Slanted Orthogonal Drawings*

Michael A. Bekos[1], Michael Kaufmann[1], Robert Krug[1], Stefan Näher[2], and Vincenzo Roselli[3]

[1] Institute for Informatics, University of Tübingen, Germany
{bekos,mk,krug}@informatik.uni-tuebingen.de
[2] Institute for Computer Science, University of Trier, Germany
naeher@uni-trier.de
[3] Engineering Department, Roma Tre University, Italy
roselli@dia.uniroma3.it

Abstract. We introduce a new model that we call *slanted orthogonal graph drawing*. While in traditional orthogonal drawings each edge is made of axis-aligned line-segments, in slanted orthogonal drawings intermediate diagonal segments on the edges are also permitted, which allows for: (a) smoothening the bends of the produced drawing (as they are replaced by pairs of "half-bends"), and, (b) emphasizing the crossings of the drawing (as they always appear at the intersection of two diagonal segments). We present an approach to compute bend-optimal slanted orthogonal representations, an efficient heuristic to compute close-to-optimal drawings in terms of the total number of bends using quadratic area, and a corresponding LP formulation, when insisting on bend optimality. On the negative side, we show that bend-optimal slanted orthogonal drawings may require exponential area.

1 Introduction

In this paper, we introduce and study a new model in the context of non-planar orthogonal graph drawing: Given a graph G of max-degree 4, determine a drawing Γ of G in which (a) each vertex occupies a point on the integer grid and has four available ports, as in the ordinary orthogonal model, (b) each edge is drawn as a sequence of horizontal, vertical and diagonal segments, (c) a diagonal segment is never incident to a vertex (due to port constraints mentioned above), (d) crossings always involve diagonal segments, and, (e) the minimum of the angles formed by two consecutive segments of any edge always is 135^o. We refer to Γ as the *slanted orthogonal drawing* of G, or, shortly, *slog drawing*. Figs.1(a) and 1(b) indicate what we might expect from the new model: crossings on the diagonals are more visible than in the traditional model and the use of area seems to be more effective.

* Part of the research was conducted in the framework of ESF project 10-EuroGIGA-OP-003 GraDR "Graph Drawings and Representations". 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.

S. Wismath and A. Wolff (Eds.): GD 2013, LNCS 8242, pp. 424–435, 2013.

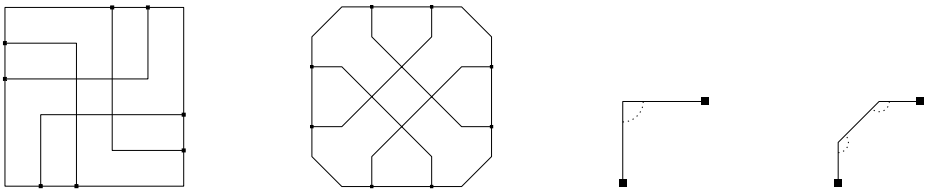


Fig. 1. (a)-(b) Traditional orthogonal and slanted orthogonal drawings of the same graph, assuming fixed ports. (c)-(d) Replacing a 90° bend by two half-bends of 135°.

Orthogonal graph drawing dates back to VLSI layouts and floor-planning applications [6,9,10]. In an *orthogonal drawing* of a graph of max-degree 4, each edge is drawn as a sequence of alternating horizontal and vertical line segments. Typical optimization functions include minimizing the used area [9], the total number of bends [4,8] or the maximum number of bends per edge [1].

Slog drawings are of improved readability and more aesthetic appeal than orthogonal drawings, since bends, which negatively affect the quality of orthogonal drawings are replaced by half-bends that have a smoother shape (see Figs.1(c) and 1(d)). In addition, slog drawings reveal crossings and help distinguishing them from vertices, since crossings are defined by diagonal segments, while vertices are incident to rectilinear segments. Our model resembles an octilinear model as it is heavily used for example in the drawing of Metro Maps [7] but it is closer to the traditional orthogonal style. In particular angles of 45° do not occur at all. So, the complexity results for the octilinear models do not apply.

For minimizing the total number of bends in orthogonal graph drawing Tamassia laid important foundations by the *topology-shape-metrics* (*TSM*) approach in [8], that works in three phases. In the first *planarization* phase a "planar" embedding is computed for a given (non)planar graph by replacing edge crossings by dummy vertices (referred to as *crossing or c-vertices*). The output is called *planar representation*. In the next *orthogonalization* phase, angles and bends of the drawing are computed, producing an *orthogonal representation*. In the third *compaction* phase the coordinates for vertices and edges are computed. The core is a min-cost flow algorithm to minimize the number of bends in the second phase [2]. We will adopt the TSM approach for our model. Note that the general problem of determining a planar embedding with the minimum number of bends is NP-hard [5], which is also the case for slog drawings. Therefore we assume in the following that a planar representation of the input graph is given.

While constructing the slog drawing we observe the following requirements: (a) all non-dummy vertices (referred to as *real or r-vertices*) use orthogonal ports and, (b) all c-vertices use diagonal ports. This ensures that the computed drawing will be a valid slog drawing that corresponds to the initial planar representation. Edges connecting real (crossing) vertices are referred to as rr-edges (cc-edges), and edges between r- and c-vertices as rc-edges.

This paper is structured as follows: In Section 2 we present an approach to compute bend-optimal slog representations. Afterwards, we present a heuristic to compute close-to-optimal slog drawings, that require polynomial drawing area, based on a given slog representation. To compute the optimal drawing, we give a formulation as a linear program in Section 4. In Section 5 we show that the optimal drawing may require exponential area. We conclude in Section 6.

2 A Flow-Based Approach

In this section, we present a modification of an algorithm by Tamassia [8], which we briefly describe in Section 2.1. Section 2.2 explains our modification and in Section 2.3, we present properties of a bend-minimal slog representation.

2.1 Preliminaries

A central notion to the algorithm of Tamassia [8] is the *orthogonal representation*, that captures the "shape" of the drawing without the exact geometry. An orthogonal representation of a plane graph $G = (V, E)$ is an assignment of four labels to each edge $(u, v) \in E$; two for each direction. Label $\alpha(u, v) \cdot 90°$ corresponds to the angle at vertex u formed by (u, v) and its counterclockwise next incident edge. Label $\beta(u, v)$ corresponds to the number of left turns along (u, v), when traversing it from u to v. Clearly, $1 \leq \alpha(u, v) \leq 4$ and $\beta(u, v) \geq 0$. The sum of angles around a vertex equals to 360°, so for each vertex $u \in V$, $\sum_{(u,v)\in N(u)} \alpha(u, v) = 4$, where $N(u)$ denotes the neighbors of u. Similarly, since the sum of the angles formed at the vertices and at the bends of a bounded face f equals to $180°(p(f) - 2)$, where $p(f)$ denotes the number of such angles, it follows that $\sum_{(u,v)\in E(f)} \alpha(u, v) + \beta(v, u) - \beta(u, v) = 2a(f) - 4$, where $a(f)$ denotes the number of vertex angles in f, and, $E(f)$ the directed arcs of f in its counterclockwise traversal. If f is unbounded, the sum is increased by 8.

In the flow network one can think of each unit of flow as a 90° angle. The vertices (vertex-nodes; sources) supply 4 units of flow, and each face (face-nodes; sinks) f demands $2a(f) - 4$ units of flow (plus 8 if f is unbounded). To maintain the properties described above each edge from a vertex-node to a face-node in the flow network has a capacity of 4 and a minimum flow of 1, while an edge between adjacent faces has infinite capacity, no lower bound but each unit of flow through it costs one unit. The total cost is actually the number of bends along the corresponding edge. Hence, the min-cost flow solution corresponds to a representation with the minimum number of bends.

2.2 Modifying the Flow Network

We now modify the algorithm of Tamassia, to obtain a slog representation of a planarized graph G with minimum number of half-bends. Recall that G contains two types of vertices, namely real and crossing vertices. Real (crossing) vertices use orthogonal (diagonal) ports. Observe that a pair of half-bends on an rr- or

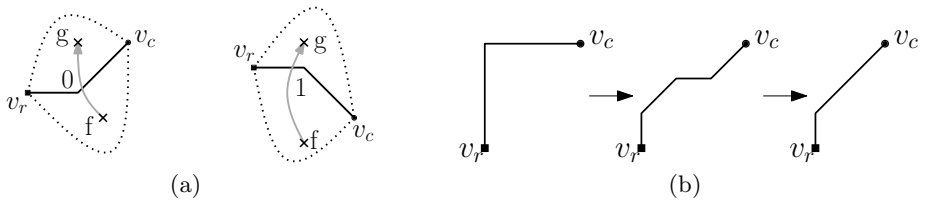


Fig. 2. (a) Two configurations corresponding to zero or one unit of flow over an rc-edge (with f and g being the two adjacent faces) (b) The right rotation of a c-vertex v_c can save a half-bend.

cc-edge of a slog drawing corresponds to a bend of an orthogonal drawing. But a rc-edge changes from an orthogonal port (incident to the r-vertex) to a diagonal port (incident to the c-vertex), requiring at least one half-bend. Consider an rc-edge (v_r, v_c) incident to faces f and g (see Fig.2(a)) and assume that the port of the real vertex v_r is fixed. Depending on the rotation of the crossing vertex v_c (clockwise or counterclockwise) we obtain two different representations with the same number of bends. To model this "free-of-cost" choice, we introduce an edge into the flow network connecting f and g with unit capacity and zero cost. For consistency we assume that, if in the solution of the min-cost flow problem there is no flow over (f, g), then there exists a left turn from the real to the crossing vertex; otherwise a right turn, as illustrated in Fig.2(a).

2.3 Properties of Optimal Slanted Orthogonal Representations

In this section we give properties of optimal slog representations. We prove that, for a planarized graph G, the computation of a slog representation with minimum number of half-bends respecting the embedding of G is always feasible. Then, we give an upper bound on the number of half-bends in optimal slog representations.

Theorem 1. *For a planarized graph G with max-degree 4, we can efficiently compute a slog representation with minimum number of half-bends respecting the embedding of G.*

Proof. We use a reduction to Tamassia's network flow algorithm. In particular, since the original flow network computes a (bend-minimal) orthogonal representation for the input plane graph, we will also obtain a slog representation with our modification. We now prove that this representation is also bend-minimal.

Assume that we are given an orthogonal representation F. We can uniquely convert F into a slog representation $S(F)$ by turning all crossing vertices counterclockwise by 45°. More precisely, the last segment of every rc-edge before the crossing vertex will become a left half-bend. Furthermore, every orthogonal bend is converted into two half-bends, bending in the same direction as the orthogonal bend (see Fig.1(c) and 1(d)). Note that the left half-bends at the crossings might neutralize with one of the half-bends originating from an orthogonal bend, if the orthogonal bend is turning to the right (see Fig.2(b)). In this case, only the second one of the right half-bends remains. Note that this is the only possible

saving operation. Therefore, since the number of rc-edges is fixed from the given embedding, a slog representation with minimum number of half-bends should minimize the difference between the number of orthogonal bends of F and the number of first right-bends on rc-edges. However, this is exactly what is done by our min-cost flow network formulation, as the objective is the minimization of the total number of bends in F without the first right-bends on rc-edges. □

This constructive approach can be reversed so that for each slog representation S, we can get a unique orthogonal representation $F(S)$. Clearly, $F(S(F)) = F$ and $S(F(S)) = S$. Note that this is true only for bend-minimal representations; otherwise we might have staircases of bends which is impossible by min-cost flow computations. From the construction, we can also derive the following.

Corollary 1. *Let $S(F)$ be a slog representation, and F a corresponding orthogonal representation. Let b_S, rb_S and rc_S be the number of half-bends, the number of first right-bends on rc-edges and the number of rc-edges in $S(F)$. Let also b_F be the number of orthogonal bends in F. Then, $b_S = 2 \cdot (b_F - rb_S) + rc_S$.*

The following theorem gives an upper bound for the number of half-bends in optimal slog representations.

Theorem 2. *The number of half-bends of a bend-minimal slog representation is at least twice the number of bends of its related bend-minimal orthogonal representation.*

Proof. Bends of a bend-minimal orthogonal representation correspond to pairs of half-bends on cc and rr edges of a bend-minimal slog representation. In this case, the claim holds with equality. For rc-edges we need a different argument. Let $\mathcal{C}$ be a maximal component spanned by cc-edges. Then, all edges with exactly one endpoint in $\mathcal{C}$ are rc-edges, which can be split into independent cycles around components of crossings. Let C be such a cycle of length k. Clearly, there should be k first half-bends on the rc-edges in the slanted representation. In the corresponding orthogonal representation, the second and third bend of each rc-edge correspond to pairs of half-bends on the same edge in the slog representation. Similarly, in the orthogonal representation the first orthogonal left-bend of each rc-edge corresponds to the second and third left half-bend of the same edge in the slog representation. The only bends that have not been paired (i.e., have no correspondence) are the first right-bends on rc-edges. We claim that in any bend-minimal orthogonal representation, there exist at most $\frac{k}{2}$ first right bends on the edges of C. Assume to the contrary, that in a bend-minimal orthogonal representation, there exist $r > \frac{k}{2}$ first right-bends on the edges of C. If we send the flow along C in reverse direction, we decrease the number of right-bends by r and increase the number of left bends by $k - r$. So, the total number of bends decreases, which shows that the input orthogonal representation was not minimal. From the claim, it follows that the number of first right-bends in the orthogonal representation is at most half of the number of first half-bends of C (in the corresponding slog representation), which concludes the proof since all other half-bends come in pairs and have their correspondences. □

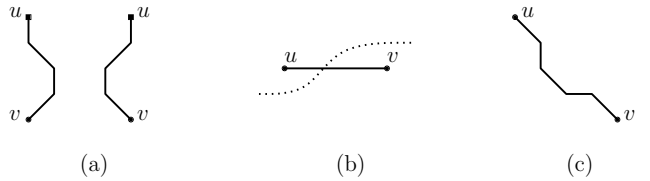


Fig. 3. (a) Spoon gadget for rc-edges. (b) Moving everything above the dotted cut up transforms the orthogonal input to a slanted drawing (c) containing 4 half-bends.

3 A Heuristic to Compute a Close to Optimal Drawing

In this section we present a heuristic which, given an optimal slog representation, computes an actual drawing, which is close to optimal, with respect to the total number of bends, and requires quadratic area. This is quite reasonable, since as we will see in Section 5, insisting in optimal slog drawing may result in exponential area. The basic steps of our approach are given in Algorithm 1. In the following, we describe them in detail.

Algorithm 1. Spoon Based Algorithm

Input: A slanted orthogonal representation F of a given planarized graph G
Output: A slanted orthogonal drawing $\Gamma_s(G)$

S1: Compute an orthogonal drawing $\Gamma_o(G)$ based on F
S2: Replace each orthogonal bend by 2 half-bends {see Figs.1(c) and 1(d)}
S3: Fix ports on rc-edges by inserting the spoon gadget {see Fig.3(a)}
S4: Apply cuts to fix ports on cc-edges {see Figs.3(b) and 3(c)}
S5: Optimize number of rc half-bends {see Fig.4(a)}
S6: Optimize number of cc half-bends {see Fig.4(b)}
S7: Compact drawing

In step 1 of Algorithm 1 we compute a bend-minimal orthogonal drawing $\Gamma_o(G)$ from the slog representation. We use the original algorithm of Tamassia [8], ignoring the flow on the additional edges and the rotation of the crossing vertices. In step 2 of Algorithm 1, we replace all orthogonal bends with pairs of half-bends. In step 3 of Algorithm 1, we connect r-vertices with c-vertices by replacing the segment incident to the c-vertex by a gadget called *spoon*, due to its shape (see Fig.3(a)). It allows switching between orthogonal and diagonal ports on an edge. Note that the slog representation specifies how each c-vertex is rotated, thereby defining the configuration it uses.

In step 4 of Algorithm 1, we employ *cuts* (i.e., is a standard technique to perform layout stretching in orthogonal graph drawing; see [3]) to fix the ports of cc-edges, which still use orthogonal ports. To apply a horizontal (vertical) cut, we have to ensure that each edge crossed by the cut has at least one horizontal

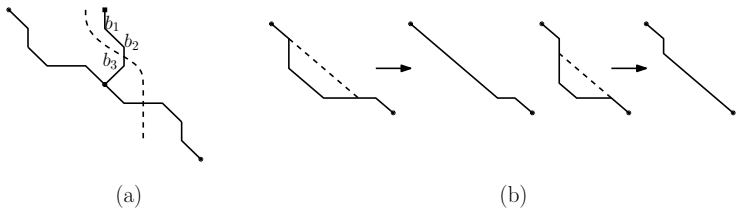


Fig. 4. (a) There is always a cut like the dashed line enabling us to move everything on its left side to the left to save half-bends b_1 and b_2. (b) Saving bends on cc-edges.

(vertical) segment. This trivially holds before the introduction of the spoons, as $\Gamma_o(G)$ is an orthogonal drawing. It also holds afterwards since a spoon replacing a horizontal (vertical) segment has two horizontal (vertical) segments. To fix a horizontal cc-edge (u, v) with u being to the left of v in the drawing, we use a "horizontal cut" which from left to right and up to vertex u either (a) lies exactly above u, then crosses edge (u, v) and stays exactly below v, or, (b) lies exactly below u, then crosses edge (u, v) and stays exactly above v (see Fig.3(b)). Our choice depends on the slog representation that specifies the rotation of each c-vertex. The result is depicted in Fig.3(c). Observe that the edge has now a horizontal and a vertical segment. Hence, we can fix all remaining cc-edges. To cope with cc-edges with bends we apply the same technique only to the first and last segments of the edge.

The resulting drawing has 2 additional half-bends on rc-edges (the spoon gadget adds 3 half-bends; one is required) and 4 additional half-bends on cc-edges (none is required), with respect to the ones suggested by the representation. By applying cuts again, we can save 2 half-bends for each rc-edge (see Fig.4(a)), by eliminating the diagonal segment of the spoon gadget (step 5 of Algorithm 1). The rectilinear segments of the edge are not affected, to be able to apply future cuts.

It is always possible to remove two of the half-bends on cc-edges (step 6 of Algorithm 1) by a local modification as depicted in Fig.4(b). If the horizontal part of a cc-edge is longer than the vertical one, a shortcut as in the left part of Fig.4(b) can be applied. If the horizontal and the vertical segments of the cc-edge have the same length all four half-bends can be saved.

After applying this operations, the drawing will contain zero additional half-bends on rr-edges and at most two additional half-bends on cc-edges, with respect to the input representation. Note that to apply our technique we need to scale up the initial drawing by a factor of 4 at the beginning, to provide enough space for additional half-bends. In subsequent steps, we increase the drawing area by cuts. However, we can reduce it by contracting along horizontal and vertical cuts at the end (step 7 of Algorithm 1). After the compaction, each horizontal and vertical grid line will be occupied by at least a half-bend, an edge or a vertex, and since all of those are linear in number, the required area of the final slanted drawing is $O(n^2)$. The following theorem summarizes our approach.

Theorem 3. *Given a slog representation of a planarized graph G with max-degree 4, we can efficiently compute a slog drawing requiring $O(n^2)$ area with (i) optimal number of half-bends on rr edges and rc edges without bends and (ii) at most two additional half-bends on cc edges and rc edges with bends.*

4 A Linear Program to Compute Optimal Drawings

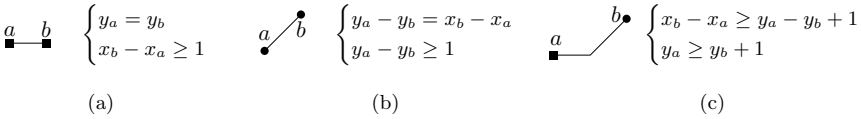
This section describes how to model a given slog representation $\mathcal{S}$ of a plane graph G as a Linear Program (LP). Based on $\mathcal{S}$, we modify G and obtain a graph G', that is a subdivision of G and has at most one half-bend on each edge.

Let $\langle b_1, \ldots, b_k \rangle$, $k \geq 2$, be the half-bends of edge (u, w) of G in $\mathcal{S}$, appearing in this order along (u, w) from u to w. Say that u is an r-vertex. We add a new c-vertex v and replace (u, w) by edges (u, v) and (v, w). The first half-bend b_1 is assigned to (u, v) and $\langle b_2, \ldots, b_k \rangle$ to (v, w). If u is a c-vertex, then v would have been an r-vertex. Observe that the type of v and its ports are defined by the slope of the segments incident to b_1 in $\mathcal{S}$. By repeating this procedure, we obtain G', that is a subdivision of G having at most one half-bend on each edge.

Each face f of G has a corresponding face f' in G' such that: (i) the vertices of G' incident to f' are the same as those incident to f in G, plus the ones from the subdivision; and (ii) the sequence of slopes assigned to the segments bounding f' is the same as that of the segments bounding f. So a drawing Γ' of G' realizing the slog representation is also a drawing of G realizing $\mathcal{S}$.

For each vertex v of G' we define two variables x_v and y_v, representing its coordinates on the plane. For each edge (a, b) of G', we define a pair of constraints similar to those in Table 1, depending on the type of vertices of a and b. The table provides an example for each type, the other configurations are analogous.

Table 1. Examples of constraints of the linear program for (a) rr-edges, (b) cc-edges and (c) rc-edges, assuming the y-axis points downwards.



(a) (b) (c)

We indirectly minimize the area of the produced drawing by minimizing the total edge length in the objective function. The slopes of the segments allow us to express the Euclidean length of each edge.

Despite the fact that every experiment we made on random and crafted graphs led to a feasible solution, we could not prove the feasibility of the LP. Nevertheless we believe that our LP always admits a feasible solution.

4.1 Addressing Planarity Issues in the LP

The LP models the shape of the edges and the relative positions of all nodes connected by an edge. Since there are no constraints for non-connected nodes

the resulting drawing could be non-planar. An example is given in Fig.5 where the relative position of nodes r_3 and c_2 is not defined by the LP. To solve this problem, we cannot apply the approach used in the original Tamassia algorithm (cutting all faces into rectangles), since our faces are, in general, not rectilinear.

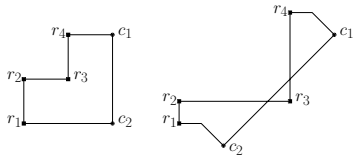


Fig. 5. A configuration that could result in a non-planar solution

In slog drawings we distinguish different corner types. There are *vertex-corners* (or simply vertices) and *bend-corners* (or simply bends). A corner is either *convex* with respect to a face, if the inner angle is $\leq 135°$, or *non-convex* otherwise. An angle of 180° at a vertex is not a corner, since it will be aligned with its neighbors by construction. This gives four possible corners: (non-)/convex vertex(or bend). To ensure planarity, we use *split-edges* and the notion of *almost-convex* faces.

Definition 1. *A* split-edge *is an edge that connects either (a) a non-convex vertex-corner v with a new vertex that subdivides a side parallel to one of the edges incident to v, or, (b) two new vertices that subdivide two parallel edges, when one of them is incident to a non-convex bend-corner (see Figs.6(a) and 6(b)).*

Definition 2. *A face f is* almost-convex *if it does not contain any non-convex vertex-corners and no split-edge exists that separates f into two non-convex faces.*

First we make all faces almost-convex. To remove non-convex vertex-corners we introduce a new split-edge; see Fig.6(a). If there is no parallel side to one of the segments of a vertex-corner we use a special structure, which we call *nose-gadget* (due to its shape); see Fig.6(c). The dashed line in the figure represents the split-edge we can apply once the gadget (dotted line) is added. The two vertices that are added on the diagonals are c-vertices, while the third one is an r-vertex. By applying the split edge the non-convex vertex corner is removed.

After the previous step no face contains non-convex vertex-corners. If a face contains non-convex bends and is not almost-convex, we search for a split-edge that creates two non-convex faces (as in Fig.6(b)). We apply such split-edges until all faces are almost-convex. Observe that all additional edges can be expressed by using the original set of constraints from the LP. To prove that we can always make all faces almost-convex, we give the following lemmas.

Lemma 1. *If a face contains two segments s_1 and s_2 defining a non-convex bend-corner, then it contains a segment s_3 parallel to either s_1 or s_2.*

Sketch of Proof. Assume to the contrary that there is no segment parallel to s_1 and s_2. Let b be the non-convex bend, and, r and c its adjacent corners;

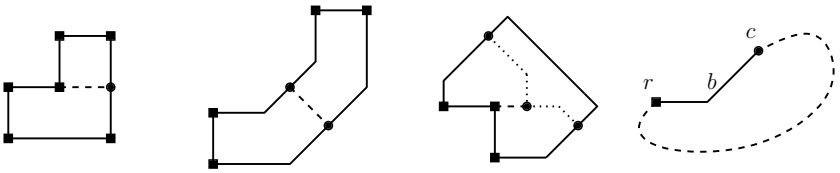


Fig. 6. Split-edges are dashed lines on (a) a vertex and (b) a bend. (c) If no split-edge is possible the nose gadget (dotted line) is used. Circles are additional nodes.(d) This face can not be completed without using at least one segment parallel to (r, b) or (b, c).

see Fig.6(d). Then, r and c cannot be connected by a polygonal chain of edges without using one of the two slopes defined by (r, b) and (b, c).

Lemma 2. *An almost-convex face f has at most two consecutive non-convex bend-corners.*

Proof. Assume that f has three consecutive non-convex corners c_1, c_2, c_3. By Lemma 1, there exists a parallel segment to one of those defining c_2. Then, there exists a split-edge separating f into two non-convex faces, one containing c_1 and one containing c_3, which is a contradiction. □

Lemma 3. *An almost-convex face f has at most two non-convex bend-corners.*

Sketch of Proof. Assuming that f contains at least three non-convex corners, one can lead to a contradiction the fact that f is almost-convex.

Lemma 4. *An almost-convex face f is always drawn planar.*

Sketch of Proof. From Lemma 3, it follows that f has at most two non-convex bend-corners. If f contains no or a single non-convex bend-corner, then it is easy to see that f is always drawn planar. If f contains exactly two non-convex bend-corners, then by Lemma 2 f cannot have more than two consecutive non-convex bend-corners. Hence, there exist in total four cases that one has to consider with respect to the shape of f. In all of them f is drawn planar.

With these lemmas we have shown how to subdivide all faces of a graph $G = (V, E)$ to obtain a graph $G' = (V', E')$ that only contains almost-convex faces. For G' we can compute a planar drawing using our linear program, giving a planar drawing for G. We ensured the planarity of the drawing by adding N new vertices and edges, where N is $O(|E|)$, since there is at most one split-edge for each vertex and for each non-convex bend. All additional edges can be modeled by the original set of constraints. Experiments on random and crafted graphs seem to confirm that our linear program always has a feasible solution.

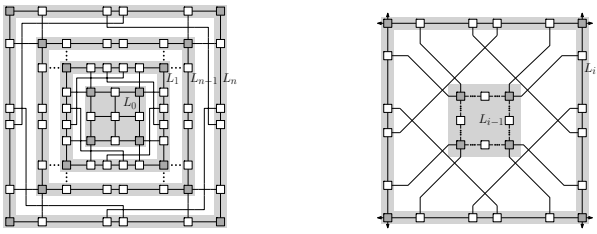
5 Area Bounds

Slog drawings have aesthetic appeal and improve the readability of non-planar graphs, when compared to traditional orthogonal drawings. However, in this section we show that such drawings might require increased drawing area. Note that most of the orthogonal drawing algorithms require $O(n) \times O(n)$ area. The situation is different if we want to generate slog drawings of optimal number of bends. In particular, we show that the area penalty can be exponential.

Theorem 4. *There exists a graph G whose slanted orthogonal drawing $\Gamma(G)$ of minimum number of bends requires exponential area, assuming that a planarized version $\sigma(G)$ of the resulting drawing is given.*

Proof. The planarized version $\sigma(G)$ of G is given in Fig.7(a) and consists of $n+1$ layers $L_0, L_1, \ldots, L_n$. Layer L_0 is the square grid graph on 9 vertices. Each layer L_i, $i = 1, 2, \ldots, n$, is a cycle on 20 vertices. Consecutive layers L_{i-1} and L_i, $i = 1, 2, \ldots, n$, are connected by 12 edges which define 12 crossings. Hence, G consists of $20n + 9$ vertices and $32n + 13$ edges that define $12n$ crossings.

A slog drawing $\Gamma(G)$ of G with minimum number of bends derived from $\sigma(G)$ ideally introduces (a) no bends on crossing-free edges of $\sigma(G)$, and, (b) two half-bends in total for each rc-edge. Now observe that at each layer there exist four vertices, that have two ports pointing to the next layer (gray-colored in Fig.7(a)). This together with requirements (a) and (b) suggests that the vertices of each layer L_i should reside along the edges of a rectangle, say R_i, such that the vertices of L_i whose ports point to the next layer coincide with the corners of R_i, $i = 0, 1, 2, \ldots, n$ (with the only exception of the "innermost" vertex of L_0; in Fig.7(b), R_i is identified with cycle L_i). Hence, the routing of the edges that connect consecutive layers should be done as illustrated in Fig.7(b). Since L_0 is always drawable in a 3×3 box meeting all requirements mentioned above, and, $\sigma(G)$ is highly symmetric, we can assume that each R_i is a square of side length w_i, $i = 0, 1, 2, \ldots, n$. Then, it is not difficult to see that $w_0 = 3$ and $w_{i+1} = 2w_i + 8$, $i = 1, 2, \ldots, n$. This implies that the area of $\Gamma(G)$ is exponential in the number of layers of G and therefore exponential in the number of vertices of G (recall that G has $n + 1$ layers and $20n + 9$ vertices). □



(a) (b)

Fig. 7. (a) A planarized version $\sigma(G)$ of a graph G. (b) Edges involved in crossings in $\sigma(G)$ contribute two half-bends.

6 Conclusion and Open Problems

We introduced a new model for drawing graphs of max-degree four, in which orthogonal bends are replaced by pairs of "slanted" bends and crossings occur on diagonal segments only. The main advantage of this model is that, even in drawings of large graphs (where vertices might not be clearly visible), it is immediately clear which pair of edges induce a crossing and where such a crossing is located in the drawing. We presented an algorithm to construct slog drawings with almost-optimal number of bends and quadratic area, for general max-degree four graphs. By a modification of Tamassia's min-cost flow approach, we showed that a bend-optimal representation of the graph can efficiently be computed in polynomial time and we presented an LP-approach to compute a corresponding drawing. A natural problem is whether every max-degree four graph admits such a drawing. Our experiments on randomly generated and crafted inputs led us to believe that it is possible, although we could not prove it.

References

1. Biedl, T.C., Kant, G.: A better heuristic for orthogonal graph drawings. In: van Leeuwen, J. (ed.) ESA 1994. LNCS, vol. 855, pp. 24–35. Springer, Heidelberg (1994)
2. Cornelsen, S., Karrenbauer, A.: Accelerated bend minimization. Journal of Graph Algorithms Applications 16(3), 635–650 (2012)
3. Fößmeier, U., Heß, C., Kaufmann, M.: On improving orthogonal drawings: The 4M-algorithm. In: Whitesides, S.H. (ed.) GD 1998. LNCS, vol. 1547, pp. 125–137. Springer, Heidelberg (1999)
4. Fößmeier, U., Kaufmann, M.: Drawing high degree graphs with low bend numbers. In: Brandenburg, F.J. (ed.) GD 1995. LNCS, vol. 1027, pp. 254–266. Springer, Heidelberg (1996)
5. Garg, A., Tamassia, R.: On the computational complexity of upward and rectilinear planarity testing. SIAM Journal of Computing 31(2), 601–625 (2001)
6. Leiserson, C.E.: Area-efficient graph layouts (for VLSI). In: 21st Symposium on Foundations of Computer Science, vol. 1547, pp. 270–281. IEEE (1980)
7. Nöllenburg, M., Wolff, A.: Drawing and labeling high-quality metro maps by mixed-integer programming. IEEE Trans. Vis. Comput. Graph. 17(5), 626–641 (2011)
8. Tamassia, R.: On embedding a graph in the grid with the minimum number of bends. SIAM Journal of Computing 16(3), 421–444 (1987)
9. Tamassia, R., Tollis, I.G.: Planar grid embedding in linear time. IEEE Transactions on Circuits and Systems 36(9), 1230–1234 (1989)
10. Valiant, L.G.: Universality considerations in VLSI circuits. IEEE Transaction on Computers 30(2), 135–140 (1981)