Smooth Orthogonal Layouts

M. A. Bekos[1], M. Kaufmann[1], S. G. Kobourov[2], A. Symvonis[3]

[1] Institute for Informatics, University of Tübingen, Germany
{bekos,mk}@informatik.uni-tuebingen.de
[2] Department of Computer Science, University of Arizona, USA
kobourov@cs.arizona.edu
[3] School of Applied Mathematical Sciences, National Technical University of Athens, Greece
symvonis@math.ntua.gr

Abstract. We study the problem of creating *smooth orthogonal layouts* for planar graphs. While in traditional orthogonal layouts every edge is made of a sequence of axis-aligned line segments, in smooth orthogonal layouts every edge is made of axis-aligned segments and circular arcs with common tangents. Our goal is to create such layouts with low edge complexity, measured by the number of line and circular arc segments. We show that every biconnected 4-planar graph has a smooth orthogonal layout with edge complexity 3. If the input graph has a complexity-2 traditional orthogonal layout we can transform it into a smooth complexity-2 layout. Using the Kandinsky model for removing the degree restriction, we show that any planar graph has a smooth complexity-2 layout.

1 Introduction

Orthogonal graph drawing has a long tradition, dating back to VLSI layouts and floorplanning applications. The input graphs are assumed to be planar and with max-degree four, also known as *4-planar graphs*. The goal is to produce a drawing in which each vertex is a point on the integer grid and each edge is represented by a sequence of horizontal and vertical line segments, while optimizing various features of the layout Typical desirable features include minimizing the used area and minimizing the total number of bends (or the maximum number of bends per edge) [1]. Finding an embedding with the minimum number of bends is an NP-hard problem; moreover, minimizing the total number of bends might lead to some edges with many bends. The readability of poly-line drawings decreases as the number of bends increases and the bend angles decrease. One explanation is that every bend interrupts the eye movement and requires a change of direction, with the effect depending on the magnitude of the bend angle.

It is likely that replacing poly-line edges with smooth curves (e.g., composed of two or more circular arcs with common tangents) will result in drawings with improved readability and more aesthetic appeal; see Fig. 1. A natural notion of *edge complexity* for such curves is the combinatorial complexity, given by the number of arcs forming an edge, and a natural optimization goal is to minimize the edge complexity.

1.1 Motivation

Recent work suggests attractive alternatives that address readability related issues posed by the presence of bends in polyline drawings. Such work is motivated by perception research, indicating that representing paths with smooth geodesic trajectories aids in

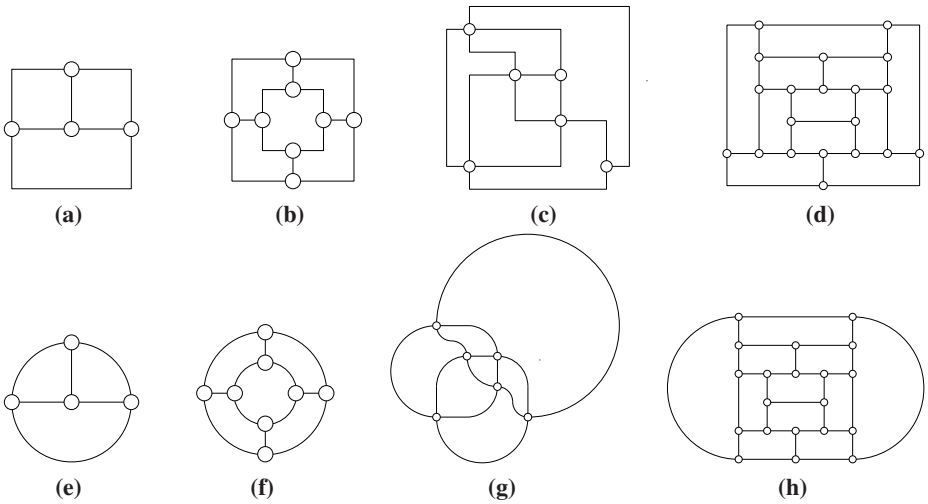

Fig. 1: All Platonic solids with degree 3 or 4 drawn in traditional orthogonal style with the minimum number of bends per edge and redrawn in the smooth orthogonal style with better edge complexity. Figs. 1d and 1h resemble the same orthogonal layout.

comprehension [16], as well as by the aesthetic appeal of drawings with smooth curves such as those of American abstract artist Mark Lombardi [20]. Two features that stand out in Lombardi's work are the use of circular-arc edges and their even distribution around vertices; see Fig. 2. Such even spacing of edges around each vertex (also known as *perfect angular resolution*), together with the use of circular arcs for edges, formally define *Lombardi drawings* of graphs [9].

Not all graphs allow for Lombardi realizations and the characterization of Lombardi graphs is an open problem. One way to visualize non-Lombardi graphs in a Lombardi fashion is to relax the circular-arc constraint; while vertices still have perfect angular resolution, the edges can be represented as smooth sequences of circular arcs. For example, Duncan *et al.* [8] describe k-Lombardi drawings, where each edge is a smooth sequence of k circular arcs.

Note that vertices of degree four have perfect angular resolution in traditional orthogonal graph layouts, by virtue of construction, and vertices of lower degrees have angular resolution within a factor of two of optimal. In this paper, we study the problem of creating *smooth orthogonal layouts*, where we use circular arcs to create smoother curves for the edges. In order to obtain smooth curves, we ensure that each edge is composed of rectilinear line segments and circular arcs with common tangents.

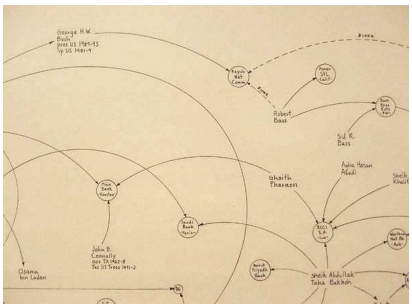

Fig. 2: Part of a Mark Lombardi drawing.

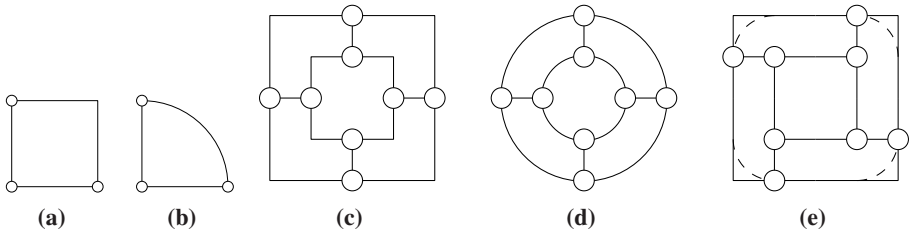

Fig. 3: (a) An orthogonal layout of the triangle graph with edge complexity two. (b) A smooth layout of the triangle graph with edge complexity one. (c) An orthogonal layout of the cube graph with edge complexity two. (d) A smooth layout of the cube graph with edge complexity one. (e) An orthogonal layout of the cube which cannot be improved w.r.t. edge complexity.

Our general approach is based on modifying a given traditional orthogonal layout by moving the vertices as needed, and replacing each bend by a smooth circular arc of appropriate radius, without introducing edge-crossings. We show that in many settings this can be accomplished without increasing edge complexity (number of segments and circular arcs needed to represent a given edge). As a result, we obtain layouts with improved readability, where it is easier to follow non straight-line edges, which are represented by smooth curves.

The use of circular arcs not only can improve readability, but also can reduce edge complexity. It is easy to see that any traditional orthogonal layout of K_3 has complexity at least two; see Fig. 3a. Allowing circular arcs reduces the complexity to one; see Fig. 3b. Similarly, the complexity-2 layout of the cube graph can be transformed into a smooth complexity-1 layout; see Fig. 3c-3d. However, as a different layout of the cube graph demonstrates, we cannot always obtain a smooth layout of complexity one by simply replacing the segments adjacent to a bend by a circular arc; see Fig. 3e.

1.2 Related Work

Early work on orthogonal layouts was done by Valiant [23] and Leiserson [19] in the context of VLSI design. Tamassia [21], Tamassia and Tollis [22], and Biedl and Kant [2] continued this line of research in the context of graph drawing. The common objectives have been the minimization of the used area, total edge length, total number of bends, and maximum number of bends per edge. By default it was often assumed that input graphs were restricted to degree-4 planar graphs. Models incorporating higher degree graphs were introduced later by Tamassia [21] and Foessmeier [12].

In addition to the work on Lombardi drawings, there has been other work on graph drawing with circular-arc or curvilinear edges for the sake of achieving good angular resolution [7, 14]. There is also significant work on *confluent drawings* [10, 15], where curvilinear edges are used to bundle similar edges together and avoid edge crossings. In confluent drawings, edges are drawn like train-tracks using locally-monotone curves which do not self-intersect and which do not have sharp turns. The curves may have overlapping portions, but no crossings.

Brandes and Wagner [5] provide a force-directed method for visualizing train schedules using Bézier curves for edges and fixed positions for vertices. Finkel and Tamassia [11] extend this work with a force-directed method for drawing graphs with curvilinear edges where vertex positions are not fixed. For fixed position drawings with cubic Bézier curves, Brandes and Schlieper [4] use force-directed methods to maximize angular resolution.

1.3 Our Contributions

We are particularly interested in providing theoretical guarantees about creating smooth orthogonal layouts, while not increasing edge complexity and not introducing edge crossings. In particular, we show that every biconnected 4-planar graph has a smooth orthogonal layout with edge complexity 3. We also show that any triconnected 3-planar graph has a smooth orthogonal layout with edge complexity 1. If the input graph has a complexity-2 traditional orthogonal layout we can transform it into a smooth complexity-2 layout. Using the Kandinsky model for removing the degree restriction, we show that any planar graph has a smooth complexity-2 layout. We also study the problem of smooth orthogonal layouts with lower complexity. That is, in some cases it is possible to reduce the complexity of a traditional orthogonal layout when using circular arcs; see Fig. 3. Here we first show that not all 4-planar graphs have smooth complexity-2 layouts. We then show that reduction in complexity may result in serious drawing area penalty. Specifically, we show that there exist graphs whose smooth complexity-1 layout requires exponential drawing area.

2 Smooth Fixed Orthogonal Layouts

The most restrictive version of our approach is the one where the layout of the graph is given and the placement of the vertices cannot be changed. In this setting we are only allowed to replace the bends of edges by circular arcs, such that adjacent segments have the same horizontal or vertical tangent at their contact points. An ad hoc approach would be to replace each bend by a very small circular arc segment, which would increase the number of segments on the edge from $k > 0$ to $2k-1$. Such increase in edge complexity might be unavoidable in the fixed layout setting; see Fig. 4.

In practice, it might be possible to avoid increasing the edge complexity. One way to achieve this in the fixed layout setting is to try to increase the radii of the circular

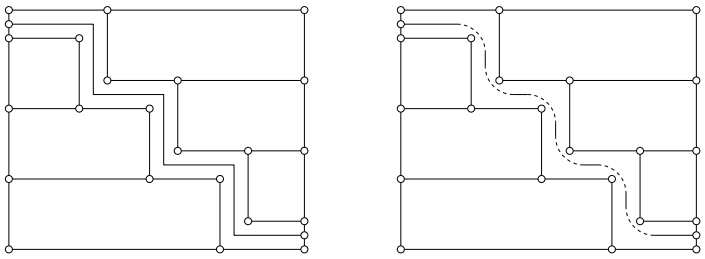

Fig. 4: Edge complexity might increase from k to $2k - 1$ for "staircase" edges.

Fig. 5: Illustration of horizontal stretching.

arcs until one of their adjacent straight-line segments disappear or the circular segment hits another vertex. In fact, it is easy to minimize the number of segments in the fixed placement scenario, if all circular arcs have the same radius, as the next lemma shows.

Lemma 1 *Given a fixed orthogonal layout, there exists an $O(N \log N)$-algorithm that maximizes the radii of the circular arcs in the drawing, where N is the total number of vertices and bends of the orthogonal layout.*

Proof: The algorithm determines whether a smooth layout of an input radius is feasible (i.e., leads to a crossing-free solution). Since the vertices have fixed positions, the corresponding decision problem can be answered in $O(N \log N)$ time by a plane-sweep method. This is enough to apply the randomized optimization technique of Chan to solve the problem in $O(N \log N)$ time [6]. □

3 Smooth Postprocessing: The Fixed-Topology Model

In this section, we assume that a layout is given, but now we are allowed to change the length of the segments of the edges, so long as no segments become zero length. Specifically, in this setting the vertex topology is fixed (i.e., if an edge connects to a vertex using the north port, then it must continue to use the north port) and the edge topology is fixed (i.e., the sequence of directional changes of an edge cannot be modified). This restriction is referred to as the *preserved orthogonal representation* in [21]. Here, we call it the *fixed-topology model*. Even though this model is also very restrictive, it provides us with enough flexibility to produce smooth layouts with low edge complexity.

3.1 Layout Stretching

We begin with a simple problem, that of postprocessing a traditional orthogonal layout in which all edges have complexity 2, so as to obtain a smooth layout of the same complexity. Here we use circular arcs of varying sizes to replace straight-line segments (unlike in Lemma 1, where we used circular arcs of the same size).

Theorem 2. *Any complexity-2 orthogonal layout can be transformed into a smooth complexity-2 layout in linear time.*

Proof: Let l be the length of the longest vertical segment in any complexity-2 edge of the input layout. Consider the vertex v which is one of the endpoints of that vertical segment of length l. Stretch the entire drawing horizontally by a factor of l; see Fig. 5. Then in the stretched drawing, the vertical segment is no larger than its matching horizontal segment. Also due to the stretching, we can easily verify that the grid of size $l \times l$ in each quadrant of vertex v is empty of other vertices and edges. Then we can safely replace the vertical segment with a quarter-circle arc, which yields a smooth complexity-2 realization of the edge.

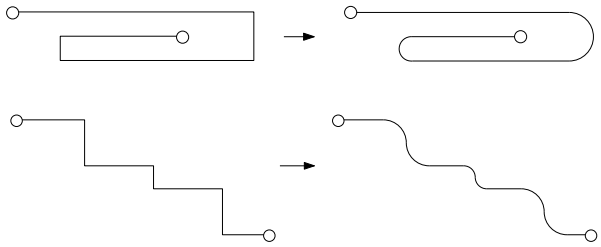

Fig. 6: Illustration of horizontal stretching for edges with many bends: uniform turns do not increase complexity while alternating turns do.

Note that the same argument can be applied to any complexity-2 edge in the original layout. That is, for any such edge the vertical segment is no larger than the horizontal segment, and there is an empty square grid in each quadrant around the vertex, allowing us to replace the vertical segment with a circular arc.

It is easy to verify that once this procedure has been used to modify all complexity-2 edges, the result is a smooth complexity-2 orthogonal layout on a grid that is a factor of n larger than the input layout. The safe insertion of circular arcs in place of straight-line segments ensures that if we started without crossings, we also finish without crossings. Finally, the transformation can be accomplished in linear time using one plane sweep to stretch the drawing and another one to introduce circular arcs of appropriate sizes. □

Corollary 3 *Any complexity-3 orthogonal layout can be transformed into a smooth complexity-4 orthogonal layout.*

Note that we can also apply the stretching technique to any edge with complexity $k > 1$; see Fig. 6. For edges that turn only in the same direction (i.e., right, right, right) the edge complexity does not increase. However, for edges that turn in alternating directions (i.e.,staircase edges), the edge complexity increases from k to $3/2k$, since each vertical segment is replaced by an S-shape pattern with complexity 2.

Corollary 4 *Stretching increases edge complexity by no more than a 1.5 factor.*

3.2 Area Bounds for Smooth Orthogonal Layouts

Smooth orthogonal layouts have aesthetic appeal and improve the readability of the underlying graph, when compared to traditional orthogonal layouts. However, our methods for creating such layouts might result in increased drawing area. Traditional orthogonal layout algorithms place the vertices on an integer grid of size $O(n) \times O(n)$. When the stretching technique described above is applied to an orthogonal layout with complexity $k > 0$, the result is a smooth layout with increased layout area from $O(n^2)$ to $O(n^3)$.

The situation is different if we want to generate smooth layouts with complexity exactly one (i.e., every edge is either one straight-line segment or one circular arc). In particular, for smooth complexity-1 layout, the area penalty can be exponential, as shown in the next theorem.

Theorem 5. *There exists a graph whose smooth complexity-1 layout requires exponential area, in the fixed topology setting.*

Sketch of proof. The graph we use is shown in Fig. 7. If every edge is to be drawn with complexity one, we must use quarter-circle arcs, as shown in the figure. Let a_i be the length of the i-th curve from the origin. Then, it is not difficult to prove that $a_n = O(\sqrt{2}^{\frac{n}{2}})$. □

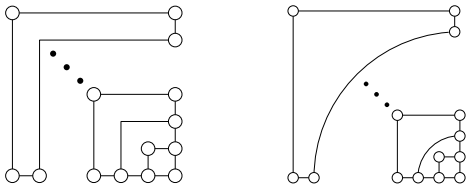

Fig. 7: Illustration of exponential area blow-up for complexity-1 drawings.

4 Smooth Layouts with Low Complexity

While many 4-planar graphs indeed have complexity-2 orthogonal layouts, and can hence be transformed into smooth complexity-2 layouts, this is not true for all graphs. We next show that all biconnected 4-planar graphs (including the octahedron) admit smooth complexity-3 layouts. This is next to optimal, as we also show that for smooth layouts, complexity 2 is necessary.

Theorem 6. *Any biconnected* 4*-planar graph admits a smooth complexity-3 layout.*

Proof: Any 4-planar graph, except the octahedron, admits an orthogonal layout of edge complexity 3. This is due to an iterative constructive algorithm of Biedl and Kant [2]. In a high-level description, their algorithm for the case of 4-planar biconnected graphs, chooses a pair of vertices s and t of minimum degree and iteratively places the vertices of the input graph in the order implied by its st-numbering, maintaining the following invariant: At each iteration, every edge, for which exactly one endpoint is drawn, is associated with a column (vertical line). The first two vertices are drawn bottommost in the same horizontal row; see Fig. 8a. A vertex later in the order is drawn in the bottommost available row and occupies as many neighboring columns as necessary to maintain the invariant.

In the resulting layout, two edges are of particular importance: the bottommost U-shape edge, and the left/right-most C-shape edge. The other edges are of edge complexity either one (drawn as straight-line vertical segments), or two (made of one vertical and one horizontal segments), or three (S-shapes, made of one vertical and two horizontal segments).

We modify the algorithm using the stretching technique above, so that every edge that is created has smooth complexity 3 or less. Here the stretching used is vertical, and ensures that we can replace every horizontal segment with a circular arc of appropriate radius, without introducing crossings; see Fig. 8b. For a complexity-2 edge, the stretch

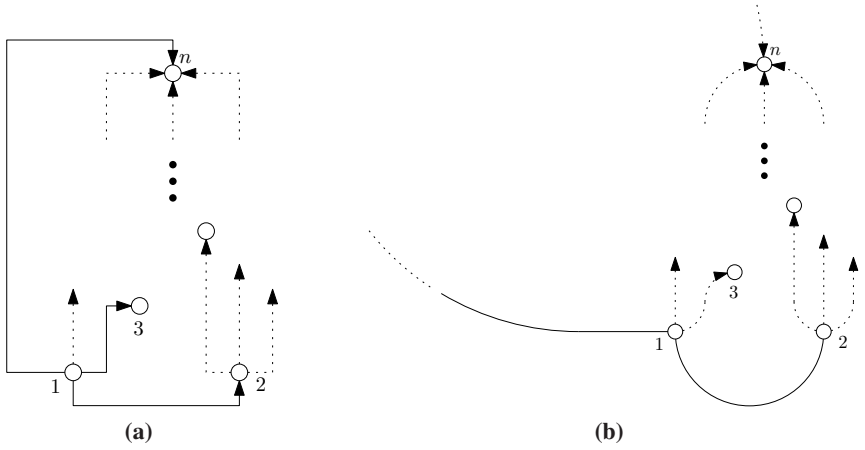

Fig. 8: Illustration of the Biedl-Kant algorithm [2] and its smooth modification.

must ensure that the horizontal segment has smaller length than the vertical segment. For a complexity-3 edge, the stretch must ensure that the sum of the lengths of the two horizontal segments is less than the length of vertical segment. This can be accomplished with a uniform vertical stretch of factor at most $2n$. The following cases based on the types of edges arise (all illustrated in the small example in Fig. 8).

A complexity-1 edge remains a straight-line segment after the uniform stretching. There are two types of complexity-2 edges. A complexity-2 edge, which initially is made of a straight-line horizontal segment followed by a vertical one, is transformed into an edge of smooth complexity two, consisting of a quarter-circle arc followed by a vertical straight-line segment. A complexity-2 edge, which initially is made of a straight-line vertical segment followed by a horizontal one, is transformed into an edge of smooth complexity two, consisting of a vertical straight-line segment followed by a quarter-circle arc.

All but two of the complexity-3 edges are S-shaped, initially made of a horizontal, vertical, horizontal segment sequence. This is transformed into an edge of smooth complexity three, consisting of a quarter-circle arc followed by a vertical straight-line segment and another quarter-circle arc; see Fig. 9.

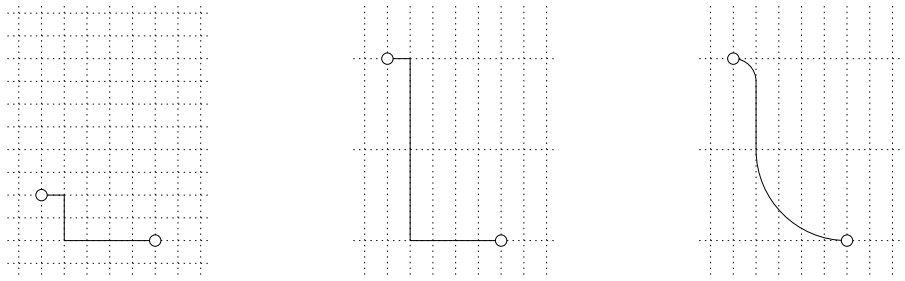

Fig. 9: Modifying the layout of a complexity-3 edge.

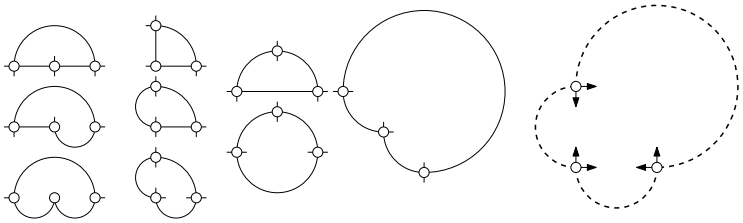

Fig. 10: All realizations of the triangle graph with edge complexity one.

The bottommost U-shape edge is transformed into an edge of smooth complexity 1 consisting of a half-circular arc. The left/right-most C-shape edge is transformed into an edge of smooth complexity 2 consisting of a horizontal straight-line segment followed by a 3/4-circular arc.

It is particularly important that we never encounter a "staircase" edge of the type that increases the complexity when its layout is transformed from orthogonal to smooth. The absence of such edges can be guaranteed by the layout construction of the Biedl-Kant algorithm [2]. Note that the algorithm of Biedl and Kant can be adjusted to support non-biconnected graphs as well. To do so, the graph is split into blocks which are embedded separately and appropriately merged. Unfortunately, we were not able to support this case, since the blocks are rotated, flipped etc. □

5 Smooth Complexity-1 Layouts

The use of circular arcs allows us to not only create smooth orthogonal layouts without increasing edge complexity, but it sometimes allows us to reduce the edge complexity. For example, we can compute smooth orthogonal layouts with reduced complexity for all 4-planar Platonic solids. The tetrahedron, cube, and dodecahedron, which require complexity-2 in traditional orthogonal layouts, all have smooth complexity-1 layouts; see Fig. 1. However, we cannot always achieve this, as shown in the next theorem.

Theorem 7. *There exists a 4-planar graph that does not admit a smooth layout with complexity one.*

Proof: Consider the octahedron graph and suppose that it has a smooth complexity-1 layout. As the graph is 4-regular and very symmetric, we can take any face as the outer-face. The outer-face is formed by three vertices of degree four, and its edges must be arranged in such a way that each vertex has two free ports pointing inside. Given these conditions it is easy to show that neither of three edges on the outer-face can be a straight-line segment or a quarter-circle arc (see all possible realizations in Fig. 10). In fact, the only way to realize the face and keep the ports inside is with two half-circle arcs and one 3/4-circle arc (see the rightmost realization in Fig. 10). Moreover, this feasible configuration is unique, up to rotation and scaling.

Note that the only feasible realization of the outer-face places the three vertices at the corners of a square (a consequence of the use of two half-circles and one 3/4-circle). Now we must place the inner three vertices of the octahedron. Consider one of the two inner vertices that is adjacent to two outer vertices that are connected by a half-circle. The inner vertex must use two consecutive ports as connections to the outer-face and

leave two free ports pointing inside. Using careful case analysis, it is not difficult to show that there is no feasible place for such a vertex. □

In traditional orthogonal setting, the octahedron is the only graph that requires complexity 4. That is, all other 4-planar graphs can be drawn with complexity 3 or less. In the smooth setting we can draw the octahedron with complexity 2, while the above theorem shows that there is no smooth complexity-1 layout. While tempting to think otherwise, it is significant to note here that the class of 4-regular planar graphs that do not admit smooth complexity-1 layouts is infinite, as the next corollary shows.

Corollary 8 *There exist infinitely many 4-planar graphs that do not admit smooth complexity-1 layouts.*

Proof: We construct a class of 4-planar graphs with a 4-regular triangular outer-face and a cycle with k vertices inside, where $k \geq 3$. The cycle has three special vertices, each of which is connected to a pair of vertices of the outer-face. Note that the case where $k = 3$ corresponds to the octahedron discussed above. In the case where $k > 3$, the three special vertices of the cycle also have degree 4 and must be connected to a pair of vertices incident to the outer-face. Each of these special vertices must use consecutive ports to connect to the outer-face and must leave two adjacent ports pointing inside available for their neighbors on the cycle. Just as in the case of the octahedron, it is impossible to place all three of the special vertices inside the outer-face under these constraints, and use only complexity-1 edges. □

Corollary 8 suggests that not all 4-planar graphs admit complexity one smooth orthogonal layouts. So, we turn our attention in the case of 3-planar graphs, for which we identify a class that admit complexity one smooth orthogonal layouts.

Theorem 9. *Any triconnected* 3*-planar graph admits a smooth complexity-1 layout.*

Sketch of proof. The proposed algorithm initially constructs an orthogonal layout (by appropriately modifying an algorithm of Kant [17]), which in turn, is transformed into a smooth orthogonal layout in the second phase of the algorithm. Due to space constraints, the detailed proof and the relevant algorithm are given in the Appendix. □

6 Smooth Orthogonal Layouts for High Degree Graphs

A serious limitation for the practical applicability of orthogonal layouts and consequently for smooth orthogonal layouts, is the vertex degree restriction. Several extensions that overcome this restriction have been proposed for orthogonal layouts [21]. The most widely used model is the *podefsnev* model [12], also known as the Kandinsky model [3]. The basic idea is to use square-shaped nodes, placed on a coarse grid, with multiple edges attached to each side of the square aligned on finer grid. Such edges conceptually end at one point, and are drawn ending next to each other; see Fig. 11a. The edges follow the *bend-or-end condition*, which means that for edge e attached at the top of v and a node w placed directly above v at the same vertical grid-line, either edge e bends to the left or right before intersecting w or it ends at the bottom side of w.

We will apply our approach for making orthogonal layouts smooth to the Kandinsky model, requiring that different edges at the same side of a node must be circular-arcs

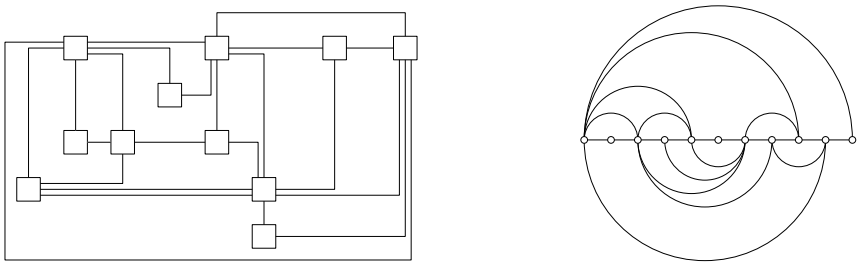

Fig. 11: (a) A *podevsef* style layout [12]; (b) A smooth Kandinsky layout of the same graph.

of different radii. As a layout algorithm suitable for this model, we use the technique of Kaufmann and Wiese [18] for point-set embedding with few bends per edge. If the graph is Hamiltonian, then we can distribute the vertices evenly along a horizontal line, in the order given by the Hamiltonian path. Vertices adjacent on the path are connected with edges that are straight-line segments. All other edges are attached either at the top or at the bottom side of the vertices and are connected by half-circles. The Hamiltonian cycle splits the remaining edges of the graph into two groups: one with edge inside the cycle and the other with edges outside the cycle. We route these edges using half-circle arcs above and below the horizontal line containing the vertices, respectively. It is easy to see that this can be accomplished using edges of complexity one; see Fig. 11b.

For non-Hamiltonian graphs, we use the technique of Kaufmann and Wiese of splitting separating triangles by inserting dummy vertices on appropriate edges and triangulating again, to make the graph 4-connected. In this way, the general case can be reduced to the Hamiltonian case, but an edge may now contain a middle dummy vertex on the horizontal line, and hence it might consist of two segments, an upper and a lower half-circle. It is also easy to see that this results in a smooth layout with edges of complexity at most two, which yields the theorem below.

Theorem 10. *Any planar graph admits a smooth complexity-2 layout in the Kandinsky model.*

Corollary 11 *Any planar Hamiltonian graph admits a smooth complexity-1 layout in the Kandinsky model.*

Corollary 12 *Any planar Hamiltonian degree-3 graph can be drawn with complexity one in the smooth orthogonal model.*

7 Conclusion and Open Problems

In this paper, we introduced and presented the first combinatorial results for smooth orthogonal layouts in the context of improved drawing readability and drawing aesthetics. Of course, there are several natural open problems. In traditional orthogonal layouts the problem of testing whether a given graph has an embedding with only straight-line segments is NP-hard. The complexity of the corresponding problem for smooth complexity-1 layouts is not known. We proved that any biconnected 4-planar graph admits a smooth complexity-3 layout. However, it remains open whether the same holds for the case of non-biconnected graphs. Another question is whether all graphs that

have complexity-2 orthogonal layouts also admit smooth complexity-1 layouts. More generally, does there exist a 4-planar graph that has a complexity-k layout but does not have a smooth complexity-$(k-1)$ layout for any $k > 1$?

References

1. S. N. Bhatt and F. T. Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28:300–343, 1984.
2. T. Biedl and G. Kant. A better heuristic for orthogonal graph drawings. In *2nd European Symposium on Algorithms (ESA'94)*, pages 24–35, 1994.
3. T. C. Biedl and M. Kaufmann. Area-efficient static and incremental graph drawings. In *5th European Symposium on Algorithms (ESA'97)*, pages 37–52, 1997.
4. U. Brandes and B. Schlieper. Angle and distance constraints on tree drawings. In *14th Symposium on Graph Drawing (GD'06)*, pages 54–65, 2007.
5. U. Brandes and D. Wagner. Using Graph Layout to Visualize Train Interconnection Data. *Journal of Graph Algorithms Applications*, 4(3):135–155, 2000.
6. T. M. Chan. Geometric applications of a randomized optimization technique. *Discrete Comput. Geom*, 22:547–567, 1999.
7. C. C. Cheng, C. A. Duncan, M. T. Goodrich, and S. G. Kobourov. Drawing planar graphs with circular arcs. *Discrete Comput. Geom.*, 25(3):405–418, 2001.
8. C. A. Duncan, D. Eppstein, M. T. Goodrich, S. G. Kobourov, and M. Löffler. Planar and Poly-Arc Lombardi Drawings. In *19th Symposium on Graph Drawing (GD'11)*, pages 309–319, 2011.
9. C. A. Duncan, D. Eppstein, M. T. Goodrich, S. G. Kobourov, and M. Nöllenburg. Lombardi Drawings of Graphs. *Journal of Graph Algorithms and Applications*, 16(1):85–108, 2011.
10. D. Eppstein, M. T. Goodrich, and J. Y. Meng. Confluent layered drawings. *Algorithmica*, 47(4):439–452, 2007.
11. B. Finkel and R. Tamassia. Curvilinear Graph Drawing Using the Force-Directed Method. In *12th Symposium on Graph Drawing (GD'04)*, pages 448–453. 2005.
12. U. Fößmeier and M. Kaufmann. Drawing high degree graphs with low bend numbers. In *3rd Symposium on Graph Drawing (GD'95)*, pages 254–266, 1995.
13. H. Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10:41–51, 1990.
14. M. T. Goodrich and C. G. Wagner. A framework for drawing planar graphs with curves and polylines. *Journal of Algorithms*, 37(2):399–421, 2000.
15. D. Holten and J. J. van Wijk. Force-directed edge bundling for graph visualization. *Computer Graphics Forum*, 28:983–990, 2009.
16. W. Huang, P. Eades, and S.-H. Hong. A graph reading behavior: Geodesic-path tendency. In *IEEE Pacific Visualization Symposium (PacificVis'09)*, pages 137–144, 2009.
17. G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16:4–32, 1996.
18. M. Kaufmann and R. Wiese. Embedding vertices at points: Few bends suffice for planar graphs. *Journal of Graph Algorithms Applications*, 6(1):115–129, 2002.
19. C. E. Leiserson. Area-efficient graph layouts (for VLSI). In *Proc. 21st Annual IEEE Symposium on Foundations of Computer Science*, pages 270–281, 1980.
20. M. Lombardi and R. Hobbs. *Mark Lombardi: Global Networks*. Independent Curators, 2003.
21. R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal of Computing*, 16:421–444, 1987.
22. R. Tamassia and I. Tollis. Planar grid embedding in linear time. *IEEE Transactions on Circuits and Systems*, 36(9):1230 –1234, 1989.
23. L. G. Valiant. Universality considerations in VLSI circuits. *IEEE Transaction on Computers*, 30(2):135–140, 1981.

A Complexity-1 Layouts of 3-Connected 3-Planar Graphs

Assume that an embedding of a 3-connected 3-planar graph G is given. Observe that any 3-connected 3-planar graph cannot contain vertices of degree 2. Hence, G is cubic. The basic idea of the algorithm is to appropriately decompose G into a spanning tree and a set of paths, so that the edges of the tree are drawn as straight-lines, while the edges of the paths are drawn either as quarter-circles or as semi-circles. This ensures that any 3-connected cubic embedded graph admits a smooth orthogonal layout of complexity one. Before, we proceed with the detailed description of our approach, we first present an auxiliary lemma which will simplify the presentation.

Lemma 13 *The edges of a 3-connected cubic embedded graph G can be colored blue or green and be directed so that the blue edges form a spanning tree of G and the green edges form a set of uni-directional paths.*

Proof: Let Π be a leftmost canonical ordering of G [13, 17]. We process each path of Π in turn. The first path consists of exclusively one edge, namely edge (v_1, v_2), which we color blue and direct towards vertex v_1. For each path $P = \{z_1, z_2, \ldots, z_\lambda\} \in \Pi$ later in this order, let c_ℓ and c_r be the left and right neighbors of P, respectively. We color edge (c_ℓ, z_1) and all edges of P blue and direct them towards c_ℓ, while we color edge (z_λ, c_r) green and direct it towards c_r. Observe that each vertex of P has blue out-degree one, with the exception of vertex $v_1 \in P_0$. Hence, the blue graph is a spanning tree of G. In addition, the *blue degree* of a vertex, which is defined ignoring the orientation of the edges, is 1, 2, or 3. Similarly, the *green degree* of a vertex is 0, 1, or 2. The latter property implies that the green edges form a set of directed paths on G. Note that only vertex v_2 has green indegree 2. □

Our drawing algorithm works into two phases. In the first phase, an orthogonal layout is constructed, which in turn, is transformed into a smooth orthogonal layout in the second phase of the algorithm. Our algorithm for the first phase is a modification of the algorithm of Kant [17] for drawing triconnected 3-planar graphs. More precisely, let $\Pi = \{P_0, \ldots, P_s\}$ be a leftmost canonical ordering of G and assume that we have drawn zero of more paths of Π. Let P_i, $0 \leq i \leq s$, be the next path in this order. If $P_i = P_0$, we draw the vertices of P_i along the line $L_0 : y = 0$, such that $x(v_1) = 0$ and $x(v_2) = 1$. If this is not the case, let c_ℓ and c_r be the left and right neighbors of P_i, respectively. If P_i consists of a single vertex, then it is placed along line $L_i : \ y = i$ and its x-coordinate is $x(c_\ell)$. Note that up to this point, our algorithm is identical to the algorithm of Kant. However, in the case where P_i consists of a chain of vertices, say $z_1, z_2, \ldots, z_\lambda$, we slightly differentiate. The leftmost vertex of P_i is drawn along the line $L_i : \ y = i$ and has x-coordinate $x(c_\ell)$. Now observe that the vertical strip defined by $x(c_\ell)$ and $x(c_r)$ (refer to the gray-colored region of Fig. 12b) may contain several vertical segments of green edges. Let ℓ_i be the vertical line containing the rightmost such segment (drawn dashed in Fig. 12b). We proceed to draw the remaining vertices of P_i (i.e., vertices $z_2, \ldots, z_\lambda$) at unit-distanced points along the line $L_i : \ y = i$ and to the right of ℓ_i by shifting the drawing (using the corresponding algorithm of Kant) such that $x(z_2) = x(\ell_i) + 1$ and $x(z_\lambda) = x(c_r) - 1$. In this way, all blue edges of G are drawn as straight-line segments, while all green edges point down and to the right, except for edge (v_1, v_n) which requires two bends (see Figure 12b). More importantly,

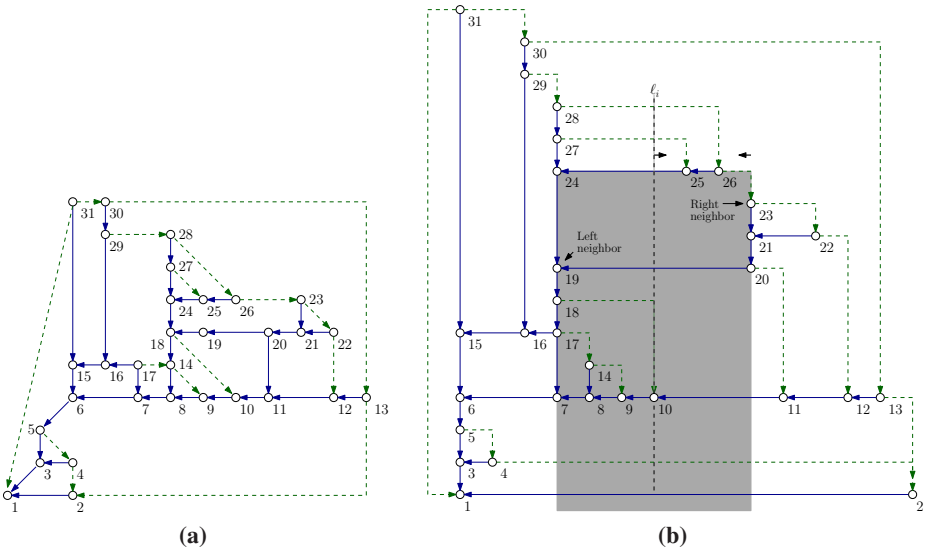

Fig. 12: (a) Edge coloring of a 3-connected cubic embedded graph (blue:solid, green:dashed). (b) The drawing obtained from the edge coloring of Fig.12a.

the vertical segments of all green edges are in different y-coordinates. Note that there may be a single edge overlap of the green edges entering v_2.

Lemma 14 *We can appropriately stretch the orthogonal drawing produced by our algorithm, so that the produced drawing is of smooth complexity one.*

Proof: In order to produce a drawing of complexity one, we process all green edges, except from the boundary green edges adjacent to edge (v_1, v_2), in the order implied by the y-coordinates of their source-vertices from bottom to top. Assume that we have processed zero or more green edges and let $e = (u_s, u_t)$ be the next green edge in this order. Let $h(e)$ and $v(e)$ be the lengths of the horizontal and vertical segments of edge e, respectively. We distinguish two case. If the length of the horizontal segment of edge e is longer than its matching vertical segment (i.e., $h(e) > v(e)$), we move all vertices of the orthogonal layout that are above or on the same y-coordinate as u_s vertically by $h(e) - v(e)$ grid points. If the length of the vertical segment of edge e is longer than its matching horizontal segment (i.e., $v(e) > h(e)$), we move all vertices of the orthogonal layout that are to the right of u_t horizontally by $v(e) - h(e)$ grid points. Hence, the length of the horizontal segment of edge e becomes equal to the length of its matching horizontal segment and therefore edge e can be drawn as a quarter-circle arc. Note that at the drawing of edge e, the stretching of its horizontal or vertical segment does not change the drawing of previously processed green edges. Therefore, once this procedure is completed, all green edges, except from the boundary green edges adjacent to edge (v_1, v_2), are drawn as quarter-circle arcs.

In order to complete the description of our transformation algorithm, it remains to consider the boundary green edges adjacent to edge (v_1, v_2). Edge (v_1, v_n) can be easily

drawn as a semi-circle. The other one can be also drawn as a semi-circle if we move vertex v_2 in the same y-coordinate of the rightmost vertex of the drawing of $G - v_2$, by maintaining the quarter-circle form of the other green edge incident to vertex v_2 (see Figure 13). □

Corollary 15 *Any triconnected* 3*-planar graph admits a smooth complexity-1 layout.*

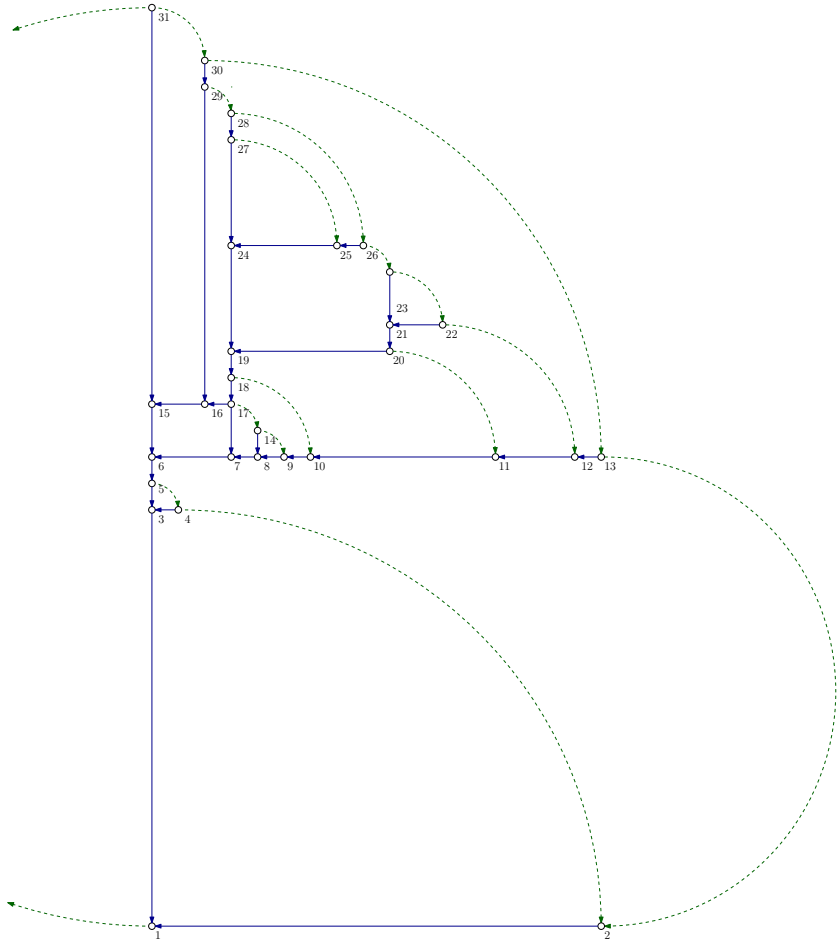

Fig. 13: A smooth orthogonal drawing obtained by appropriately transforming the drawing of Figure 12b.