Planar Drawings of Higher-Genus Graphs

Christian A. Duncan[1], Michael T. Goodrich[2,⋆], and Stephen G. Kobourov[3,⋆⋆]

[1] Dept. of Computer Science, Louisiana Tech Univ.
http://www.latech.edu/~duncan/
[2] Dept. of Computer Science, Univ. of California, Irvine
http://www.ics.uci.edu/~goodrich/
[3] Dept. of Computer Science, University of Arizona
http://www.cs.arizona.edu/~kobourov/

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. 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.

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. 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 [2].

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

⋆ Work partially supported by NSF grants OCI-0724806, IIS-0713046, CCR-0830403, and ONR MURI N0014-08-1-1015.
⋆⋆ Work partially supported by NSF CAREER grant CCF-0545743.

D. Eppstein and E.R. Gansner (Eds.): GD 2009, LNCS 5849, pp. 45–56, 2010.

and, when that is not easily done, to produce planar grid drawings with edges drawn as simple polygonal chains.

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 [6]. 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 $O(|V| + g)$ [10].

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 [11]. 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) [1]. 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. The traditional method for performing such a cutting is with a *canonical polygonal schema*, $\mathcal{P}$, which is a set of $2g$ cycles on $\mathcal{S}$ all containing a common point, p, such that cutting $\mathcal{S}$ along these cycles results in a topological disk. These cycles are *fundamental* in that each of them is a continuous closed curve on $\mathcal{S}$ that cannot be retracted continuously to a point. Moreover, these 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, so that these two sides of $\mathcal{P}$ are matched in orientation on $\mathcal{S}$. Thus, the canonical polygonal schema for a genus-g surface $\mathcal{S}$ has $4g$ sides that are pairwise identified.

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 [8] and furthermore, as we show in [3], 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:

- 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 edge of the polygonal schema should be rendered as a straight line segment, with the vertices and edges of the corresponding fundamental cycle, placed along this segment. 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. 1.

Our Contributions. We provide several methods for producing planar polygonal-schema drawings of higher-genus graphs. In particular, we provide four algorithms, one for torodial ($g = 1$) 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*, achieve two of the three aesthetic criteria and differ in which criteria they fail to meet.

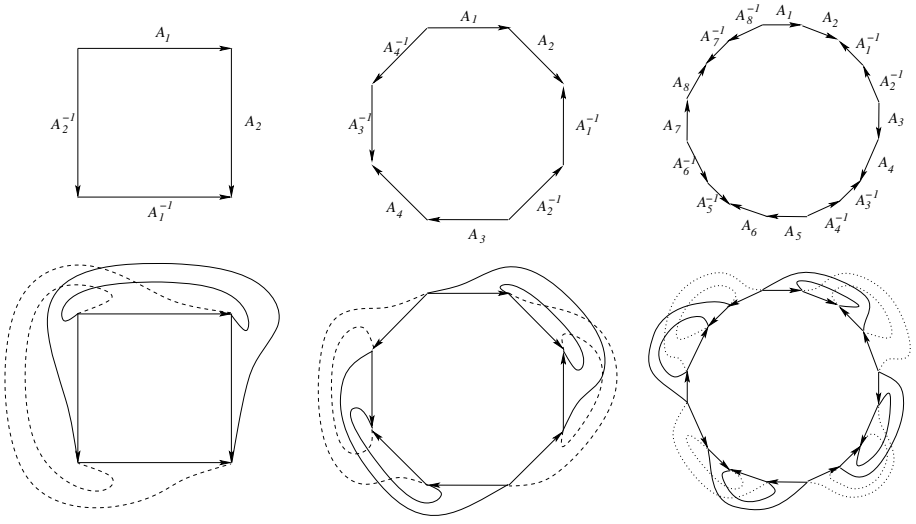


Fig. 1. First row: Canonical polygonal schemas for graphs of genus one, two and four. Second row: Unrolling the high genus graphs with the aid of the overpasses and underpasses.

2 Finding Polygonal Schemas

Suppose we are given a graph G together with its cellular embedding in 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.* [8] 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 (recall that 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 polygonal schema). In addition, we show in [3] that finding 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 above, a canonical polygon schema of a graph G 2-cell embedded in a surface of genus g consists

of $4g$ sides, which correspond to $2g$ fundamental cycles all containing a common vertex. Lazarus *et al.* [8] 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 [5] 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.

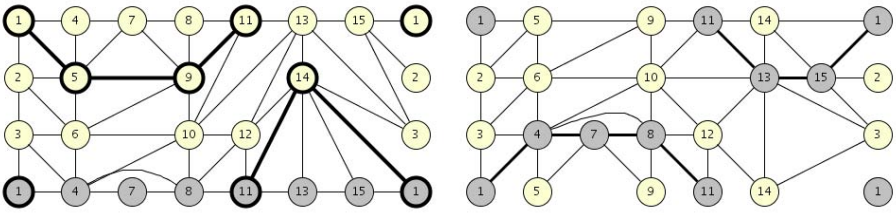
2.2 Constructing Chord-Free Polygonal Schemas

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 the 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 Figure 2(a). We show next that a new chord-free polygonal schema can be efficiently determined from the original schema.

Theorem 1. *Given a graph G combinatorially embedded in 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 topological disk; see Fig. 2(a). Notice this cutting will cause certain vertices to be split into multiple vertices. For each fundamental cycle in $c_i \in \mathcal{P}$, we 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 will have two copies of the 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



(a) (b)

Fig. 2. (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 vertex 1. (b) The topological disk *after* cutting along this new fundamental cycle. The grey nodes show the old fundamental cycle.

chosen was not the shortest path; see Fig. 2(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.

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. We rely on a modified version of the algorithm of de Fraysseix, Pach and Pollack [2] for the drawing. Sections 3.1 and 3.2 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 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 2. *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, and when treated as a single cycle, $\mathcal{P}$ forms the external face of G^*. If $g = 1$, 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.*

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) [2] does not directly solve our problem because of the additional requirement for the drawing of the external face. In the case of $g = 1$, 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.

Recall that 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. 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. Consequently, we propose an approach similar to that of Miura, Nakano, and Nishizeki [9]. 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.

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 $e_l = (u_l, v_l)$ and $e_r = (u_r, v_r)$, we have $u_l, u_r \in V_1$ and $v_l, v_r \in V_2$;*
3. *the union U of the set of 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 ignore the external face in this step. For clarity we shall refer to vertices and edges in the primal and nodes and arcs in the dual; see Fig. 3(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^*.

Let the source node s be the node corresponding to the edge e_l and the sink node t be the node corresponding to the edge e_r. We then perform a breadth-first shortest-path traversal from s to t on D^*; see Fig. 3(b). Let p^* be a shortest (augmented) path in D^* obtained by this search. We now create a (regular) path p by expanding the augmented arcs added. That is, if there is an arc $(u, v) \in p^*$ such that u and v 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; see Fig. 3(c).

All of the steps described above can be easily implemented in linear time. The details of the proof can be found in [3]. □

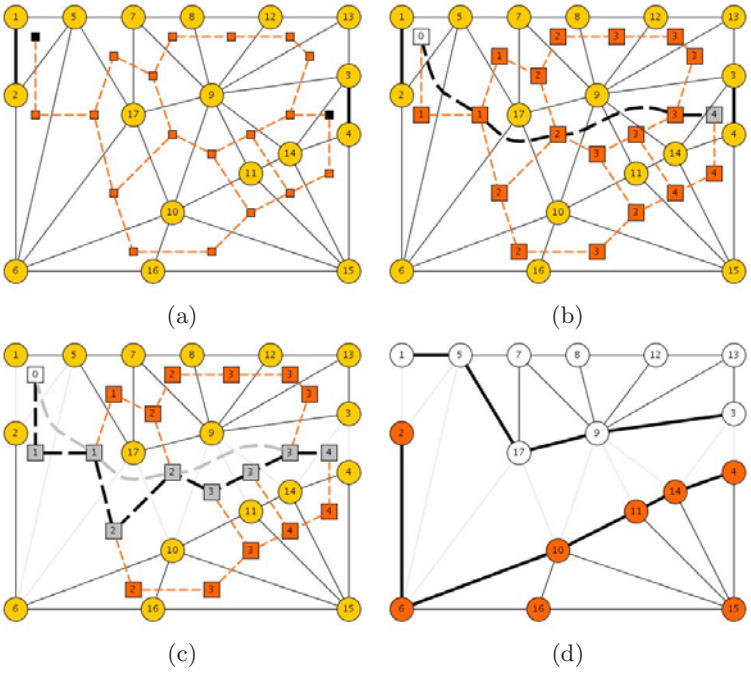


Fig. 3. (a) A graph G and its dual D. The dark edges/nodes represent the sink and source nodes. (b) Each dual node is labeled with its distance (in D^*) from the start node 0. A shortest path p^* is drawn with thick dark arcs. This path includes the augmented arcs of D^*. (c) The path p formed after 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 $(3, 4)$.

Figure 3(d) illustrates the result of one such partition. In some cases we might have to start and end with a set of edges rather than just the two edges e_l and e_r. The following extension of Lemma 1 addresses this issue; the details of the proof can be found in [3].

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 $V(L) \setminus V_2$ (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 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.*

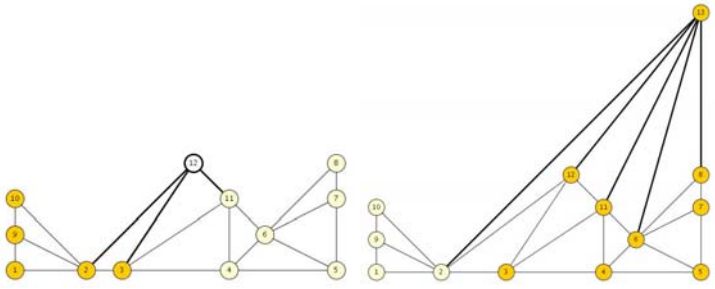
We can now discuss the steps for the grid drawing of the genus-1 graph G^* with an external face formed by $\mathcal{P}$. Using Lemma 2, with $L = P_4$ and $R = P_2$, divide G^* into two subgraphs G_1 and G_2. We proceed to embed G_1 with G_2 being symmetric. Assume without loss of generality that G_1 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. Place all of the vertices of P_3 on a horizontal line, $p_{3,k_3}, p_{3,k_3-1}, \ldots, p_{3,1}$ placed consecutively on $y = 0$. This is possible since there are no edges between them (because the path is chord-free). Recall that the standard dPP algorithm [2] 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 Fig. 4(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 traditional dPP algorithm. This insertion might require up to two horizontal shifts determined by the shifting sets; see Fig. 4(a).

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. 4(b).

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

Because of Lemma 2, after processing both G_1 and G_2, we can proceed to stitch the two portions together. Shift the left wall of the narrower graph sufficiently to match the width of the other graph. 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



(a) (b)

Fig. 4. (a) The embedding process after insertion of the first 11 vertices and the subsequent insertion of a Type 0 vertex with $v = 12$, $x_\ell = 2$, and $x_r = 11$. Note the invariant condition allowing the two partial vertical walls $\{8, 7, 5\} \subset P_2$ and $\{1, 9, 10\} \subset P_4$. The light vertices to the right of 12 including x_r have been shifted over one unit. (b) The result of inserting a Type 1 vertex with $v = 13$, $x_\ell = 2$, and $x_r = 8$. Note, the light vertices to the left of and including $x_\ell = 2$ are shifted over one unit.

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 $2n$. The stitching stage for example only adds at most $W \leq 2n$ units to the final height. 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 $O(n) \times O(n^2)$. Ideally, the height of our drawing would also match the width bound. □

3.2 The Peel-and-Bend Algorithm

The case for $g > 1$ is similar but involves a few alterations. First, we use $n = |V|$ unlike prior sections which used $n = |V| + g$. However, the main difference is that having chord-free fundamental cycles is insufficient to allow rendering the outer face as a rectangle unless edge bends are allowed. The following theorem describes our resulting drawing method, called the *Peel-and-Bend* algorithm.

Theorem 3. *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, 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.

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 2 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 2, subsequently replacing inserted vertices with a bend point. □

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 in an (orientable) 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 we desire for polygonal-schema drawings. For the sake of space, we describe these algorithms at a very high level and leave their details and full analysis to the full version of this paper [3].

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 layout the sides of $\mathcal{P}$ in a straight-frame manner as a regular convex polygon, with 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 [12,13] 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. For this method, 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. We then create a new triangular face, T, place G^* in the interior of T, and fully triangulate this graph. We then apply the dPP algorithm [2] 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 dPP algorithm makes 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 were restricted to special cases or did not always produce polygonal-schema renderings [4,7,14]. Our methods for non-toroidal graphs can achieve any two of these three criteria. It is an open problem whether it is possible to achieve all three of these aesthetic criteria for non-toroidal graphs. To our knowledge, previous algorithms for general graphs in genus-g surfaces were restricted to those with "nice" polygonal schemas [15].

References

1. Chen, J., Kanchi, S.P., Kanevsky, A.: A note on approximating graph genus. Information Processing Letters 61(6), 317–322 (1997)
2. de Fraysseix, H., Pach, J., Pollack, R.: How to draw a planar graph on a grid. Combinatorica 10(1), 41–51 (1990)
3. Duncan, C.A., Goodrich, M.T., Kobourov, S.G.: Planar drawings of higher-genus graphs. Technical report (August 2009), http://arxiv.org/abs/0908.1608
4. Eppstein, D.: The topology of bendless three-dimensional orthogonal graph drawing. In: Tollis, I.G., Patrignani, M. (eds.) GD 2008. LNCS, vol. 5417, pp. 78–89. Springer, Heidelberg (2009)
5. Erickson, J., Har-Peled, S.: Optimally cutting a surface into a disk. In: Proc. of the 18th ACM Symp. on Computational Geometry (SCG), pp. 244–253 (2002)
6. Garey, M.R., Johnson, D.S.: Crossing number is NP-complete. SIAM J. Algebraic Discrete Methods 4(3), 312–316 (1983)
7. Kocay, W., Neilson, D., Szypowski, R.: Drawing graphs on the torus. Ars Combinatoria 59, 259–277 (2001)
8. Lazarus, F., Pocchiola, M., Vegter, G., Verroust, A.: Computing a canonical polygonal schema of an orientable triangulated surface. In: Proc. of the 17th ACM Symp. on Computational Geometry (SCG), pp. 80–89 (2001)
9. Miura, K., Nakano, S.-I., Nishizeki, T.: Grid drawings of 4-connected plane graphs. Discrete and Computational Geometry 26(1), 73–87 (2001)
10. Mohar, B., Thomassen, C.: Graphs on Surfaces. Johns Hopkins U. Press, Baltimore (2001)
11. Thomassen, C.: The graph genus problem is NP-complete. J. Algorithms 10(4), 568–576 (1989)
12. Tutte, W.T.: Convex representations of graphs. Proceedings London Mathematical Society 10(38), 304–320 (1960)
13. Tutte, W.T.: How to draw a graph. Proc. Lon. Math. Soc. 13(52), 743–768 (1963)
14. Vodopivec, A.: On embeddings of snarks in the torus. Discrete Mathematics 308(10), 1847–1849 (2008)
15. Zitnik, A.: Drawing graphs on surfaces. SIAM J. Disc. Math. 7(4), 593–597 (1994)