Smooth Orthogonal Drawings of Planar Graphs*

Muhammad Jawaherul Alam[1], Michael A. Bekos[2], Michael Kaufmann[2], Philipp Kindermann[3], Stephen G. Kobourov[1], and Alexander Wolff[3]

[1] Department of Computer Science, University of Arizona, USA
{mjalam,kobourov}@cs.arizona.edu
[2] Wilhelm-Schickhard-Institut für Informatik, Universität Tübingen, Germany
{bekos,mk}@informatik.uni-tuebingen.de
[3] Lehrstuhl für Informatik I, Universität Würzburg, Germany
http://www1.informatik.uni-wuerzburg.de/en/staff

Abstract. In *smooth orthogonal layouts* of planar graphs, every edge is an alternating sequence of axis-aligned segments and circular arcs with common axis-aligned tangents. In this paper, we study the problem of finding smooth orthogonal layouts of low *edge complexity*, that is, with few segments per edge. We say that a graph has *smooth complexity* k—for short, an SC_k-layout—if it admits a smooth orthogonal drawing of edge complexity at most k.

Our main result is that every 4-planar graph has an SC_2-layout. While our drawings may have super-polynomial area, we show that for 3-planar graphs, cubic area suffices. We also show that any biconnected 4-outerplane graph has an SC_1-layout. On the negative side, we demonstrate an infinite family of biconnected 4-planar graphs that require exponential area for an SC_1-layout. Finally, we present an infinite family of biconnected 4-planar graphs that do not admit an SC_1-layout.

1 Introduction

In the visualization of technical networks such as the structure of VLSI chips [8] or UML diagrams [10] there is a strong tendency to draw edges as rectilinear paths. The problem of laying out networks in such a way is called *orthogonal graph drawing* and has been studied extensively. For drawings of (planar) graphs to be readable, special care is needed to keep the number of bends small. In a seminal work, Tamassia [11] showed that one can efficiently minimize the total number of bends in orthogonal layouts of *embedded 4-planar graphs*, that is, planar graphs of maximum degree 4 whose combinatorial embedding (the cyclic order of the edges around each vertex) is given. In contrast to this, minimizing the number of bends over all embeddings of a 4-planar graph is NP-hard [6].

* Research of M.J. Alam and S.G. Kobourov is supported in part by NSF grants CCF-1115971 and DEB 1053573. The work of M.A. Bekos is implemented within the framework of the Action "Supporting Postdoctoral Researchers" of the Operational Program "Education and Lifelong Learning" (Action's Beneficiary: General Secretariat for Research and Technology), and is co-financed by the European Social Fund (ESF) and the Greek State. M. Kaufmann as well as Ph. Kindermann and A. Wolff acknowledge support by the ESF EuroGIGA project GraDR (DFG grants Ka 812/16-1 and Wo 758/5-1, respectively).

A. Pardo and A. Viola (Eds.): LATIN 2014, LNCS 8392, pp. 144–155, 2014.

In a so far unrelated line of research, circular-arc drawings of graphs have become a popular matter of research in the last few years. Inspired by American artist Mark Lombardi (1951–2000), Duncan et al. [4] introduced and studied *Lombardi drawings*, which are circular-arc drawings with the additional requirement of *perfect angular resolution*, that is, for each vertex, all pairs of consecutive edges form the same angle. Among others, Duncan et al. treat drawings of d-regular graphs where all vertices have to lie on one circle. They show that under this restriction, for some subclasses, Lombardi drawings can be constructed efficiently, whereas for the others, the problem is NP-hard. They also show [5] that ordered trees can always be Lombardi drawn in polynomial area, whereas straight-line drawings with perfect resolution may need exponential area.

Very recently, Bekos et al. [2] introduced the *smooth orthogonal* graph layout problem that combines the two worlds; the rigidity and clarity of orthogonal layouts with the artistic style and aesthetic appeal of Lombardi drawings. Formally, a smooth orthogonal drawing of a graph is a drawing on the plane where (i) each vertex is drawn as a point; (ii) edges leave and enter vertices horizontally or vertically, (iii) each edge is drawn as an alternating sequence of axis-aligned line segments and circular-arc segments such that consecutive segments have a common *horizontal* or *vertical* tangent at their intersection point. In the case of (4-) planar graphs, it is additionally required that (iv) there are no edge-crossings. Note that, by construction, (smooth) orthogonal drawings of 4-planar graphs have angular resolution within a factor of two of optimal.

Figure 1 shows a real-world example: a smooth orthogonal drawing of an Austrian regional bus and train map. Extending our model, the map has (multi-) edges that enter vertices diagonally (as in *Grünau im Almtal Postamt*; bottom right).

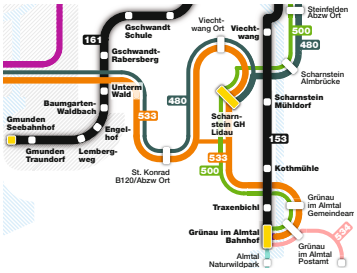


Fig. 1. Clipping of the public transport map *Gmunden – Vöcklabruck – Salzkammergut*, Austria [12]

For usability, it is important to keep the visual complexity of such drawings low. In a (smooth) orthogonal drawing, the *complexity* of an edge is the number of segments it consists of, that is, the number of inflection points plus one. Then, a natural optimization goal is to minimize, for a given (embedded) planar graph, the *edge complexity* of a drawing, which is defined as the maximum complexity over all edges. We say that a graph has *orthogonal complexity* k if it admits an orthogonal drawing of edge complexity at most k, for short, an *OC_k-layout*. Accordingly, we say that a graph has *smooth complexity* k if it admits a smooth orthogonal drawing of edge complexity at most k, for short, an *SC_k-layout*. We seek for drawings of 4-planar graphs with low smooth complexity.

Our Contribution. Known results and our contributions to smooth orthogonal drawings are shown in Table 1. The main result of our paper is that any 4-planar graph admits an SC_2-layout (see Sections 2 and 3). Our upper bound of 2 for the smooth complexity of 4-planar graphs improves the previously known bound of 3 and matches the corresponding lower bound [2]. In contrast to the known algorithm for SC_3-layout [2], which is based on an algorithm for OC_3-layout of Biedl and Kant [3], we use an algorithm of Liu at al. [9] for OC_3-layout, which avoids *S-shaped edges* (see Figure 2b, top). Such edges

Table 1. Comparison of our results to the results of Bekos et al. [2]

graph class	our contribution			Bekos et al. [2]
	complexity	area	reference	
biconnected 4-planar	SC_2	super-poly	Theorem 1	SC_3
4-planar	SC_2	super-poly	Theorem 2	
3-planar	SC_2	$\lfloor n^2/4 \rfloor \times \lfloor n/2 \rfloor$	Theorem 3	
biconnected 4-outerplane	SC_1	exponential	Theorem 4	
triconnected 3-planar				SC_1
Hamiltonian 3-planar				SC_1
4-planar, poly-area	$\not\supseteq \mathrm{SC}_1$		full version [1]	
OC_3, octahedron				$\not\subseteq \mathrm{SC}_1$
OC_2	$\not\subseteq \mathrm{SC}_1$		full version [1]	

are undesirable since they force their endpoints to lie on a line of slope ± 1 in a smooth orthogonal layout (see Figure 2b, bottom). Our construction requires super-polynomial area. Hence, we have made no effort in proving a concrete bound.

Further, we prove that every biconnected 4-outerplane graph admits an SC_1-layout (see Section 4), expanding the class of graphs with SC_1-layout from triconnected or Hamiltonian 3-planar graphs [2]. Note that in our result, the outerplane embedding can be prescribed, while in the other results the algorithms need the freedom to choose an appropriate embedding.

We complement our positive results by two negative results; see the last three lines of Table 1. Due to lack of space, the detailed proofs are given in [1]. Many problems remain open: does polynomial area suffice for SC_2-layouts of 4-planar graphs? Do larger graph classes admit SC_1? What's the computational complexity of deciding SC_1?

2 Smooth Layouts for Biconnected 4-Planar Graphs

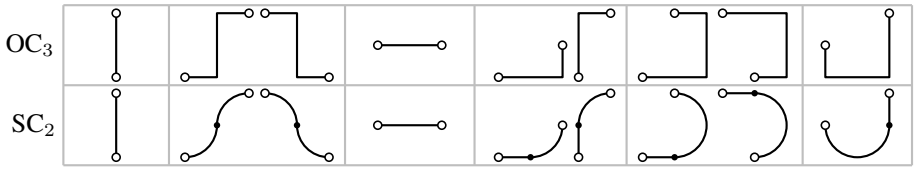
In this section, we prove that any biconnected 4-planar graph admits an SC_2-layout. Given a biconnected 4-planar graph, we first compute an OC_3-layout, using an algorithm of Liu et al. [9]. Then we turn the result of their algorithm into an SC_2-layout.

Liu et al. choose two vertices s and t and compute an st-ordering of the input graph. An st-ordering is an ordering $(s = 1, 2, \ldots, n = t)$ of the vertices such that every j $(2 < j < n-1)$ has neighbors i and k with $i < j < k$. Then they go through all vertices as prescribed by the st-ordering, placing vertex i in row i. Calling an edge of which exactly one end-vertex is already drawn an *open edge*, they maintain the following invariant:

(I_1) In each iteration, every open edge is associated with a *column* (a vertical grid line).

An algorithm of Biedl and Kant [3] yields an OC_3-layout similar to that of Liu et al. However, Liu et al. additionally show how to modify their algorithm such that it produces OC_3-layouts without the undesirable S-shapes; see Fig. 2b (top).

In their modified algorithm, Liu et al. search for paths in the drawing that consist only of S-shapes; every vertex lies on at most one such path. They place all vertices on



(a) vertical (b) S-shapes (c) horizontal (d) L-shapes (e) C-shapes (f) U-shape

Fig. 2. Converting shapes from the OC_3-layout to SC_2

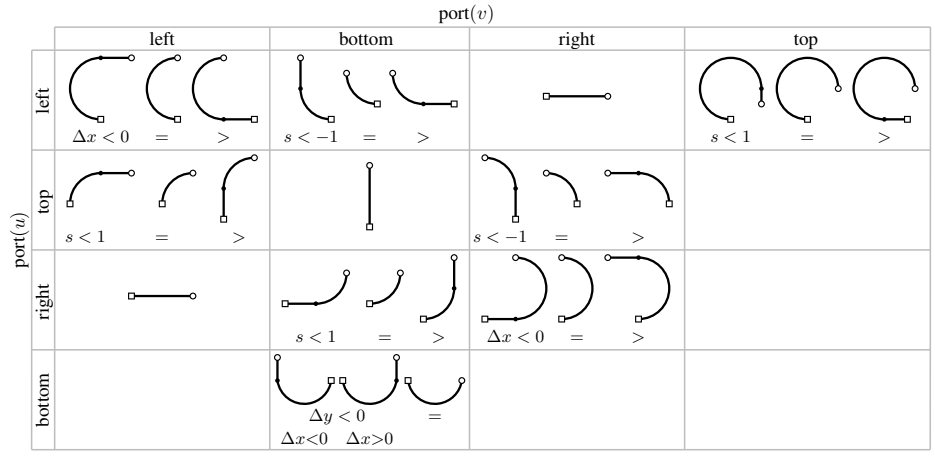
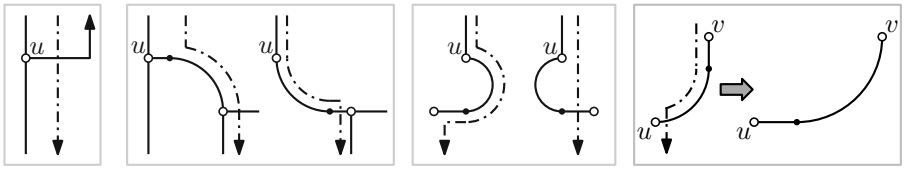


Fig. 3. Cases for drawing the edge (u, v) based on the port assignment. In each case, u is the lower of the two vertices ($y(u) < y(v)$). As shorthand, we use $\Delta x = x(u) - x(v)$, $\Delta y = y(u) - y(v)$, and $s = \text{slope}(u, v) = \Delta x/\Delta y$.

such a path in the same row, without changing their column. This essentially converts all S-shapes into horizontal edges. Now every edge (except $(1, 2)$ and $(1, n)$) is drawn as a vertical segment, horizontal segment, L-shape, or C-shape; see Fig. 2. The edge $(1, 2)$ is drawn as a U-shape and the edge $(1, n)$, if it exists, is either drawn as a C-shape or (only in the case of the octahedron) as a three-bend edge that uses the left port of vertex 1 and the top port of vertex n.

We convert the output of the algorithm of Liu et al. from OC_3 to SC_2. The coordinates of the vertices and the port assignment of their drawing define a (non-planar) SC_2-layout using the conversion table in Fig 3. In order to avoid crossings, we carefully determine new vertex positions scanning the drawing of Liu et al. from bottom to top.

We now introduce our main tool for the conversion: a *cut*, for us, is a y-monotone curve consisting of horizontal, vertical, and circular segments that divides the current drawing into a left and a right part, and only intersects horizontal segments and semicircles of the drawing. In the following, we describe how one can find such a cut from any starting point at the top of the drawing; see Fig. 4. (In spite of the fact that we define the cut going from top to bottom, "to its left" will, as usually, mean "with smaller x-coordinate".)



(a) open edge (b) L-shape (c) C-shape (d) maintaining I_2

Fig. 4. Finding a cut

When such a cut encounters a vertex u to its right with an outgoing edge associated with its left port, then the cut continues by passing through the segment incident to u. On the other hand, if the port has an incoming L-shaped or C-shaped edge, the cut just follows the edge. The case when the cut encounters a vertex to its left is handled symmetrically.

Let v be a vertex incident to two incoming C-shapes (u, v) and (w, v). If $y(w) \leq y(u)$ we call the C-shape (u, v) *protected* by (w, v); otherwise, we call it *unprotected*. In order to ensure that a cut passes only through horizontal segments and that our final drawing is planar, our algorithm will maintain the following new invariants:

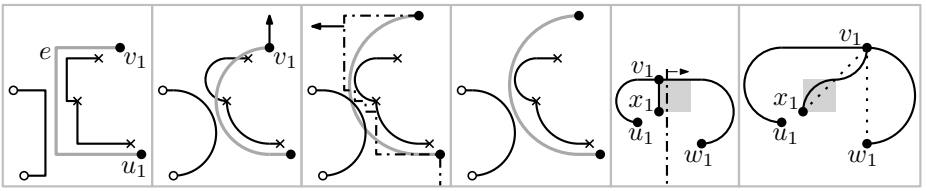
- (I_2) An L-shape never contains a vertical segment (as in Fig. 2d right); it always contains a horizontal segment (as in Fig. 2d left) or a single quarter-circle.
- (I_3) An unprotected C-shape never contains a horizontal segment incident to its top vertex (as in Fig. 2e right); it always contains a horizontal segment incident to its bottom vertex (as in Fig. 2e left) or no straight-line segment.
- (I_4) The subgraph induced by the vertices that have already been drawn has the same embedding as in the drawing of Liu et al.

Below, we treat L- and C-shapes of complexity 1 as if they had a horizontal segment of length 0 incident to their bottom vertex. Note that we always cut around protected C-shapes, so we will never end up in their interior. Now we are ready to state the main theorem of this section by presenting our algorithm for SC_2-layouts.

Theorem 1. *Every biconnected 4-planar graph admits an SC_2-layout.*

Proof. In the drawing Γ of Liu et al., vertices are arranged in rows. Let $V_1, \ldots, V_r$ be the partition of the vertex set V in rows $1, \ldots, r$. Following Liu et al., the vertices in each such set induce a path in G. We place vertices in the order $V_1, \ldots, V_r$. In this process, we maintain a planar drawing Γ' and the invariants I_1 to I_4. As Liu et al., we place the vertices on the integer grid. We deal with the special edges $(1, 2)$ and $(1, n)$ at the end, leaving their ports, that is, the bottom and left port of vertex 1 and the top port of vertex n, open.

For invariant I_1, we associate each open edge with the column on which the algorithm of Liu et al. places it. If their algorithm draws the first segment of the open edge horizontally (from the source vertex to the column), we use the same segment for our drawing. We use the same ports for the edges as their algorithm. Thus, our drawing keeps the embedding of Liu et al., maintaining invariant I_4.



(a) OC_3-layout (b) move v_1 up (c) find a cut (d) SC_2-layout (e) & (f) protected C-shape

Fig. 5. Handling C-shapes

Assume that we have placed $V_1, \ldots, V_{i-1}$ and that the vertices in V_i are $v_1, \ldots, v_c$ in left-to-right order (the case $v_1 = v_c$ is possible; this is the only case in which a vertex can have incoming L- or C-shapes at both its left and right port). Vertex v_j ($1 \leq j \leq c$) is placed in the column with which the edge entering the bottom port of v_j is associated. If the left port of v_1 is used by an incoming L- or C-shape $e = (u_1, v_1)$, we place v_1 (and the other vertices in V_i) on a row high enough so that a smooth drawing of e does not create any crossings with edges lying on the right side of e in Γ; see Fig. 5b.

In order to make sure that the new drawing of e does not create crossings with edges on the left side of e in Γ, we need to "push" those edges to the left of e. We do this by computing a cut that starts from v_1, separates the vertices and edges that lie on the left side of e in Γ from those on the right side, passes u_1 slightly to the left, and continues downwards as described above; see Fig. 5c. Since, by invariant I_4, our drawing so far is planar and each edge is drawn in a y-monotone fashion, we can find a cut, too, that is y-monotone. We move everything on the left side of the cut further left such that e has no more crossings. Note that the cut intersects only horizontal edge segments. These will simply become longer by the move.

Let $\Delta x_i = x(v_i) - x(u_i)$ and $\Delta y_i = y(v_i) - y(u_i)$ for $i = 1, \ldots, c$. It is possible that the drawing of e violates invariant I_3—if u_1 lies to the left of v_1. We consider two cases. First, assume that the edge (u_1, v_1) is the only incoming C-shape at v_1. Note that if $c > 1$ this is always the case. In this case, we simply define a cut that starts slightly to the right of v_1, follows e, intersects e slightly to the left of u_1, and continues downwards. Then we move everything on the left side of the cut by $\Delta x_1 + 1$ units to the left. Next, assume that $c = 1$ and there is another C-shape (w_1, v_1) entering the right port of v_1; see Fig. 5e. We assume w.l.o.g. that $y(w_1) \leq y(u_1)$. Let (x_1, v_1) be the edge incident to the bottom port of v_1. In this case, we first find a cut that starts slightly to the right of v_1, follows (x_1, v_1), passes x_1 slightly to the right, and continues downwards. Then we move everything on the right side of the cut by $y(v_1) - y(x_1)$ units to the right. Thus, there is an empty square to the right of (x_1, v_1) of side length $y(v_1) - y(x_1)$. Now we place v_1 at the intersection of the slope-1 diagonal through x_1 and the vertical line through w_1. Due to this placement, we can draw (x_1, v_1) using two quarter-circles with a common horizontal tangent in the top right corner of the empty square; see Fig. 5f. Note that the edge (u_1, v_1) is protected by (w_1, v_1), so it can have a horizontal segment incident to v_1. This establishes I_3.

It is also possible that the drawing of e violates invariant I_2—if slope$(u_1, v_1) > 1$. In this case we define a cut that starts slightly to the left of v_1, intersects e and continues

downwards. Then we move everything on the left side of the cut by Δy_1 units to the left. This establishes I_2.

We treat v_c, the rightmost vertex in the current row, symmetrically to v_1.

For the case that v_1 does not have incoming C-shapes at both its left and right port, we still have to treat the edges entering vertices $v_1, \ldots, v_c$ from below. Note that these edges can only be vertical or L-shaped. Vertical edges can be drawn without violating the invariants. However, invariant I_2 may be violated if an edge $e_i = (u_i, v_i)$ entering the bottom port of vertex v_i is L-shaped; see Fig. 4d. Assume that $x(u_i) < x(v_i)$. In this case we find a cut that starts slightly to the left of v_i, follows e_i, intersects e_i slightly to the right of u_i, and continues downwards. Then we move everything on the left side of the cut by Δy_i units to the left. This establishes I_2. We handle the case $x(u_i) > x(v_i)$ symmetrically.

We thus place the vertices row by row and draw the incoming edges for the newly placed vertices, copying the embedding of the current subgraph from Γ. This completes the drawing of $G - \{(1, 2), (1, n)\}$. Note that vertex 1 has no incoming edge and vertex 2 has only one incoming edge, that is, $(1, 2)$. Thus, the bottom port of both vertices is still unused. We draw the edge $(1, 2)$ as a U-shape. Finally, we finish the layout by drawing the edge $(1, n)$. By construction, the left port of vertex 1 is still unused. Note that vertex n has no outgoing edges, so the top port of n is still free. Hence, we can draw the edge $(1, n)$ as a horizontal or vertical segment followed by a three-quarter-circle. To avoid crossings, we may have to move vertex n upwards. This way, we will get a horizontal segment at vertex 1, and the three-quarter-circle will completely lie outside of the rest of the drawing. This completes the proof of Theorem 1. □

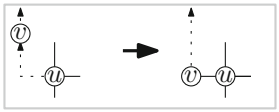
3 Smooth Layouts for Arbitrary 4-Planar Graphs

In this section, we describe how to create SC_2-layouts for arbitrary 4-planar graphs. To achieve this, we decompose the graph into biconnected components, embed them separately and then connect them. For the connection it is important that one of the connector vertices lies on the outer face of its component. Within each component, the connector vertices have degree at most 3; if they have degree 2, we must make sure that their incident edges don't use opposite ports. Following Biedl and Kant [3], we say that a degree-2 vertex v is drawn *with right angle* if the edges incident to v use two neighboring ports.

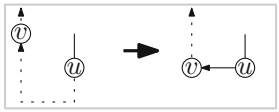
Lemma 1. *Any biconnected 4-planar graph admits an SC_2-layout such that all degree-2 vertices are drawn with right angle.*

Proof. Let v be a degree-2 vertex. We now show how to adjust the algorithm of Section 2 such that v is drawn with right angle. By construction, the top and the bottom ports of v are used. Let (u, v) be the edge entering v from below (we allow $v = 1$ and $u = 2$). We modify the algorithm such that (u, v) uses the left or right rather than the bottom port of v. We consider three cases; (u, v) is either L-shaped, U-shaped, or vertical. These cases are handled when v is inserted into the smooth orthogonal drawing.

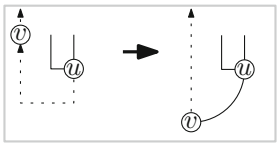
First, we assume that (u, v) is L-shaped; see Fig. 6a. Then, we can simply move v to the same row as u, making the edge horizontal.



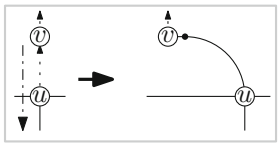
(a) an L-shape becomes a horizontal edge



(b) a U-shape becomes a horizontal edge



(c) a U-shape becomes an L-shape



(d) a vertical edge becomes an L-shape

Fig. 6. Modification of the placement of degree-2 vertices

Now, we assume that (u, v) is U-shaped; see Fig. 6b, 6c. Then $u = 1$ and $v = 2$ or vice versa. If both have degree 2, we move the higher vertex to the row of the lower vertex (if necessary) and replace the U-shaped edge by a horizontal edge. Otherwise we move the vertex with degree-2, say v, downwards to row $y(u) - \Delta x$ such that we can replace the U-shape by an L-shape.

Otherwise, (u, v) is vertical; see Fig. 6d. Then, we compute a cut that starts slightly below v, follows (u, v) downwards, passing u to its left. We move all vertices (including u, but not v) that lie on the right side of this cut (by at least Δy) to the right. Then we can draw (u, v) as an L-shape that uses the right port of v.

Observe that, in each of the three cases, we redraw all affected edges with SC_2. Hence, the modified algorithm still yields an SC_2-layout. At the same time, all degree-2 vertices are drawn with right angle as desired. □

Now we describe how to connect the biconnected components. Recall that a *bridge* is an edge whose removal disconnects a graph G. We call the two endpoints of a bridge *bridge heads*. A *cut vertex* is a vertex whose removal disconnects the graph, but is not a bridge head.

Theorem 2. *Any 4-planar graph admits an SC_2-layout.*

Proof. Let G_0 be some biconnected component of G, and let $v_1, \ldots, v_k$ be the cut vertices and bridge heads of G in G_0. For $i = 1, \ldots, k$, if v_i is a bridge head, let v_i' be the other head of the bridge, otherwise let $v_i' = v_i$. Let G_i be the subgraph of G containing v_i' and the connected components of $G - v_i'$ not containing G_0. Following Lemma 1, G_0 can be drawn such that all degree-2 vertices are drawn with right angles.

The algorithm of Section 2 that we modified in the proof of Lemma 1 places the last vertex (n) at the top of the drawing and thus on the outer face. When drawing G_i, we choose v_i' as this vertex. By induction, G_i can be drawn such that all degree-2 vertices are drawn with right angles.

In order to connect G_i to G_0, we make G_0 large enough to fit G_i into the face that contains the free ports of v_i. We may have to rotate G_i by a multiple of $90°$ to achieve

the following. If v_i is a cut vertex, we make sure that v'_i uses the ports of v_i that are free in G_0. Then we identify v_i and v'_i. Otherwise we make sure that a free port of v_i and a free port of v'_i are opposite. Then we draw the bridge (v_i, v'_i) horizontally or vertically. This completes our proof.

For an example run of our algorithm, see the full version [1]. For graphs of maximum degree 3, we can make our drawings more compact. This is due to the fact that we can avoid C-shaped edges (and hence cuts) completely. In the presence of L-shapes only, it suffices to stretch the orthogonal drawing by a factor of n.

Theorem 3. *Every biconnected 3-planar graph with n vertices admits an SC_2-layout using area $\lfloor n^2/4 \rfloor \times \lfloor n/2 \rfloor$.*

Proof. It is known that every biconnected 3-planar graph except K_4 has an OC_2-layout using area $\lfloor n/2 \rfloor \times \lfloor n/2 \rfloor$ from Kant [7]. Now we use the same global stretching as Bekos et al. when they showed that every OC_2-layout can be transformed into an SC_2-layout [2, Thm. 2]: we stretch the drawing horizontally by the height of the drawing, that is, by a factor of $\lfloor n/2 \rfloor$. This makes sure that we can replace every bend by a quarter circle without introducing crossings. Figure 7 shows an SC_1-layout of K_4; completing our proof. □

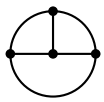


Fig. 7. SC_1-layout of K_4

4 SC_1-Layouts of Biconnected 4-Outerplane Graphs

In this section, we consider *4-outerplane* graphs, that is, 4-outerplanar graphs with an outerplanar embedding. We prove that any biconnected 4-outerplane graph admits an SC_1-layout. To do so, we first prove the result for a subclass of 4-outerplane graphs, which we call $(2, 3)$-restricted outerplane graphs; then we generalize. Recall that the *weak dual* of a plane graph is the subgraph of the dual graph whose vertices correspond to the bounded faces of the primal graph.

Definition 1. *A 4-outerplane graph is called* $(2, 3)$-restricted *if it contains a pair of consecutive vertices on the outer face, x and y, such that* $\deg(x) = 2$ *and* $\deg(y) \leq 3$.

Lemma 2. *Any biconnected $(2, 3)$-restricted 4-outerplane graph admits an SC_1-layout.*

Proof. Let x and y be two vertices, consecutive on the outer face of the given graph G such that $\deg(x) = 2$ and $\deg(y) \leq 3$. Let also T be the weak dual tree of G rooted at the node, say v^*, of T corresponding to the bounded face, say f^*, containing both x and y. We construct the SC_1-layout Γ for G by traversing T, starting with v^*. When we traverse a node of T, we draw the corresponding face of G with SC_1.

Consider the case when we have constructed a drawing $\Gamma(H)$ for a connected subgraph H of G and we want to add a new face f to $\Gamma(H)$. For each vertex u of H, let $p_u = (x(u), y(u))$ denote the point at which u is drawn in $\Gamma(H)$. The *remaining degree* of u is the number of vertices adjacent to u in $G - H$. Since we construct $\Gamma(H)$ face by face, the remaining degree of each vertex in H is at most two. The *free* ports of u are the ones that are not occupied by an edge of H in $\Gamma(H)$. During the construction of Γ, we maintain the following four invariants:

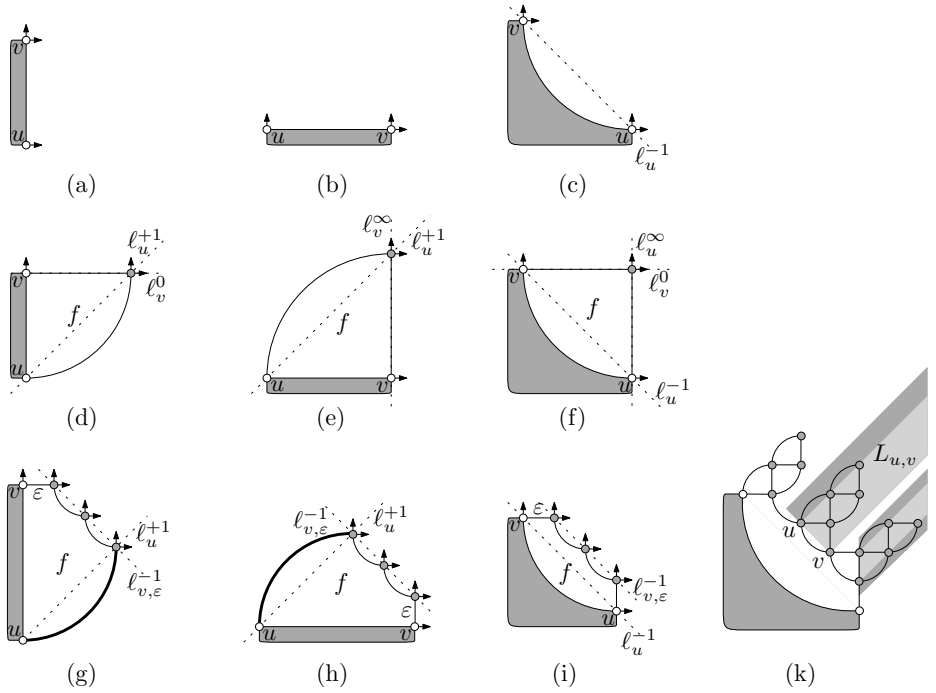


Fig. 8. (a)-(i) Different cases that arise when drawing face f of G. (k) A sample drawing.

- (J_1) $\Gamma(H)$ is an SC_1-layout that preserves the planar embedding of G, and each edge is drawn either as an axis-parallel line segment or as a quarter-circle in $\Gamma(H)$. (Note that we do not use semi- and 3/4-circles.)
- (J_2) For each vertex u of H, the free ports of u in $\Gamma(H)$ are consecutive around u, and they point to the outer face of $\Gamma(H)$.
- (J_3) Vertices with remaining degree exactly 2 are incident to an edge drawn as a quarter-circle.
- (J_4) If an edge (u, v) is drawn as an axis-parallel segment, then at least one of u and v has remaining degree at most 1. If (u, v) is vertical and $y(u) < y(v)$, then u has remaining degree at most 1 and the free port of u in $\Gamma(H)$ is horizontal; see Figs. 8a, 8d and 8g. Symmetrically, if (u, v) is horizontal and $x(u) < x(v)$, then u has remaining degree at most 1 and the free port of u in $\Gamma(H)$ is vertical; see Figs. 8b, 8e and 8h.

We now show how we add the drawing of the new face f to $\Gamma(H)$. Since G is biconnected and outerplanar, and due to the order in which we process the faces of G, f has exactly two vertices, say u and v, which have already been drawn (as p_u and p_v). The two vertices are adjacent. Depending on how the edge (u, v) is drawn in $\Gamma(H)$, we draw the remaining vertices and edges of f.

Let $k \geq 3$ be the number of vertices on the boundary of f. The slopes of the line segment $\overline{p_u p_v}$ is in $\{-1, 0, +1, \infty\}$, where ∞ means that $\overline{p_u p_v}$ is vertical. For $s \in \{-1, 0, +1, \infty\}$, we denote by ℓ_u^s the line with slope s through p_u. Similarly, we denote

by $\ell^s_{u,\varepsilon}$ the line with slope s through the point $(x(u) + \varepsilon, y(u))$, for some $\varepsilon > 0$. Figs. 8d–8f show the drawing of f for $k = 3$, and Figs. 8g–8i for any $k \geq 4$.

Note that the lengths of the line segments and the radii of the quarter-circles that form f are equal (except for the radii of the bold-drawn quarter-circles of Figs. 8g and 8h which are determined by the remaining edges of f). Hence, the lengths of the line segments and the radii of the quarter-circles that form any face that is descendant of face f in T are smaller than or equal to the lengths of the line segments and the radii of the quarter-circles that form f. Our construction ensures that all vertices of the subgraph of G induced by the subtree of T rooted at f lie in the interior or on the boundary of the diagonal semi-strip L_{uv} delimited by ℓ_u^{+1}, ℓ_v^{+1}, and $\overline{p_u p_v}$ (see Fig. 8k). The only edges of this subgraph that are drawn in the complement of L_{uv} (and are potentially involved in crossings) are incident to two vertices that both lie on the boundary of L_{uv}. In this particular case, however, the degree restriction implies that L_{uv} is surrounded from above and/or below by two empty diagonal semi-strips of at least half the width of semi-strip L_{uv}, which is enough to ensure planarity for two reasons. First, any face that is descendant of face f in T is formed by line segments and quarter-circles of radius that are at most as big as the corresponding ones of face f. Second, due to the degree restrictions, if two neighboring children of f are triangles, the left one cannot have a right child and vice versa.

Let us summarize. Figures 8d–8i show that the drawing of f ensures that invariants (J_1)–(J_4) of our algorithm are satisfied for $H \cup \{f\}$. We begin by drawing the root face f^*. Since G is $(2,3)$-restricted, f^* has two vertices x and y consecutive on the outer face with $\deg(x) = 2$ and $\deg(y) \leq 3$. We draw edge (x, y) as a vertical line segment. Then the remaining degrees of x and y are 1 and 2, respectively, which satisfies the invariants for face f^*. Hence, we complete the drawing of f^* as in Fig. 8d or 8g. Traversing T in pre-order, we complete the drawing of G. □

Next, we show how to deal with general biconnected 4-outerplane graphs. Suppose G is not $(2,3)$-restricted. As the following lemma asserts, we can always construct a biconnected $(2,3)$-restricted 4-outerplane graph by deleting a vertex of degree 2 from G.

Lemma 3. *Let $G = (V, E)$ be a biconnected 4-outerplane graph that is not $(2,3)$-restricted. Then G has a degree-2 vertex whose removal yields a $(2,3)$-restricted biconnected 4-outerplane graph.*

Proof. The proof is by induction on the number of vertices. The base case is a maximal biconnected outerplane graph on six vertices, which is the only non-$(2,3)$-restricted graph with six or less vertices. It is easy to see that in this case the removal of any degree-2 vertex yields a biconnected $(2,3)$-restricted 4-outerplane graph. Now assume that the hypothesis holds for any biconnected 4-outerplane graph with $k \geq 6$ vertices. Let G_{k+1} be a biconnected 4-outerplane graph on $k+1$ vertices, which is not $(2,3)$-restricted. Let $\mathcal{F}$ be a face of G_{k+1} that is a leaf in its weak dual. Then $\mathcal{F}$ contains only one internal edge and exactly two external edges since, if it contained more than two external edges, G_{k+1} would be $(2,3)$-restricted. Therefore, $\mathcal{F}$ consists of three vertices, say a, b and c, consecutive on the outer face and $\deg(a) = \deg(c) = 4$, since otherwise G_{k+1} would be $(2,3)$-restricted. By removing b, we obtain a new graph, say G_k, on k vertices. If a or c is incident to a degree-2 vertex in G_k, then G_k is $(2,3)$-restricted.

Otherwise, by our induction hypothesis, G_k has a degree-2 vertex whose removal yields a $(2,3)$-restricted outerplanar graph. Since this vertex is neither adjacent to a nor c, the removal of this vertex makes G_{k+1}, too, $(2,3)$-restricted. □

Theorem 4. *Any biconnected 4-outerplane graph admits an SC_1-layout.*

Proof. If the given graph G is $(2,3)$-restricted, then the result follows from Lemma 2. Thus, assume that G is not $(2,3)$-restricted. Then, G contains a degree-2 vertex, say b, whose removal yields a biconnected $(2,3)$-restricted 4-outerplane graph, say G'. Hence, we can apply the algorithm of Lemma 2 to G' and obtain an outerplanar SC_1-layout $\Gamma(G')$ of G'. Since this algorithm always maintains consecutive free ports for each vertex and the neighbors of b are on the outer face of $\Gamma(G')$, we insert insert b and its two incident edges to obtain an SC_1-layout $\Gamma(G)$ of G as follows. Let a and c be the neighbors of b and assume w.l.o.g. that c is drawn above a. If edge (a,c) is drawn as a quarter-circle, then a 3/4-circle arc from p_c to p_b and a quarter-circle from p_b to p_a suffice. Otherwise, line segment $\overline{p_a p_b}$ and a quarter-circle from p_b to p_c do the job. □

References

1. Alam, M.J., Bekos, M.A., Kaufmann, M., Kindermann, P., Kobourov, S.G., Wolff, A.: Smooth orthogonal drawings of planar graphs. Arxiv report arxiv.org/abs/1312.3538 (2013)
2. Bekos, M.A., Kaufmann, M., Kobourov, S.G., Symvonis, A.: Smooth orthogonal layouts. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 150–161. Springer, Heidelberg (2013)
3. Biedl, T., Kant, G.: A better heuristic for orthogonal graph drawings. Comput. Geom. Theory Appl. 9(3), 159–180 (1998)
4. Duncan, C.A., Eppstein, D., Goodrich, M.T., Kobourov, S.G., Nöllenburg, M.: Lombardi drawings of graphs. J. Graph Algorithms Appl. 16(1), 85–108 (2012), http://dx.doi.org/10.7155/jgaa.00251
5. Duncan, C.A., Eppstein, D., Goodrich, M.T., Kobourov, S.G., Nöllenburg, M.: Drawing trees with perfect angular resolution and polynomial area. Discrete Comput. Geom. 49(2), 157–182 (2013)
6. Garg, A., Tamassia, R.: On the computational complexity of upward and rectilinear planarity testing. SIAM J. Comput. 31(2), 601–625 (2001)
7. Kant, G.: Drawing planar graphs using the canonical ordering. Algorithmica 16(1), 4–32 (1996)
8. Leiserson, C.E.: Area-efficient graph layouts (for VLSI). In: Proc. 21st Annu. IEEE Symp. Foundat. Comput. Sci. (FOCS 1980), pp. 270–281 (1980)
9. Liu, Y., Morgana, A., Simeone, B.: A linear algorithm for 2-bend embeddings of planar graphs in the two-dimensional grid. Discrete Appl. Math. 81(1-3), 69–91 (1998)
10. Seemann, J.: Extending the Sugiyama algorithm for drawing UML class diagrams: Towards automatic layout of object-oriented software diagrams. In: Di Battista, G. (ed.) GD 1997. LNCS, vol. 1353, pp. 415–424. Springer, Heidelberg (1997)
11. Tamassia, R.: On embedding a graph in the grid with the minimum number of bends. SIAM J. Comput. 16(3), 421–444 (1987)
12. Waldherr, H.: Network Gmunden–Vöcklabruck–Salzkammergut of the Publ. Transp. Assoc. OÖVG, Austria (November 2012), http://www.ooevv.at/uploads/media/OOE2_Salzkammergut_V17_END.pdf (accessed September 10, 2013)