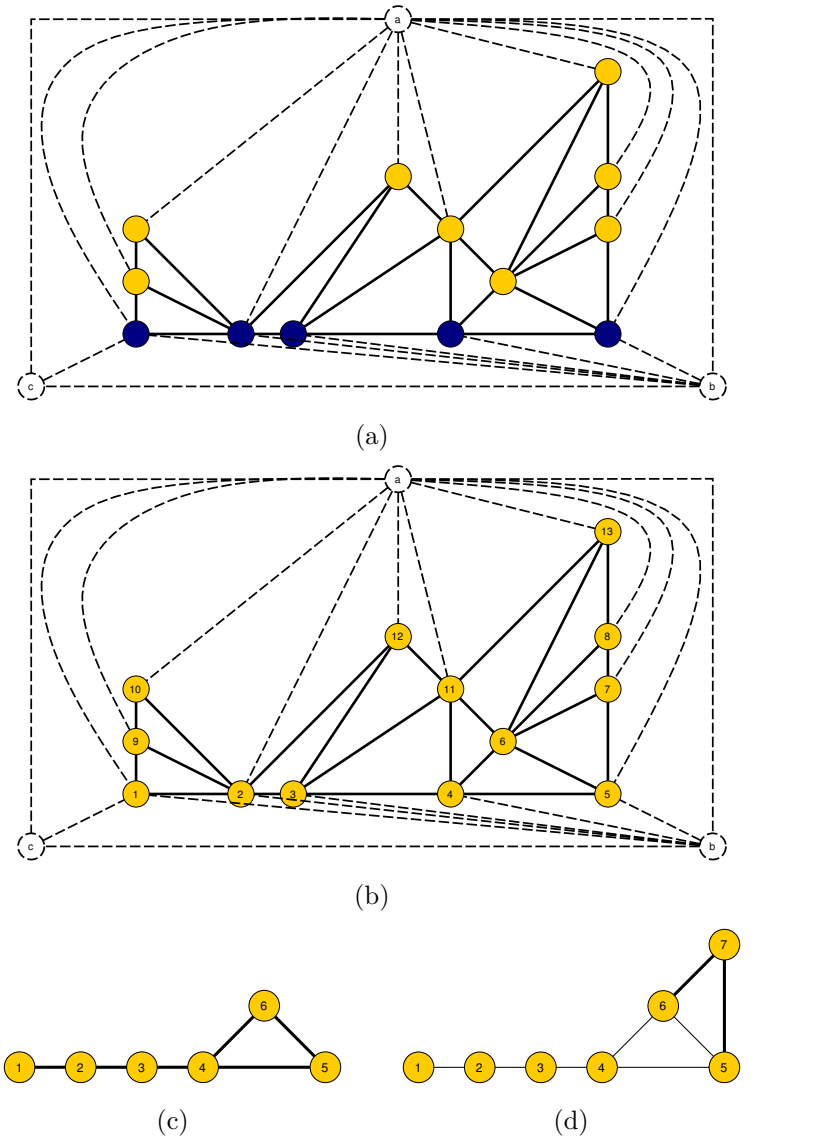


Figure 9: (a) A graph G_1 with no canonical labeling. The external face a, b, c has been added along with its adjacent edges. Though the embedding is fixed, the placement is done for illustrative purposes to show the final positions after the drawing process. The path P_3 is highlighted with darkened nodes. (b) One possible canonical labeling on the vertices, ignoring the added external face. Observe that the vertices of P_3 are labeled 1–5. (c) The initial embedding step placing all of P_3 on a horizontal line and inserting the next vertex 6. (d) The result after adding the vertex 7, which being part of P_2 must lie directly above vertex 5.

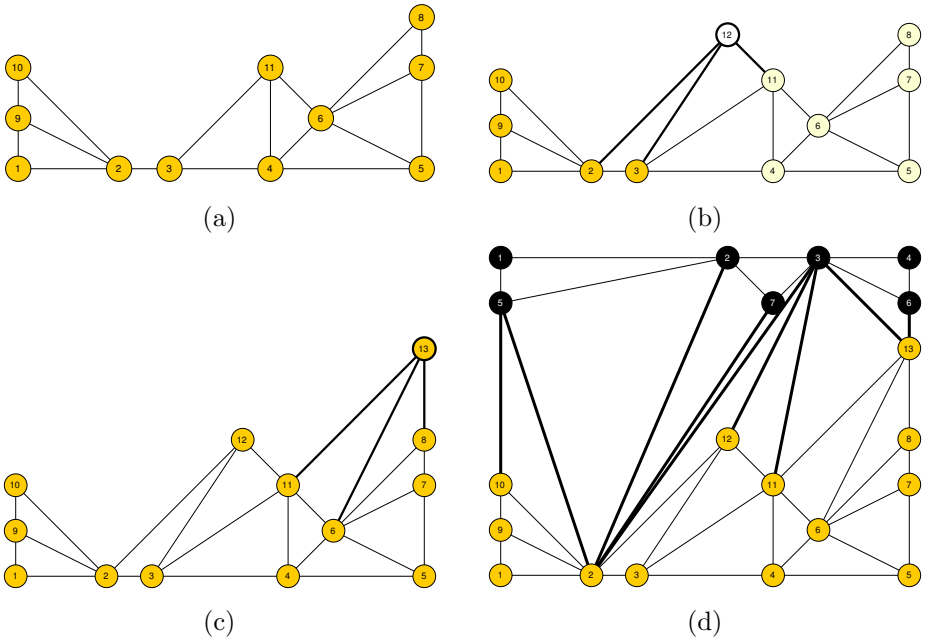


Figure 10: (a) The embedding process after insertion of the first 11 vertices. Note the invariant condition allowing horizontal lines along the bottom row $\{1, 2, 3, 4, 5\} = P_3$ and the two partial vertical walls $\{8, 7, 5\} \subset P_2$ and $\{1, 9, 10\} \subset P_4$. (b) The subsequent insertion of a Type 0 vertex with $v = 12$, $x_\ell = 2$, and $x_r = 11$. The light vertices to the right of 12 including x_r have been shifted over one unit. (c) The result of inserting a Type 1 vertex with $v = 13$, $x_\ell = 11$, and $x_r = 8$. In this particular instance, we did not have to shift. (d) Two subgraphs G_1 and G_2 stitched together, the thicker edges are the edges initially removed in the separation phase. Although p can be used as the point of placement, as this example illustrates, it is possible to compress the stitch further.

ditional dPP algorithm. That is, the vertex v is connected to its two neighboring vertices x_ℓ and x_r on the external face using a slope of $+1$ and -1. This insertion might require up to two horizontal shifts determined by the shifting sets; see Fig. 10(b). The remaining neighboring vertices $x_{\ell+1}, \ldots, x_{r-1}$ are connected with straight line segments as usual.

Type 1: Vertices belonging to P_2, which must be placed vertically along the right boundary, are inserted with a line segment of slope $+1$ between x_ℓ and v and a vertical line segment between v and x_r. Notice that x_r must also be in P_2. And because P_2 is chord-free x_r is the topmost vertex on the right side of the current external face. That is, v can see x_r. By Lemma 2 and the fact that the graph was fully triangulated, we also know that v must have a vertex x_ℓ. This insertion requires only 1 shift, for the visibility of x_ℓ and v. Again the remaining vertices $x_{\ell+1}, \ldots, x_{r-1}$ are connected as usual; see Fig. 10(c).

Type 2: Vertices belonging to P_4, which must be placed vertically along the left boundary, are handled similarly to Type 1 vertices.

Because of Lemma 2, after processing both G_1 and G_2, we can stitch the two portions together; see Fig. 10(d). Shift the left wall of the narrower graph sufficiently to match the width of the other graph. Thus, both graphs have width W. For simplicity, refer to the vertices on the external face of each subgraph that are not exclusively part of the wall or bottom row as *upper external vertices*. For each subgraph, consider the point p located at the intersection of the lines of slope ± 1 extending from the left and rightmost external vertices. Flip G_2 vertically placing it so that its point p lies either on or just above (in case of non-integer intersection) G_1's point. Because the edges between the upper external vertices have slope $|m| \leq 1$ and because of the vertical separation of the two subgraphs, every upper external vertex on G_1 can directly see every upper external vertex on G_2. By Lemma 2, we know that the set of edges removed in the separation along with the edges connecting the upper external vertices forms an outerplanar graph. Therefore, we can reconnect the removed edges, joining the two subgraphs, without introducing any crossings.

We claim that the area of this grid is $O(n) \times O(n^2)$. First, let us analyze the width. From our discussion, we have accounted for each insertion step using shifts. Since the maximum amount of shifting of 2 units is done with Type 0 vertices, we know that each of the two subgraphs has width at most $2n$. In addition, the stitching stage only required a shifting of the smaller width subgraph. Therefore, the width of our drawing is at most $2n$. After completion of the traditional dPP algorithm the height is half the width, because the edges on the external face have slopes ± 1. We can only say this about the upper vertices. Fortunately, after the insertion of each wall vertex we know that the height increases by at most W. Therefore, we know that the height is at most Wn or $2n^2$ and consequently we have a correct drawing using a grid of size $O(n) \times O(n^2)$. □

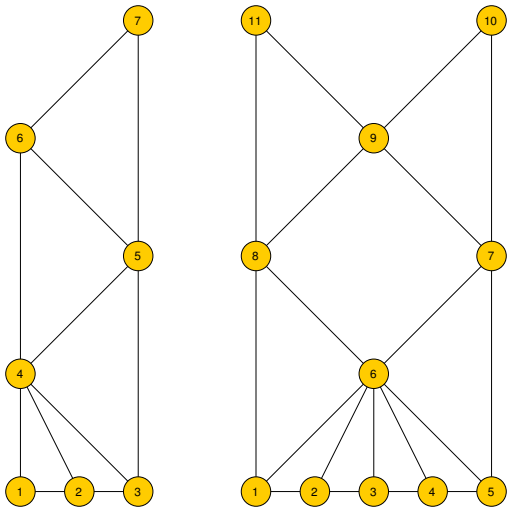


Figure 11: Two examples forcing a height that grows quadratically, as our method is currently described. A relaxation of the slope constraints could resolve this issue.

Thus, we get a planar polygonal-schema drawing of a toroidal graph G in a rectangle, which simultaneously achieves polynomial area, a straight frame, and straight-line edges. Ideally, the height of our drawing would also match the linear width bound. The stitching stage for example only adds at most $W \leq 2n$ units to the final height. Unfortunately, as Fig. 11 illustrates, a problem arises when vertices of Type 1 and Type 2 interconnect.

In particular, let the width at the start of an iteration be W. If we add a vertex v on the left wall that is connected to a vertex on the right wall, then this vertex must be inserted at a height W above the vertex on the right wall, to maintain proper slope. Repeating this, leads to a height of the grid that is quadratic in the width of the grid. As the underlying dPP algorithm requires width linear in the number of vertices in the graph, we get the $O(n) \times O(n^2)$ area.

3.3 The Peel-and-Bend Algorithm

The case for $g > 1$ is similar but involves a few alterations. First, we use $n = |V|$ unlike the previous sections which used $n = |V| + g$. The main difference, however, is that we cannot embed the outer face using only horizontal and vertical walls unless the union of the fundamental cycles is chord-free or unless edge bends are allowed. If we desire a straight-frame rendering of the polygonal schema $\mathcal{P}$ in a rectangle, we must allow some edge bends. The following theorem describes our resulting drawing method, which we call the *Peel-and-Bend* algorithm.

Theorem 4 *Let G^* be an embedded planar graph and $\mathcal{P} = \{P_1, P_2, \ldots, P_{4g}\}$ in G^* be a collection of $4g$ paths such that each path $P_i = \{p_{i,1}, p_{i,2}, \ldots, p_{i,k_i}\}$ is chord-free, the last vertex of each path matches the first vertex of the next path, ($p_{i,k_i} = p_{i+1,1}$) including P_{4g} and P_1, and when treated as a single cycle, $\mathcal{P}$ forms the external face of G^*. Let $k = \sum_{i=1}^{4g}(k_i - 1)$ be the number of vertices on the external cycle. We can draw G^* on an $O(n) \times O(n^2)$ grid with straight-line edges and no crossings and at most $k - 3$ single-bend edges in such a way that for each path P_i on the external face the vertices on that path form a straight line. Moreover, the external face forms a rectangle.*

Proof: First, let us assume that the entire external face, represented by $\mathcal{P}$, is completely chord-free. That is, if two vertices on the external cycle share an edge then they are adjacent on the cycle. In this case we can create a new set of 4 paths, $\mathcal{P}' = \{P_1, \cup_{i=2,\ldots,2g} P_i, P_{2g+1}, \cup_{i=2g+2,4g} P_i\}$. We can then use Theorem 3 to prove our claim using no bends.

If, however, there exist chords on the external face, embedding the graph with straight-lines becomes problematic, and in fact impossible to do using a rectangular outer face. By introducing a temporary bend vertex for each chord and retriangulating the two neighboring faces, we can make the external face chord free. Clearly, this addition can be done in linear time. Since there are at most k vertices on the external face and since the graph is planar, there are no more than $k - 3$ such bend points to add. We then proceed as before using Theorem 3, subsequently replacing each temporary vertex with a bend point and deleting the added edges. □

4 Algorithms for Non-Toroidal Graphs

In this section, we describe two more algorithms for producing a planar polygonal-schema drawing of a non-toroidal graph G, which is given together with its combinatorial embedding on a genus-g surface, $\mathcal{S}$, where $g > 1$. As mentioned above, these algorithms provide alternative trade-offs with respect to the three primary aesthetic criteria for polygonal-schema drawings: straight-line edges, straight frames and polynomial area.

The Peel-and-Stretch Algorithm. In the Peel-and-Stretch Algorithm, we find a chord-free polygonal schema $\mathcal{P}$ for G and cut G along these edges to form a planar graph G^*. We then lay out the sides of $\mathcal{P}$ in a straight-frame manner as a regular convex polygon, with each fundamental path forming one side of the polygon and the vertices along each boundary edge spaced as evenly as possible. We then fix this as the outer face of G^* and apply Tutte's algorithm [18, 19] to construct a straight-line drawing of the rest of G^*. This algorithm therefore achieves a drawing with straight-line edges in a (regular) straight frame, but it may require exponential area when normalized to an integer grid, since Tutte's drawing algorithm may generate vertices with coordinates that require $\Theta(n \log n)$ bits to represent.

The Peel-and-Place Algorithm. In the Peel-and-Place Algorithm, we start by finding a polygonal schema $\mathcal{P}$ for G and cut G along these edges to form a planar graph G^*, as in all our algorithms. In this particular case, we next create a new triangular face, T, and place G^* in the interior of T. After fully triangulating this graph, we then apply the standard dPP algorithm [7] or the planar drawing algorithm of Schnyder [16] to construct a drawing of this graph in an $O(n) \times O(n)$ integer grid with straight-line edges. Finally, we remove all extra edges to produce a polygonal schema drawing of G. The result will be a polygonal-schema drawing with straight-line edges having polynomial area, but there is no guarantee that it is a straight-frame drawing, since the two algorithms make no collinear guarantees for vertices adjacent to the vertices on the bounding triangle.

5 Conclusion and Future Work

In this paper, we present several algorithms for polygonal-schema drawings of higher-genus graphs. Our method for toroidal graphs achieves drawings that simultaneously use straight-line edges in a straight frame and polynomial area. Previous algorithms for the torus are restricted to special cases or do not always produce polygonal-schema renderings [5, 10, 20]. Our methods for non-toroidal graphs can achieve any two of these three criteria. To the best of our knowledge, previous algorithms for general graphs in genus-g surfaces were restricted to those with "nice" polygonal schemas [21]. The problem of whether it is possible to achieve all three of these aesthetic criteria for non-toroidal graphs has recently been resolved in [1]. However, we feel one big open problem in both the current paper and the more recent generalization is embedding the drawing such that the positions of matching vertices coincide. This is a common way to depict the split vertices particularly in the rendering of a toroidal graph; see for example Fig. 1(b). Reducing the area bound established in both papers is another important research direction as discussed at the end of Section 3.2.

Acknowledgments

The authors would like to thank the anonymous referees for their numerous helpful suggestions for improving this paper.

References

[1] E. Chambers, D. Eppstein, M. T. Goodrich, and M. Löffler. Drawing graphs in the plane with a prescribed outer face and polynomial area. *Proc. 18th Int. Symp. on Graph Drawing (GD 2010)*. Springer-Verlag, to appear, arXiv:1009.0088.

[2] J. Chen, S. P. Kanchi, and A. Kanevsky. A note on approximating graph genus. *Inform. Process. Lett.* 61(6):317–322, 1997, doi:10.1016/S0020-0190(97)00203-2.

[3] M. Chrobak and T. Payne. A linear-time algorithm for drawing planar graphs. *Inform. Process. Lett.* 54(4):241–246, 1995.

[4] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, Englewood Cliffs, NJ, 1999.

[5] D. Eppstein. The topology of bendless three-dimensional orthogonal graph drawing. *Graph Drawing: 16th Int. Symp. (GD 2008)*, pp. 78–89, 2009, doi:10.1007/978-3-642-00219-9_9.

[6] J. Erickson and S. Har-Peled. Optimally cutting a surface into a disk. *Discrete Comput. Geom.* 31(1):37–59, 2004.

[7] H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica* 10(1):41–51, 1990.

[8] M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM J. Algebraic Discrete Methods* 4(3):312–316, 1983.

[9] G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica* 16:4–32, 1996.

[10] W. Kocay, D. Neilson, and R. Szypowski. Drawing graphs on the torus. *Ars Combinatoria* 59:259–277, 2001.

[11] F. Lazarus, M. Pocchiola, G. Vegter, and A. Verroust. Computing a canonical polygonal schema of an orientable triangulated surface. *Proc. of the 17th ACM Symp. on Computational Geometry (SCG)*, pp. 80–89, 2001.

[12] M. Middendorf and F. Pfeiffer. On the complexity of the disjoint paths problem. *Combinatorica* 13(1):97–107, 1993.

[13] K. Miura, S.-I. Nakano, and T. Nishizeki. Grid drawings of 4-connected plane graphs. *Discrete and Computational Geometry* 26(1):73–87, 2001.

[14] B. Mohar and C. Thomassen. *Graphs on Surfaces*. Johns Hopkins University Press, 2001.

[15] T. Nishizeki and N. Chiba. *Planar Graphs: Theory and Algorithms.* North-Holland, Amsterdam, 1988, p. 232.

[16] W. Schnyder. Embedding planar graphs on the grid. *Proceedings of the 1st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 138–148, 1990.

[17] C. Thomassen. The graph genus problem is NP-complete. *J. Algorithms* 10(4):568–576, 1989, doi:10.1016/0196-6774(89)90006-0.

[18] W. T. Tutte. Convex representations of graphs. *Proc. London Math. Society* 10(38):304–320, 1960.

[19] W. T. Tutte. How to draw a graph. *Proc. London Math. Society* 13(52):743–768, 1963.

[20] A. Vodopivec. On embeddings of snarks in the torus. *Discrete Mathematics* 308(10):1847–1849, 2008.

[21] A. Zitnik. Drawing graphs on surfaces. *SIAM J. Disc. Math.* 7(4):593–597, 1994.