Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 18, no. 3, pp. 459–489 (2014)
DOI: 10.7155/jgaa.00332

Slanted Orthogonal Drawings: Model, Algorithms and Evaluations

M. A. Bekos[1] *M. Kaufmann*[1] *R. Krug*[1] *T. Ludwig*[1] *S. Näher*[2] *V. Roselli*[3]

[1]Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Germany
[2]Institute for Computer Science, University of Trier, Germany
[3]Engineering Department, Roma Tre University, Italy

Abstract

We introduce a new model in the context of non-planar orthogonal graph drawing that we call *slanted orthogonal graph drawing.* While in traditional orthogonal drawings each edge is made of alternating axis-aligned line segments, in slanted orthogonal drawings intermediate diagonal segments on the edges are 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 slanted orthogonal drawings with respect to the total number of bends in 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.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
December 2013	August 2014	September 2014	November 2014	November 2014

Published: November 2014

Article type: Regular paper | Communicated by: G. Liotta

This article is based on the preliminary version [M. A. Bekos, M. Kaufmann, R. Krug, S. Näher, V. Roselli, Slanted Orthogonal Drawings, In S. Wismath and A. Wolff editors, Proc. of 21st International Symposium on Graph Drawing (GD2013), LNCS 8242, pp. 244-255, 2013].

E-mail addresses: bekos@informatik.uni-tuebingen.de (M. A. Bekos) mk@informatik.uni-tuebingen.de (M. Kaufmann) krug@informatik.uni-tuebingen.de (R. Krug) ludwigt@informatik.uni-tuebingen.de (T. Ludwig) naeher@uni-trier.de (S. Näher) roselli@dia.uniroma3.it (V. Roselli)

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 $\Gamma(G)$ of G in which: (a) each vertex occupies a point on the integer grid and has four available *ports*, as in the ordinary orthogonal graph drawing model; (b) each edge is drawn as a sequence of alternating 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 an edge always is $135°$, which suggests that a bend in $\Gamma(G)$ is always incident to a diagonal segment and to either a horizontal or a vertical one. We refer to $\Gamma(G)$ as the *slanted orthogonal drawing* of G, or, shortly, *slog drawing*. For an example, refer to Figure 1(a). The corresponding slog drawing of this example is illustrated in Figure 1(b). This example indicates what we might expect from the new model: crossings on the diagonals are more visible than the corresponding ones in the traditional orthogonal graph drawing model and the use of area seems to be more demanding.

Slog drawings generalize orthogonal drawings in the following sense: If the input graph G is planar, then any planar orthogonal drawing $\Gamma(G)$ of G can be transformed into a planar slog drawing $\Gamma'(G)$ of G, by replacing each bend of $90°$ of $\Gamma(G)$ by two "*half-bends*" of $135°$ in $\Gamma'(G)$, as illustrated in Figures 1(c) and 1(d)[1]. The resulting drawings will be of improved readability and more aesthetic appeal, since bends, which negatively affect the quality of orthogonal drawings (as they interrupt the eye movement and require sharp changes of direction), are replaced by pairs of half-bends that have a smoother shape. In addition, in the non-planar case, slog drawings reveal the presence of crossings and help distinguishing them from vertices of the drawing, because crossings are defined by diagonal segments, while vertices are always incident to rectilinear segments.

1.1 Related Work

Orthogonal graph drawing has a long tradition, dating back to VLSI layouts and floor-planning applications [9, 14, 15]. Formally, an *orthogonal drawing* of a graph of max-degree 4 is a drawing in which each edge is drawn as a sequence of alternating horizontal and vertical line segments. Usually, one wants to compute an orthogonal drawing that is optimal under a pre-specified optimization function which measures the niceness of the resulting drawing. Typical optimization functions include minimizing the used area [12, 14], the total number of bends [6, 7, 13] or the maximum number of bends per edge [2, 10]; for an overview see e.g. [4].

For minimizing the total number of bends in orthogonal graph drawing Tamassia laid important foundations by the *topology-shape-metrics* (*TSM*) ap-

[1]Potential crossings posed by the presence of half-bends can be avoided, if one scales $\Gamma'(G)$ by a factor of 2 and the diagonal segment defined by a pair of half-bends lies in a 1×1 box.

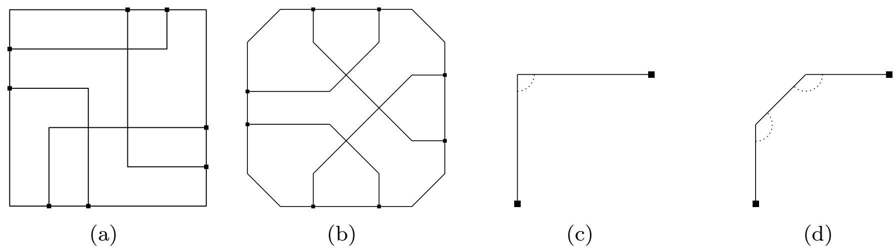


Figure 1: (a)-(b) Traditional orthogonal and slanted orthogonal drawings of the same graph, assuming fixed ports. (c)-(d) Replacing a 90° bend by two half-bends of 135°.

proach in [13], 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 vertices or c-vertices*). The output is called a *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 [3]. Note that the general problem of determining a planar embedding with the minimum number of bends is NP-hard [7].

Our model resembles an octilinear model which is heavily used for example in the drawing of metro maps [11] but it is closer to the traditional orthogonal style. In particular, angles of 45° do not occur at all. Therefore, the complexity results for the octilinear model do not apply to our model.

Closely related to the problem we study is also the *smooth orthogonal drawing* problem [1], which asks for a planar drawing of an input planar graph of maximum degree 4, in which every edge is made of axis-aligned line segments and circular-arcs with common horizontal or vertical tangents; the main goal is to determine such drawings with low edge complexity, measured by the number of line segments and circular-arc segments forming each edge. Note that both approaches try to smoothen orthogonal drawings either by the usage of circular arc segments (smooth orthogonal drawings) or by replacing orthogonal bends by half-bends (slog drawings).

1.2 Preliminaries and Notation

Let $G = (V, E)$ be an undirected graph. Given a drawing $\Gamma(G)$ of G, we denote by $p_u = (x_u, y_u)$ the position of vertex $u \in V$ on the plane. The degree of vertex $u \in V$ is denoted by $d(u)$. Let also $d(G) = max_{u \in V} d(u)$ be the degree of graph G. Given a pair of points $q, q' \in \mathbb{R}^2$, we denote by $|qq'|$ the Euclidean distance between q and q'. We refer to the line segment defined by q and q' as $\overline{qq'}$.

For planar slog drawings, observe that the problem of minimizing the number of bends over all embeddings of an input planar graph of maximum degree

4 is NP-hard. This directly follows from [7], since the bends of a planar orthogonal drawing are in one to one correspondence with pairs of half-bends of the corresponding slanted orthogonal drawing. This negative result led us to adopt the TSM approach for our model. So, in the following, we assume that a planar representation of the input graph is given. Then, one can easily observe the following requirements: (a) all non-dummy vertices (referred to as *real vertices 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.*

Throughout this paper, we also use the notion of a *left* or *right* turn, which we formally define in the following.

Definition 1 *Let $e = (u, v)$ be an edge with at least one bend, say b, and let s and s' be two consecutive segments of e with b being the common bend of s and s'. Furthermore, let ϕ be the angle formed by s and s' on their left side when moving along e from u to v. Edge e has a* left turn *on b if $\phi \leq 180°$, otherwise there is a* right turn *on b.*

1.3 Our Contribution.

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. In Sections 6 and 7, we present an experimental evaluation and some sample drawings of our algorithms, respectively. We conclude in Section 8 with open problems for future work.

2 Bend-Optimal Slanted Orthogonal Representations

In this section, we present an algorithm for computing a bend-optimal slog representation of an input plane graph of maximum degree 4. This algorithm is a modification of a well-known algorithm by Tamassia [13] for computing bend-optimal orthogonal representations of plane graphs of maximum degree 4, by modeling the problem as a min-cost flow problem on a flow network derived from the embedding of the graph. However, before we proceed with the detailed description of our modification (in Section 2.2), we briefly describe the algorithm of Tamassia (in Section 2.1). Section 2.3 presents properties of bend-minimal slog representations.

2.1 Preliminaries

A central notion to the algorithm of Tamassia [13] is the *orthogonal representation*, which in a sense captures the "shape" of the resulting drawing, neglecting the exact geometry underneath. Typically, 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 edge (u, v) and its next incident edge counterclockwise around u. Label $\beta(u, v)$ corresponds to the number of left turns of angle $90°$ along (u, v), when traversing it from u towards v. Clearly, $1 \leq \alpha(u, v) \leq 4$ and $\beta(u, v) \geq 0$. Since the sum of angles around a vertex equals to $360°$, it follows that for each vertex $u \in V$, $\sum_{v \in N(u)} \alpha(u, v) = 4$, 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 $180° \cdot (p(f) - 2)$, where $p(f)$ denotes the total 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 total number of vertex angles in f, and, $E(f)$ the directed arcs of f in its counterclockwise traversal. If f is unbounded, the respective sum is increases by eight. It is known that two orthogonal drawings with the same number of bends at each edge have the same orthogonal representation.

There is a nice correspondence between the min-cost network flow formulation of Tamassia and the underlying orthogonal representation with minimum number of bends of the input plane graph. In the flow network, one can think that each unit of flow corresponds to a $90°$ angle. Then, the vertices (*vertex-nodes*, *sources*) supply four units of flow each, which have to be consumed by the faces (*face-nodes*, *sinks*). Each face f demands $2a(f) - 4$ units of flow (increased by eight if f is unbounded). The relation now seems clear. To maintain the properties described above each edge from a vertex-node to a face-node in the flow network is equipped with 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 introduces a respective unit cost. The total cost is actually the total number of bends along the respective edge. Hence, the min-cost flow solution corresponds to a representation of the plane graph with minimum total number of bends.

2.2 Modifying the Flow Network

We are now ready to present how to modify the algorithm of Tamassia, in order to obtain a slog representation of an input plane graph G with minimum number of half-bends. Recall that G contains two types of vertices, namely real and crossing vertices. Real (crossing, respectively) vertices use orthogonal (diagonal, respectively) ports. Observe that a pair of half-bends on an rr-edge of a slog drawing corresponds to a bend of an orthogonal drawing. The same holds for half-bends on cc-edges. However, an rc-edge must switch from an orthogonal port (incident to the r-vertex) to a diagonal port (incident to the c-vertex). This implies that each rc-edge has at least one half-bend.

Consider an rc-edge (v_r, v_c) incident to faces f and g (see Figure 2) and

(a) Zero units of flow

(b) One unit of flow

Figure 2: Two configurations corresponding to zero or one unit of flow over an rc-edge; f and g are the two adjacent faces.

assume that the port of real vertex v_r is fixed. Depending on the (diagonal) port on the crossing vertex v_c we obtain two different representations with the same total 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, i.e., through this edge just one unit of flow can be transmitted and this is for free. Hence, the first half-bend of each rc-edge is free of cost, as desired. 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 on the bend before the crossing; otherwise a right turn, as illustrated in Figures 2(a) and 2(b).

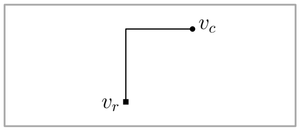
2.3 Proof of Correctness and Properties of Bend-Optimal Slanted Orthogonal Representations

In this section we present 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 that respects the embedding of G is always feasible. Then, we present an upper bound for the number of half-bends in optimal slog representations. In the following we assume that, together with a planarization, the embedding is also given.

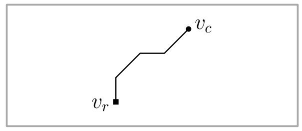
Theorem 1 *For a planarized graph G of maximum degree 4, we can efficiently compute a slog representation with minimum number of half-bends respecting the embedding of G.*

Proof: The idea is to use a reduction to Tamassia's network flow algorithm. In particular, since the original flow network algorithm computes a (bend-optimal) orthogonal representation for the input plane graph, our algorithm will also compute a slog representation. In the following, we prove that this representation is also bend-optimal.

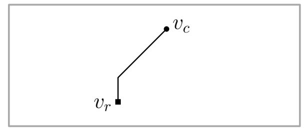
Assume that we are given an orthogonal representation F. We can uniquely convert F into a slog representation $S(F)$ by turning all crossing vertices counterclockwise by 45°. More precisely, the last segment of every rc-edge before the crossing vertex will become a left half-bend. Furthermore, every orthogonal



(a) Orthogonal input



(b) Three half-bends



(c) Reduction by 2 half-bends

Figure 3: A case in which two half-bends can be eliminated.

bend is converted into two half-bends, bending in the same direction as the orthogonal bend (see Figures 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 Figure 3). In this case, only the first 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 also be reversed such that for each slog representation S, we can construct 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. If this is not the case, then one has to deal with staircases of subsequent bends; a case that cannot occur in 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 corresponding bend-minimal orthogonal representation.*

Proof: Let R_s^{opt} be the bend-optimal slog representation and R_o^{opt} be the bend-optimal orthogonal representation. Given a representation R (either slog or orthogonal), we denote by $hb(R)$ the number of half-bends and by $b(R)$ the number of bends of R. Assume for a proof by contradiction that it holds that $hb(R_s^{opt}) < 2b(R_o^{opt})$. We will show how to construct an orthogonal representation R'_o that has less bends than R_o^{opt}.

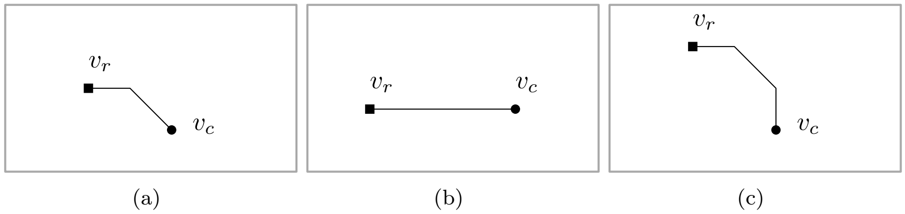


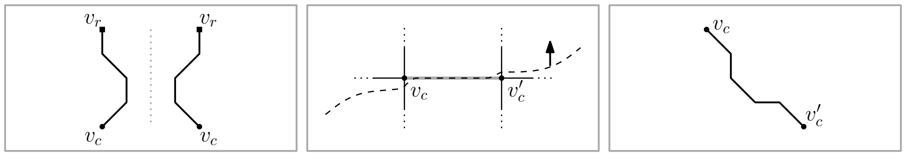
Figure 4: Illustration for the proof of Theorem 2: (a) An rc-edge with a left-turn from v_c to v_r. (b) Rotating c-vertex v_c by 45° to the left saves one half-bend. (c) Rotating c-vertex v_c by 45° to the right increases the number of half-bends.

To transform a slog into an orthogonal representation the diagonal ports on the c-vertices have to become orthogonal. To this end, we define the notion of *rotating* a vertex to the left or right, which means that the ports assigned to the edges incident to it are turned cyclical into the respective direction and the number of bends on the edges changes according to Figure 4. For rc-edges, we always do so from the crossing to the real vertex. Note that if there is exactly one half-bend to the left from a crossing-vertex v_c to a real-vertex v_r (see Figure 4(a)) and v_c is rotated by 45° to the left, the resulting drawing has zero bends on this edge (see Figure 4(b)); a rotation by 45° to the right would result in two half-bends (see Figure 4(c)).

By construction, all rr- and cc-edges have an even number of half-bends. In order to obtain R'_o, we replace each pair of half-bends on rr- and cc-edges of R_s^{opt} by an orthogonal bend (as in Figures 1(d) and 1(c)). So, for rr- and cc-edges it holds that $hb(R_s^{opt} \setminus \{\text{rc-edges}\}) = 2b(R'_o \setminus \{\text{rc-edges}\})$.

On the other hand, all rc-edges have an odd number of half-bends. Similarly to the case of rr- and cc-edges, we replace each pair of half-bends except the one half-bend closest to the crossing vertex by an orthogonal bend in R'_o. Let $\mathcal{C} = \{C_1, \ldots, C_l\}$ be the set of maximal connected components consisting only of c-vertices and the cc-edges between them. By maximality, it follows that $C_i \cap C_j = \emptyset$, for $i \neq j$. Also, observe that if $l = 0$, then we already would have a contradiction. Therefore, $l > 0$ must hold. Now, let $lhb(C_i)$ be the number of left half-bends and $rhb(C_i)$ be the number of right half-bends on the rc-edges connected to C_i that have not been replaced already. If $lhb(C_i) = rhb(C_i)$ for all $i = 1, 2, \ldots, l$, then without loss of generality we rotate all vertices in C_i to the left. This implies that all left half-bends disappear and the right half-bends get replaced by orthogonal bends, which contradicts the assumption that R_o^{opt} is bend-optimal.

So, there has to be at least one C_i with $lhb(C_i) \neq rhb(C_i)$. Assume $lhb(C_i) > rhb(C_i)$ for some $i = 1, 2, \ldots, l$. If we now rotate C_i to the left, then we obtain an orthogonal solution that has even less bends than twice the number of half-bends of the slog solution. Similarly, we can obtain a better orthogonal solution when $lhb(C_i) > rhb(C_i)$ by rotating C_i to the right.



(a) The spoon gadget (b) Cut through (v_c, v_c') (c) 4 half-bends are needed

Figure 5: (a) Illustration of the spoon gadget. (b) The orthogonal input can be transformed into a slog drawing when everything above the dashed cut is moved up. (c) The result contains 4 half-bends.

By this construction, it holds that $2b(R_o') \leq hb(R_s^{opt}) < 2b(R_o^{opt})$, which is a contradiction to the assumption that R_o^{opt} is optimal. □

3 A Heuristic to Compute Close-to-Optimal Slanted Orthogonal Drawings

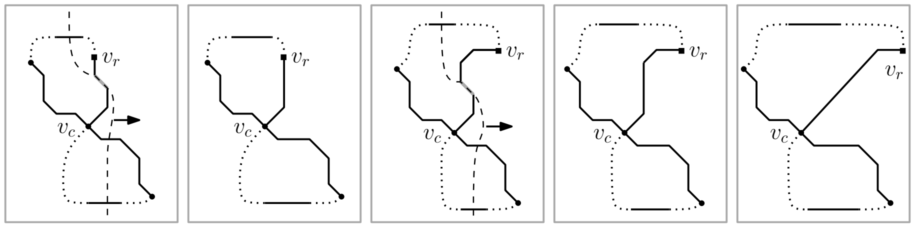
In this section, we present a heuristic which, given an optimal slog representation, computes an actual drawing, which is close-to-optimal with respect to the total number of bends and requires quadratic area. This is a quite reasonable approach, since insisting on optimal slog drawings may result in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

Algorithm 1: SPOON BASED ALGORITHM

Input : A slog representation S of a given plane graph G.
Output: A slog drawing $\Gamma_s(G)$.

S1: Compute an orthogonal drawing $\Gamma(G)$ based on S
S2: Replace each orthogonal bend by 2 half-bends {see Figs.1(c) and 1(d)}
S3: Fix ports on rc-edges using the spoon gadget {see Fig.5(a)}
S4: Apply cuts to fix ports on cc-edges {see Figs.5(b) and 5(c)}
S5: Optimize the number of rc half-bends {see Figs.6(a) and 6(b)}
S6: Optimize the number of cc half-bends {see Figs.7(a), 7(b) and 7(c)}
S7: Heuristically compact the drawing (as post-processing)

In Step 1 of Algorithm 1, we compute an orthogonal drawing $\Gamma(G)$ based on the input slog representation. If there is flow on an edge e connecting faces f_i and f_j that we added to Tamassia's model, we treat it as if it was flow on the other edge connecting f_i and f_j that was part of the flow network of the original algorithm. With this we get a flow that is still valid and corresponds to an orthogonal representation for which the algorithm of Tamassia [13] can compute a drawing. In the next step, we replace all orthogonal bends with pairs



(a) (b) (c) (d) (e)

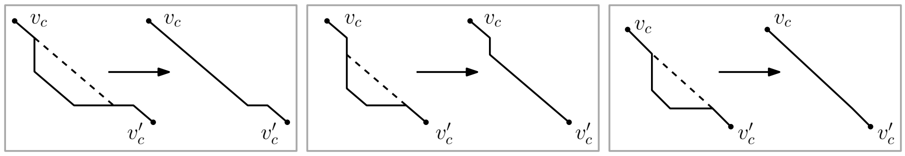
Figure 6: Saving bends on rc-edges: (a) A vertical cut through a bend-less rc-edge results in (b) a reduction by two half-bends. (c) Similarly, a vertical cut through a bent rc-edge also results in (d) a reduction by two half-bends. (d) However, the optimal may require four half-bends reduction.

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 of each rc-edge by a gadget, which we call *spoon* due to its shape (see Figure 5(a)). This gadget allows us to switch between orthogonal and diagonal ports on an edge. Note that the input slog representation specifies the ports on all vertices, thereby defining which configuration is used.

In order to fix the ports of cc-edges (which still use orthogonal ports), we employ appropriate cuts[2] (Step 4 of Algorithm 1). A *cut*, for us, is either (i) an x-monotone continuous curve that crosses only vertical segments and divides the current drawing into a top and a bottom part (*horizontal cut*), or, (ii) a y-monotone continuous curve that crosses only horizontal segments and divides the current drawing into a left and a right part (*vertical cut*). Observe that in order to apply a horizontal (vertical, respectively) cut, we have to ensure that each edge crossed by the cut has at least one vertical (horizontal, respectively) segment. This holds before the introduction of the spoons, as $\Gamma(G)$ is an orthogonal drawing. We claim that this also holds when all spoons are present. This is because a spoon replacing a horizontal (vertical, respectively) segment has two horizontal (vertical, respectively) segments.

To fix a horizontal cc-edge (v_c, v_c') with v_c being to the left of v_c' in the drawing, we first momentarily remove this edge from the drawing. Then we use a horizontal cut which from left to right passes exclusively through vertical segments. There exist two options for such a cut. The first one starts in the outer face and continues up to the face below (v_c, v_c'), then to the face above and from there again to the outer face. The second one again starts in the outer face and continues up to the face above (v_c, v_c'), then to the face below and from there to the outer face (see Figure 5(b)). Our choice depends on the input slog representation that specifies the ports on each crossing vertex. The result of such a cut is depicted in Figure 5(c) and has a new horizontal and a new vertical segment that replaces edge (v_c, v_c'). The first (second, respectively) one is

[2] A *cut* is a standard tool to perform stretchings in orthogonal drawings, see e.g. [5].



(a) Reduction by 2 half-bends (b) Reduction by 2 half-bends (c) Reduction by 4 half-bends

Figure 7: Saving bends on cc-edges by a local operation.

necessary for potential future vertical (horizontal, respectively) cuts. Similarly, we cope with cc-edges with bends by applying the same technique only to the first and last segments of the edge.

The resulting slog drawing has two additional half-bends for each rc-edge (the spoon gadget adds three half-bends; one is required) and four additional half-bends for each cc-edge (none is required), with respect to the ones suggested by the input representation. With similar cuts as the ones described above, we can save two half-bends for each rc-edge, by eliminating the diagonal segment of the spoon gadget (Step 5 of Algorithm 1). Our approach is illustrated in Figures 6(a) and 6(b). Observe that in this case the cut simply requires the removal of the diagonal segment that is to be eliminated and not the whole edge. The result is optimal for bend-less rc-edges (see Figure 6(b)). However, for rc-edges with bends (see Figure 6(c)), our approach guarantees two half-bends reduction (see Figure 6(d)), while in the optimal case four half-bends could be removed (see Figure 6(e)). Observe that the rectilinear segments of the edge are not affected, in order to be able to apply future cuts.

As already stated, each cc-edge admits four additional half-bends (none is required). It is always possible to remove two of them (Step 6 of Algorithm 1) by applying a local modification as depicted in Figure 7. If for example the horizontal part of such an edge is longer than the vertical one, a shortcut like the one in the left part of Figure 7(a) can be applied. Note that this operation does not require any cuts. If the horizontal and the vertical segments of the cc-edge have the same length, then all four half-bends can be saved; see Figure 7(c).

Once the operations we described above are applied, the drawing will contain zero additional half-bends on rr-edges and bend-less rc-edges and at most two additional half-bends on each cc-edge and each rc-edge with bends, with respect to the input representation. Note that in order to apply our technique we need to scale up the initial drawing by a factor of 5 at the beginning of our algorithm, to provide enough space for additional half-bends. In subsequent steps, the cuts increase the drawing area. However, since each cut implies a constant factor increment to the drawing area and each edge yields at most one cut, the total drawing area asymptotically remains quadratic. The following theorem summarizes our approach.

Theorem 3 *Given a slog representation of a planarized graph G of maximum degree 4, we can efficiently compute a slog drawing requiring $O(n^2)$ area with* *(i) optimal number of half-bends on rr- and bend-less rc-edges and (ii) at most two additional half-bends on cc edges and rc-edges with bends.*

Note that the scaling of the drawing by a factor of five in Step 2 of Algorithm 1 does not asymptotically affect the drawing area; in practice, however, it has negative effects. This motivated us to heuristically further compact the drawing at the cost of some extra bends (as a post-processing; Step 7 of Algorithm 1). First, we enrich all diagonal segments that are of a certain length by a new horizontal and a new vertical segment, so that the remaining diagonal segment is of unit length. To ensure planarity, we apply a rectangular decomposition similar to the one of Tamassia [13] and then we contract along horizontal and vertical cuts. Finally, we remove unnecessary half-bends similarly to Step 6 of Algorithm 1 (see also Figure 7).

4 A Linear Program To Compute Optimal Drawings

In this section, we develop a Linear Program (LP) which, given an optimal slog representation S of a plane graph G, computes an actual drawing $\Gamma(G)$, which is optimal with respect to the total number of bends; if one exists. Before we proceed with the description of our linear program, we mention that 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 linear program.

4.1 The Core of the Linear Program

Initially, we appropriately augment graph G and obtain a new graph that is a subdivision of G and has at most one half-bend on each edge. More precisely, let (u, v) be an edge of G with more than two half-bends (as defined by the slog representation S). Let $\langle b_1, b_2, \ldots, b_k \rangle$, $k \geq 2$, be the half-bends of edge (u, v) and assume without loss of generality that $b_1, b_2, \ldots, b_k$ appear in this order along the edge (u, v), when traversing (u, v) from vertex u towards vertex v. We first consider the case where vertex u is a real vertex. In this case, we add a new crossing vertex w in G and then we replace the edge (u, v) of G with the edges (u, w) and (w, v). The first half-bend b_1 of the edge (u, v) is assigned to the edge (u, w), while the remaining half-bends $\langle b_2, \ldots, b_k \rangle$ of the edge (u, v) are assigned to the edge (w, v). The case where vertex u is a crossing vertex is treated analogously, with the only exception that in this particular case vertex w would have been a real vertex. If we apply the procedure that we just described on each edge of G with more than two half-bends (as long as there exist such edges), then we will obtain an augmented graph, say G_{aug}, that is clearly a subdivision of G and has at most one half-bend on each edge, as desired. Furthermore, neither the type of each new vertex nor its ports are arbitrarily chosen, as they depend on the types of its incident segments given

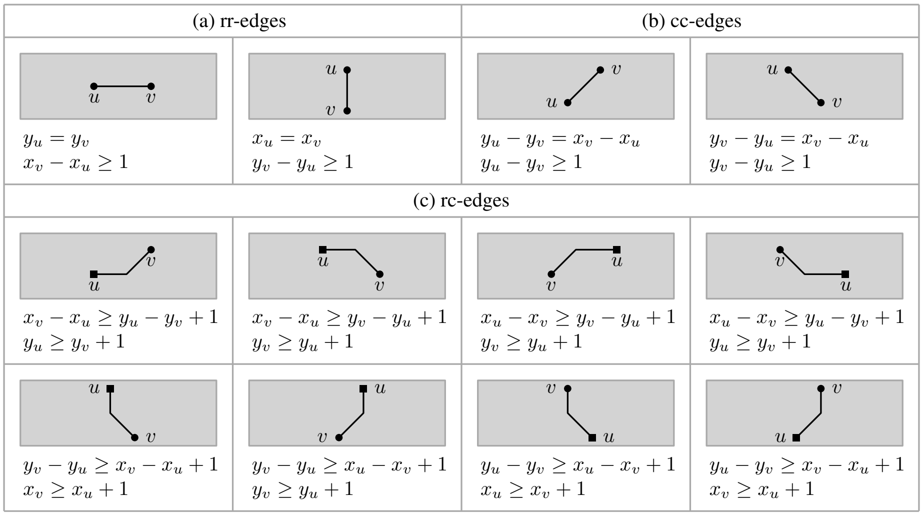


Figure 8: The list of constraints used by the linear program for (a) rr-edges, (b) cc-edges and (c) rc-edges, assuming that the y-axis points downwards.

by the input representation S (either orthogonal or diagonal). This implies a new slog representation, say S_{aug}, for G_{aug}.

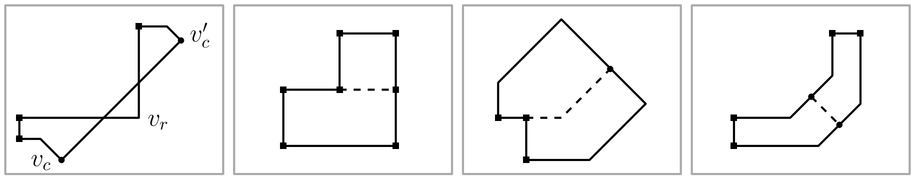
Now observe that each face f of G has a corresponding face f' in G_{aug} such that: (i) the vertices of G_{aug} incident to face f' are the same as the ones incident to face f of G, plus the ones from the subdivision; and (ii) the sequence of slopes assigned to the segments bounding f' is the same as the ones of the segments bounding f in G. Hence, a drawing $\Gamma(G_{aug})$ of G_{aug} realizing the slog representation S_{aug} is also a drawing $\Gamma(G)$ of G realizing the slog representation S, where subdivided edges are routed as their corresponding paths in G_{aug}.

We are now ready to describe our linear program, which computes a drawing $\Gamma(G_{aug})$ of G_{aug} realizing the slog representation S_{aug}. For each vertex u of G_{aug}, we introduce a pair of variables x_u and y_u that corresponds to the coordinates of vertex u on the plane. Then, for each edge (u, v) of G_{aug}, we define a pair of constraints, depending on the type of vertices u and v (i.e., real or crossing vertices). The detailed list of constraints is given in Figure 8.

In order to obtain "compact" drawings, we indirectly minimize the area by minimizing the total edge length. In particular, this is our objective function. Note that the slopes of the segments allow us to express the Euclidean length of each edge as a linear function. As an example, the length of the edge depicted in the first cell of Figure 8c is defined as $(\sqrt{2} - 1) \cdot (y_u - y_v) + x_v - x_u$.

4.2 Addressing Planarity Issues

The linear program, as described so far, models the shape of the edges (and subsequently the shape of the faces) and the relative positions between pairs of



(a) A non-planar face (b) Split-edge (vertex) (c) Split-edge (vertex) with half-bend (d) Split-edge (bend)

Figure 9: In all figures, real (crossing, respectively) vertices are drawn as squares (disks, respectively); split-edges are drawn dashed.

adjacent vertices. Since there are no constraints among non-adjacent vertices, it is highly possible that the resulting drawing is non-planar. We provide an example in Figure 9(a), where the relative positions between vertices (*i*) v_r and v_c, and, (*ii*) v_r and v_c' are not defined by the liner program, yielding to a (potential) crossing situation. To cope with this problem, unfortunately, we cannot follow an approach similar to the one that Tamassia suggests in his original algorithm (i.e., he "splits" all non-rectangular faces into rectangular ones), since in our case a face is not necessarily rectilinear.

In order to describe our approach to ensure that each face is drawn planar, we first introduce some necessary terminology. We distinguish two types of corners of a face in a slog representation; *vertex-corners* (or simply vertices) and *bend-corners* (or simply bends). With respect to a face, a corner is either *convex*, if the inner angle is $\leq 135°$, or *non-convex* otherwise[3]. Hence, there are four possible types of corners in total: convex vertex-corner, convex bend-corner, non-convex vertex-corner, non-convex bend-corner. The *configuration* of a corner describes the shape of the corner by the pair of orientations of its two incident segments in the order they are visited by a counterclockwise traversal of the corresponding face. Possible *orientations* are horizontal (h), vertical (v), diagonal-up (du), and diagonal-down (dd). For example, the configuration of the bend-corner incident to segments s' and s'' of Figure 10(a) is given by *dd-h*. The *type* of a configuration describes the corresponding corner in a more general way by just distinguishing between orthogonal (o) or diagonal (d) orientations. In the example of Figure 10(a), the configuration of the bend-corner incident to segments s' and s'' is of type *d-o*. We next define the notions of a *split-edge* and an *almost-convex face*, that are both central in our approach.

Definition 2 *For a given face f, a* split-edge *is an edge that:*

- *is bend-less and connects a non-convex vertex-corner v with a new vertex that we introduce by subdividing a side parallel to one of the edges incident to v (see Figure 9(b)).*

[3]We ignore vertices and bends on corners that form 180° angles, since by construction they are always aligned with their neighbors.

- *or, has a half-bend and connects a non-convex vertex-corner v with a new vertex that we introduce by subdividing a diagonal side of f (see Figure 9(c)).*
- *or, is a bend-less edge that connects two new vertices that we introduce by subdividing two parallel edges, when one of them is incident to a non-convex bend-corner (see Figure 9(d)).*

Definition 3 *A face is* almost-convex *if it does not contain any non-convex vertex-corners and no split-edge exists that separates the face into two non-convex faces.*

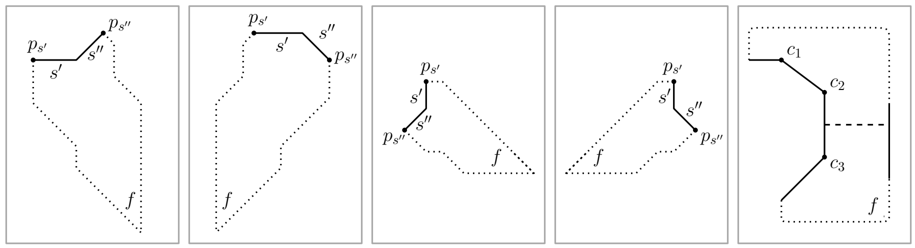
First, we make all faces almost-convex (by further augmenting our graph). Later, we will show that the linear program will always compute a planar drawing if all faces are almost-convex.

A non-convex vertex-corner is *eliminated* by introducing a new split-edge (corresponding to new constraints in the linear program) as shown in Figure 9(b). When there is no parallel side to one of the segments incident to the vertex-corner, we introduce a split-edge with a half-bend, as illustrated in Figure 9(c). It is important to note that the elimination of a non-convex vertex-corner does not introduce new ones. Hence, all of them can be eliminated sequentially by appropriately adopting one of the two approaches described above.

In order to eliminate a non-convex bend-corner of a face that is not almost-convex, we search for a split-edge (again corresponding to new constraints in the linear program) that yields two non-convex faces. Such a split-edge is illustrated in Figure 9(d). We will appropriately introduce such split-edges until all faces are almost-convex (without introducing non-convex vertex-corners). To prove that it is always feasible to make all faces almost-convex, we give the following lemma.

Lemma 1 *Let s' and s'' be two segments of a face f incident to a non-convex bend-corner. Face f contains a segment $s \notin \{s', s''\}$ that is parallel to either s' or s''.*

Proof: For a proof by contradiction, we assume that there is no segment of face f parallel to s' and s''. Without loss of generality, we further assume that s' is a horizontal segment and s'' is a diagonal segment of positive slope; see Figure 10(a). The cases, where s' is a vertical segment and/or s'' is a diagonal segment of negative slope, are analogous (see Figures 10(b), 10(c), and 10(d)). Let $p_{s'}$ and $p_{s''}$ be the end-points of segments s' and s'', respectively, which are not identified with the non-convex bend-corner incident to both s' and s''. Since f is a face, there exists a polygonal chain of segments of f connecting $p_{s'}$ and $p_{s''}$. In our drawing model, such a chain consists of horizontal, vertical and diagonal segments. Now observe that a horizontal or a positively-sloped diagonal segment of the chain connecting $p_{s'}$ and $p_{s''}$ is parallel to s' or s'', respectively, which contradicts our initial assumption that there is no segment of face f parallel to s' and s''. Hence, the polygonal chain connecting $p_{s'}$ and $p_{s''}$ consists of vertical



(a) (b) (c) (d) (e)

Figure 10: (a)-(d) Different configurations used in the proof of Lemma 1. (e) Configuration used in the proof of Lemma 2.

and negatively-sloped diagonal segments, which is a contradiction since $p_{s'}$ and $p_{s''}$ cannot be connected by such a chain, without forming an angle of 45° at a corner of f (a situation that is not allowed by our drawing model). □

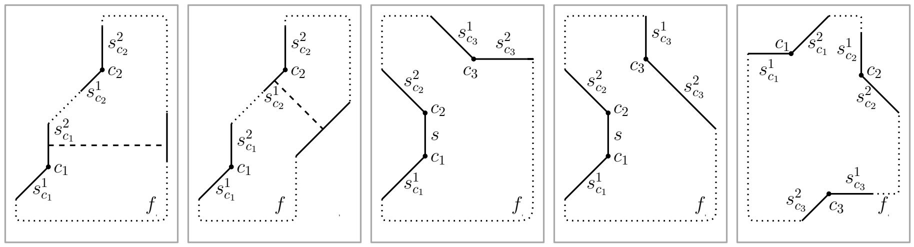
From Lemma 1, it follows that, for a non-convex bend-corner of a face f, there is a split-edge emanating from one of its incident segments towards to a parallel segment of face f. If f is not almost-convex (and contains no convex vertex-corners) and this edge is carefully selected such that it yields exactly two non-convex "subfaces", say f' and f'', of face f, then it is not difficult to see that both f' and f'' have fewer non-convex bend-corners than f. In addition, no convex vertex-corners are introduced. This implies that if one recursively applies this procedure to f' and/or f'' (if either of these is not almost-convex), f will eventually be split into a particular number of "subfaces" that are all almost-convex. In addition, it is not difficult to see that all additional edges, that are required to make all faces almost-convex can be expressed by using the original set of constraints of our linear program. So, it now remains to prove that almost-convex faces are drawn planar. To do so, we give the following lemmas.

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

Proof: Assume to the contrary that f has three consecutive non-convex bend-corners, say c_1, c_2 and c_3; see Figure 10(e). Assume that c_1, c_2 and c_3 appear in this order in the counterclockwise traversal of face f. By Lemma 1, there exists a segment of f that is parallel to one of the segments incident to c_2. This implies that, there exists a split-edge that partitions f into two non-convex faces; one containing c_1 and one containing c_3, which is a contradiction since f is almost-convex. □

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

Proof: In the proof, we use the notion of a configuration. More precisely, we assume to the contrary that an almost-convex face f contains at least three



(a) (b) (c) (d) (e)

Figure 11: Different configurations used in the proof of Lemma 3.

non-convex bend-corners c_1, c_2 and c_3 and distinguish four cases. In our case analysis, we denote by $s^1_{c_i}$ and $s^2_{c_i}$ the segments incident to corner c_i and assume the $s^1_{c_i}$ precedes $s^2_{c_i}$ in the clockwise traversal of face f, $i = 1, 2, 3$.

Case 1: *Two of these non-convex bend-corners have the same configuration*; see Figure 11(a) or 11(b) for an illustration. By Lemma 1, there exists a parallel segment to either $s^1_{c_1}$ or $s^2_{c_1}$, and thereby to either $s^1_{c_2}$ or $s^2_{c_2}$. In both cases, one of the split-edges separates c_1 from c_2, so that the resulting faces are both non-convex. Hence, f is not almost-convex; a contradiction. So, in the following cases we assume that c_1, c_2 and c_3 are of different configurations.

Case 2: *Corners c_1 and c_2 are consecutive corners of f and the first segment of c_3 is parallel to the second segment of c_2*; see Figure 11(c) for an illustration. We denote by s the segment that is incident to both c_1 and c_2 (i.e., $s = s^2_{c_1} = s^1_{c_2}$). From the previous case it follows that c_1 and c_3 are of different configurations. In order to close the face there has to be a segment that is parallel to either s or $s^2_{c_2}$ that is not $s^1_{c_3}$, thereby allowing a split-edge that separates either c_1 from c_2 and c_3, or, c_1 and c_2 from c_3. The resulting faces are both non-convex. Hence, f is not almost-convex; a contradiction.

Case 3: *Corners c_1 and c_2 are consecutive and the second segment of c_3 is parallel to the second segment of c_2*; see Figure 11(d) for an illustration. Again, we denote by s the segment that is incident to both c_1 and c_2 (i.e., $s = s^2_{c_1} = s^1_{c_2}$) and assume that c_1 and c_3 are of opposite configurations. In this case there is a split-edge between segments $s^2_{c_2}$ and $s^2_{c_3}$ thereby separating c_1 and c_2 from c_3 and resulting in two non-convex faces. Hence, f is not almost-convex; a contradiction.

Case 4: *Corners c_1, c_2 and c_3 are pairwise non-consecutive*; see Figure 11(e) for an illustration. Since there are only two types of diagonals, at least two non-convex corners, say c_1 and c_3, are of the same type. Since they are forced to have opposite configurations (*d-o* or *o-d*) a split-edge between those two parallel diagonals would separate the two respective corners, resulting in two non-convex faces. Hence, f is not almost-convex; a contradiction.

The proof is completed by the observation that one of these four cases will always apply to every almost-convex face with more than two non-convex bend-corners. □

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

Proof: Let f be an almost-convex face. By Lemma 3, face f has at most two non-convex bend corners.

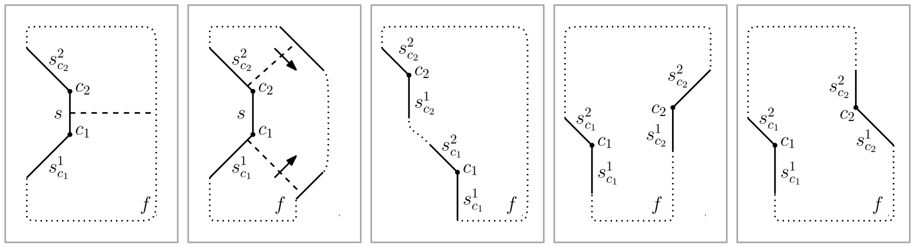
We claim that in the case where f has exactly one non-convex bend-corner, f is drawn planar. In fact, since completely convex faces are drawn planar by construction, we know that, if there is exactly one non-convex bend-corner c and f is not drawn planar, then one of the two segments s_c^1 and s_c^2 incident to c must be involved in a crossing. All the other vertex- and bend-corners of f are convex or collinear by construction. Let s be the segment that crosses s_c^1 or s_c^2. Since s is only adjacent to convex corners or 180° corners, it is not possible to close f without violating the port assignments as given by the representation. So, our claim holds.

Consider now the more interesting case where face f has exactly two non-convex bend-corners, say c_1 and c_2. We denote by $s_{c_i}^1$ and $s_{c_i}^2$ the segments incident to corner c_i and assume the $s_{c_i}^1$ precedes $s_{c_i}^2$ in the clockwise traversal of face f, $i = 1, 2$. We distinguish the following cases:

Case 1: *Corners c_1 and c_2 are consecutive*; see Figure 12(a) for an illustration. In this case, there is a segment, say s, that is incident to both c_1 and c_2 (i.e., $s = s_{c_1}^2 = s_{c_2}^1$). If there is a segment of f parallel to s, then there exists a split-edge separating f into two non-convex subfaces; one containing c_1 and one containing c_2 (see Figure 12(a)). Hence, f is not almost-convex. It follows that there is no segment of f that is parallel to s. By Lemma 1, there exist parallel segments to the other two segments that are incident to c_1 and c_2 (see Figure 12(b)). However, since f is almost-convex, a "split-edge" connecting the respective parallel segments would result in at least one convex face. We can move these "split-edges" arbitrary close to c_1 and c_2, so that they separate f into three convex regions. Since convex regions are drawn convex and hence planar by definition, no crossing can occur.

Case 2: *Corners c_1 and c_2 have the same configuration and orientation*; see Figure 12(c) for an illustration. This particular case is identical to Case 1 of Lemma 3 and therefore cannot occur.

Case 3: *Corners c_1 and c_2 have opposite configuration (meaning that they are made of the same orthogonal and diagonal part but in different orders) and orientation*; see Figure 12(d) for an illustration. Since the number of crossings that occur has to be even (otherwise ports would be violated),



(a) (b) (c) (d) (e)

Figure 12: Different configurations used in the proof of Lemma 4.

and the only way to have two crossings requires that one of the convex regions is drawn non-convex, this situation cannot introduce any crossings.

Case 4: *Corners c_1 and c_2 have the same configuration but opposite orientations*; see Figure 12(e) for an illustration. In this case, it is not difficult to see that there exists a split-edge between the two orthogonal or the two diagonal segments incident to c_1 and c_2, separating them into two non-convex subfaces, so f cannot be almost-convex.

The proof is completed by the observation that one of these four cases will always apply to an almost-convex face with exactly two non-convex bend-corners. □

5 Area Bounds

Slog drawings have aesthetic appeal and seem to improve the readability of non-planar graphs, when compared to traditional orthogonal drawings. However, in this section we show that such drawings may require increased drawing area. Note that most of the known orthogonal drawing algorithms require $O(n) \times O(n)$ area. The situation is different if one insists on slog drawings of optimal number of bends. As the following theorem asserts, 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 graph G is given.*

Proof: The planarized version $\sigma(G)$ of G is given in Figure 13(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 with 4 internal chords. Consecutive layers L_{i-1} and L_i, $i = 1, 2, \ldots, n$, are connected by 8 edges which together with the chords of layer L_i define 12 crossings. Hence, G consists of $20n + 9$ vertices and $32n + 12$ 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

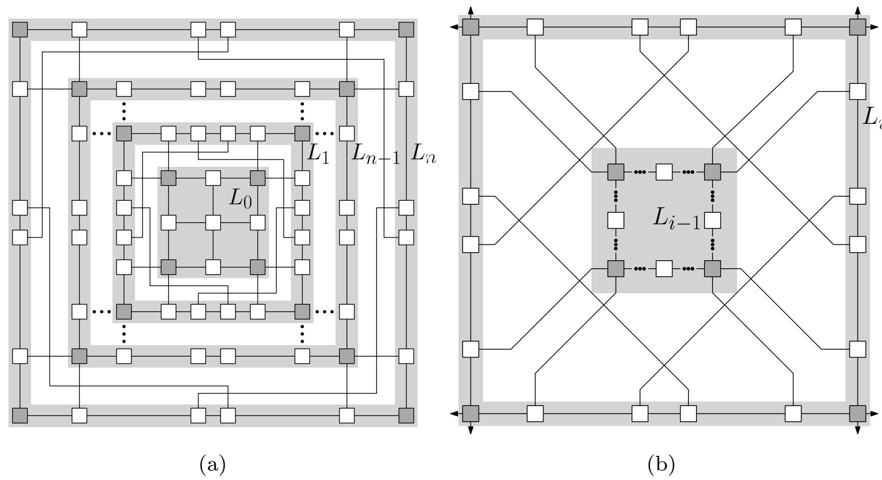


Figure 13: (a) A planarized version $\sigma(G)$ of a graph G. (b) Edges involved in crossings in $\sigma(G)$ contribute two half-bends.

four vertices, that have two ports pointing to the next layer (gray-colored in Figure 13(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 Figure 13(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 Figure 13(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). □

6 Experimental Evaluation

In this section, we present an experimental evaluation of our model. We compare classic orthogonal drawings obtained with the implementation of the original Tamassia algorithm [13] of the yFiles library (http://www.yworks.com) with bend-optimal slog drawings and drawings computed by the heuristic presented in Section 3. As a test set, we used the Rome graphs (obtained from http://www.graphdrawing.org) which are approximately 11.500 graphs. We filtered them for connected graphs with maximal degree 4, which left 1.122 graphs. Of this 1.122 graphs, 1.039 were planar and 83 non-planar. The average den-

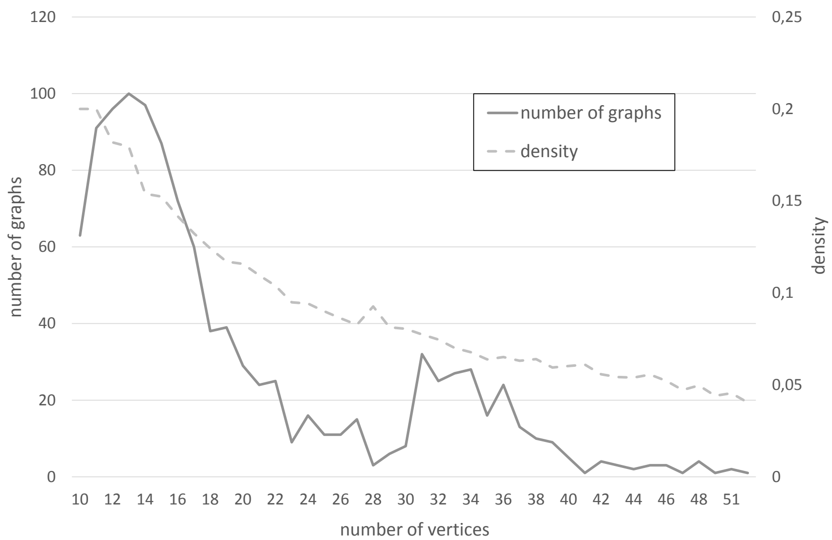


Figure 14: Number of instances and density of the test set.

sity over all graphs was 0, 14; recall that the density of a graph is defined as the ratio of the number of its edges to the maximum possible number of edges. The number of vertices ranged from 10 to 56. Figure 14 gives a more detailed description of the test set: the number of instances and their average density are plotted against the number of vertices of the test set.

We ran our experiments on a Linux machine with four cores at 2, 5 GHz and 3 GB of RAM. All implementations were done in Java using the yFiles library. For solving the linear programs, we used the Gurobi solver [8].

To obtain an input for our algorithms, we computed an embedding with the *Combinatorial Embedder* from the yFiles library, which guarantees a planar embedding for planar instances. In all following plots, the curve denoted by *orthogonal* stands for results for the orthogonal drawings, while the curves denoted by *slog* and *heuristic* correspond to the results for bend-optimal and heuristic slog-drawings. To obtain the actual numbers, the results for all graphs with the same number of vertices were averaged.

All results we present in this section were computed in less than 200 ms each, as depicted in the cpu-time chart in Figure 15. Apparently, the heuristic requires the most computation time. We observed that this is due to its last step, where the drawing is heuristically compacted. It seems that the computation of the cuts, which are required in order to reduce the area, is relatively time-consuming.

Of course, the graphs of our test set are rather small and not very dense. However, even for hand-crafted dense graphs with more than 400 vertices, the optimal slog drawings could be computed in less than 2 seconds, which suggests that our LP-formulation can be useful for practical applications. Note that

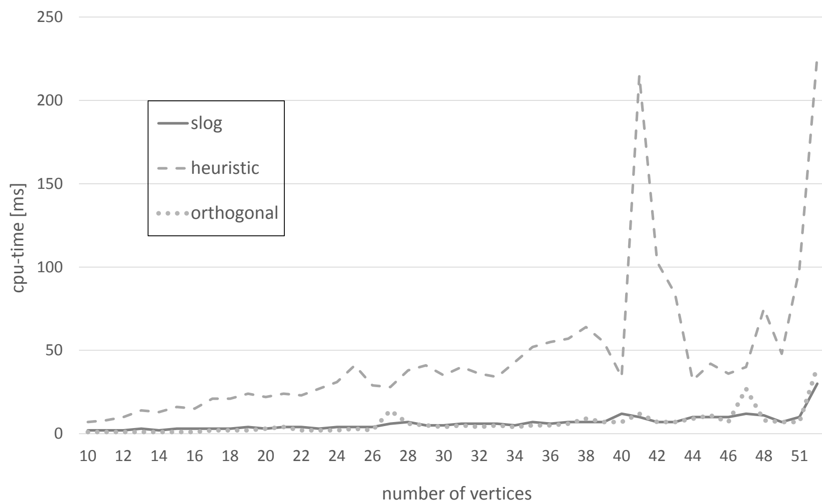


Figure 15: Number of vertices against cpu-time.

these hand-crafted graphs are not included in the experimental evaluation of this section; we simply used them in order to verify that the LP can still be solved within reasonable time even for large and dense input graphs.

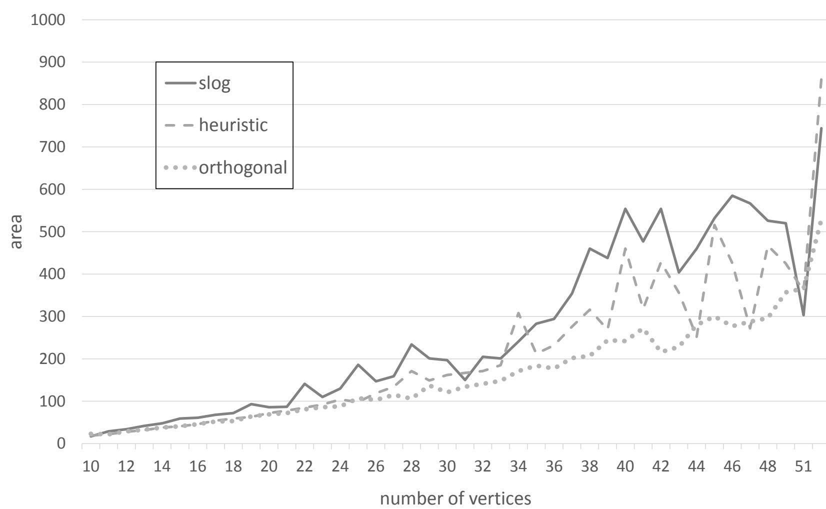


Figure 16: Number of vertices against area.

In Figure 16, the required area is plotted against the number of vertices. As expected, the area of the slog drawings is larger than the corresponding one of the orthogonal drawings. On the other hand, the bend-optimal slog drawings and the ones computed by the heuristic presented in Section 3 are of

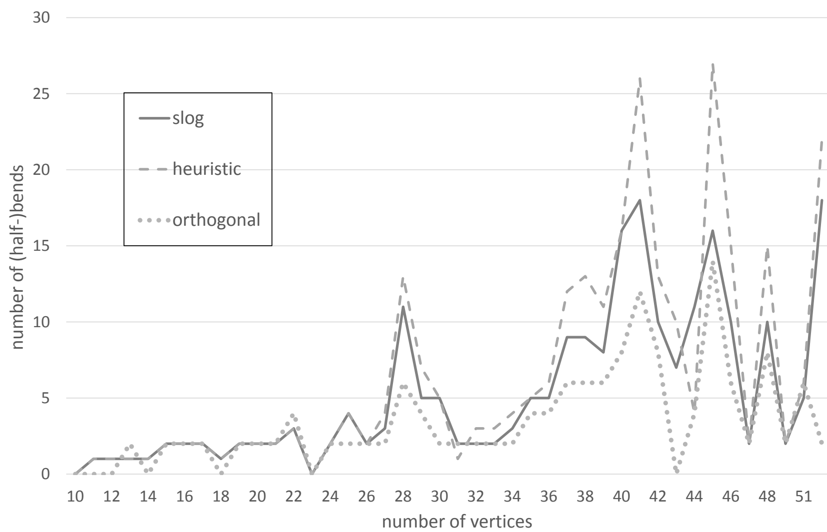


Figure 17: Number of vertices against number of bends.

comparable area, indicating that the minimization of the total edge length as an objective function of the linear program seems to be very effective. Recall that the orthogonal drawing which is used as an input in the heuristic is scaled up by a factor of five (which yields a factor of 25 in the total area). So, one would expect that the drawings computed by the heuristic would (in practice) require significantly more area than the corresponding orthogonal ones, which apparently is not evident in Figure 16. This is due to the last step of the heuristic, where the drawing area is reduced.

As stated in Section 2, the number of half-bends in the bend-optimal slog drawings is at least twice the number of bends in the bend-optimal orthogonal drawings. So, in Figure 17 we plotted twice the number of orthogonal bends against the number of half-bends produced by our algorithms. As expected, the orthogonal drawings require the least amount of bends. We measured that on average the bend-optimal slog drawings required 2.84 times more half-bends than the orthogonal drawings, while the drawings computed by the heuristic required 1.18 times more half-bends than the bend-optimal slog drawings. In actual numbers, the bend-optimal slog drawings require (on average) 5 more half-bends, while the drawings computed by the heuristic require 8 more half-bends than the corresponding orthogonal drawings.

Figure 18 shows the total edge length in relation to the number of vertices. In our experiments, we found that the plots of the total edge length are comparable to the plots of the area (Figure 16). This is exactly as expected, since the larger the area is the larger the total edge length is expected to be. When comparing the ratio of the longest to the shortest edge, again the orthogonal algorithm produced the smallest results, as can be seen in Figure 19. This is because the orthogonal drawings were the most compact ones. For the bend-optimal

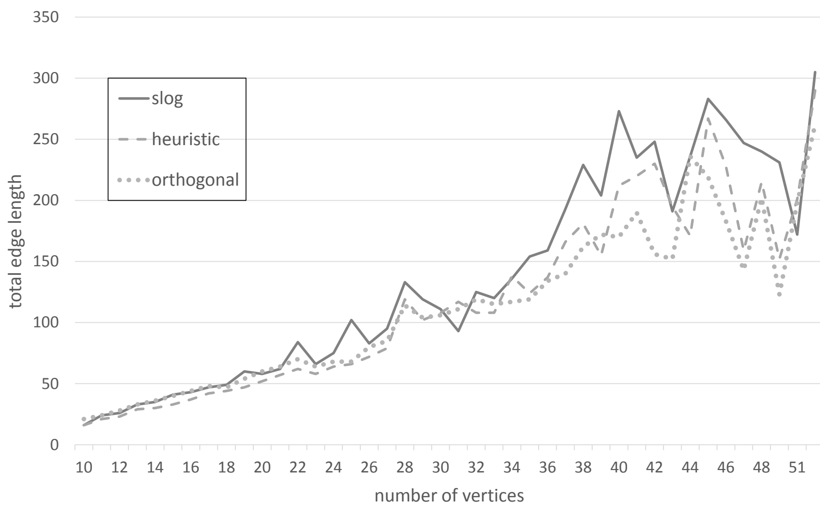


Figure 18: Number of vertices against total edge length.

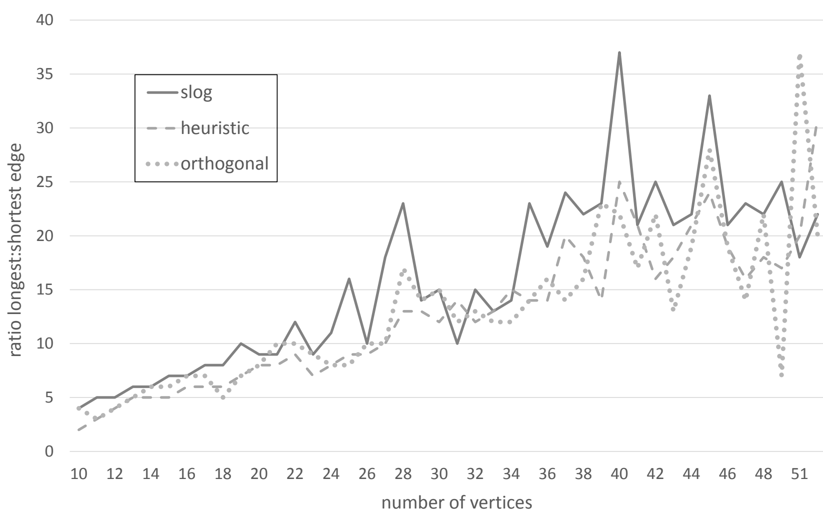


Figure 19: Number of vertices against ratio of longest to shortest edge.

slog drawings, this ratio went up to 37 in our experiments, while the heuristic had a better ratio between the longest and the shortest edge. Note that the high ratios observed in the bend-optimal slog drawings are caused by the long diagonal segments required in the slanted model.

7 Sample Drawings

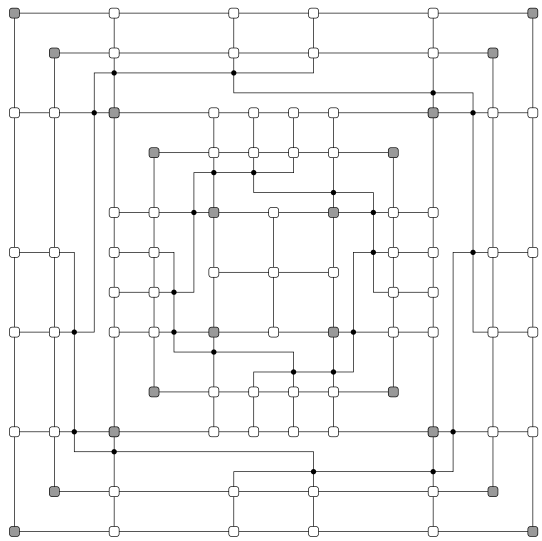


Figure 20: An orthogonal drawing of minimum number of bends for the graph of Figure 13 establishing the exponential area bound for slog drawings.

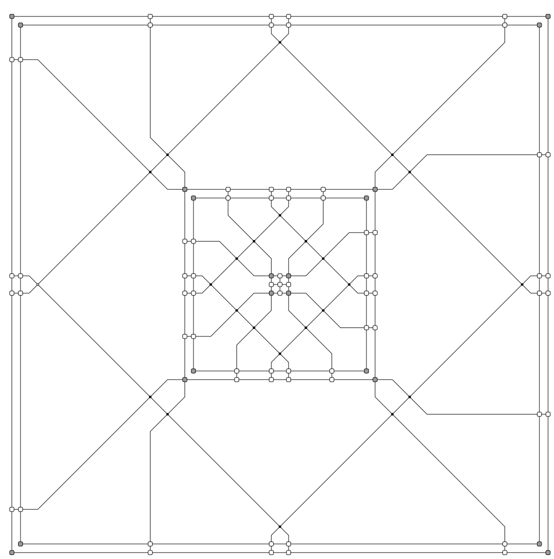


Figure 21: The bend-optimal slog drawing corresponding to the one of Figure 20.

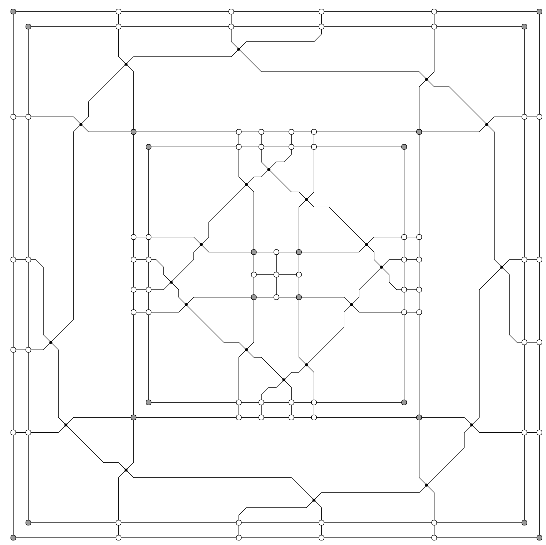


Figure 22: The close-to-optimal slog drawing (corresponding to the one of Figure 20) produced by our heuristic algorithm of Section 3 without Step 7 of Algorithm 1.

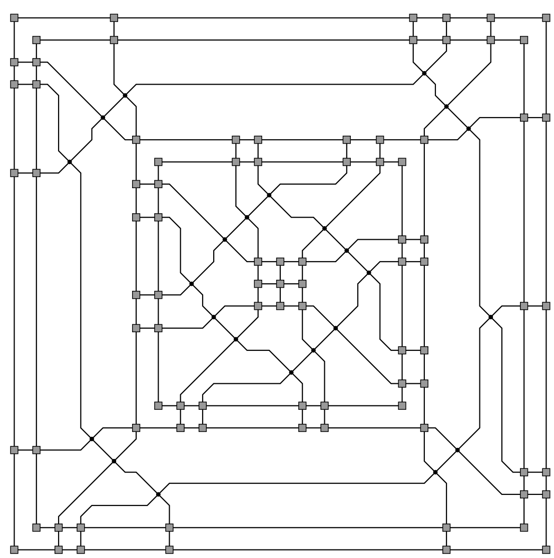


Figure 23: The close-to-optimal slog drawing (corresponding to the one of Figure 20) produced by our heuristic algorithm of Section 3 with Step 7 of Algorithm 1.

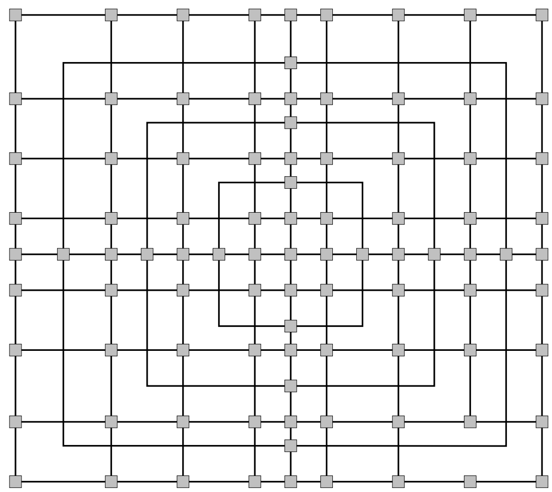


Figure 24: A highly symmetric non-planar orthogonal drawing.

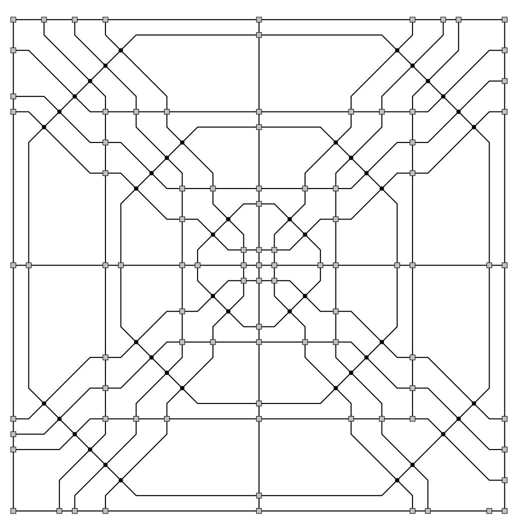


Figure 25: The bend-optimal slog drawing corresponding to the one of Figure 24.

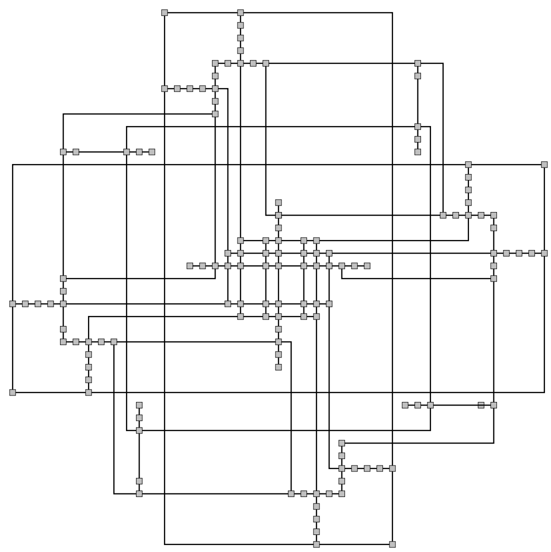


Figure 26: A non-planar orthogonal drawing

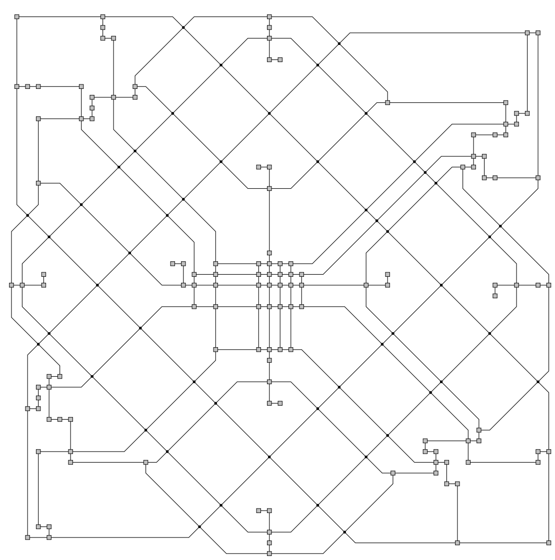


Figure 27: The bend-optimal slog drawing corresponding to the one of Figure 26.

8 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 question is whether every max-degree four graph admits such a drawing. Our experiments led us to believe that it is possible, although we could not prove it. Variants of our basic model may lead to even more flexibility for the drawings. An extension to support higher degree graphs will be necessary to make the approach practical.

Acknowledgements

The authors would like to thank the anonymous reviewers for helpful suggestions and detailed comments. 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. The work of V. Roselli is supported in part by the MIUR project AMANDA "Algorithmics for MAssive and Networked DAta", prot. 2012C4E3KT_001.

References

[1] M. A. Bekos, M. Kaufmann, S. G. Kobourov, and A. Symvonis. Smooth orthogonal layouts. In *Proc. of 20th International Symposium on Graph Drawing*, volume 7704 of *LNCS*, pages 150–161, 2012. doi:10.7155/jgaa.00305.

[2] T. Biedl and G. Kant. A better heuristic for orthogonal graph drawings. In *Proc. of 2nd European Symposium on Algorithms*, volume 855 of *LNCS*, pages 24–35, 1994. doi:10.1007/BFb0049394.

[3] S. Cornelsen and A. Karrenbauer. Accelerated bend minimization. *J. Graph Algorithms Appl.*, 16(3):635–650, 2012. doi:10.7155/jgaa.00265.

[4] M. Eiglsperger, S. P. Fekete, and G. W. Klau. Orthogonal graph drawing. In M. Kaufmann and D. Wagner, editors, *Drawing Graphs*, volume 2025 of *Lecture Notes in Computer Science*, pages 121–171. Springer Berlin Heidelberg, 2001.

[5] U. Fößmeier, C. Heß, and M. Kaufmann. On improving orthogonal drawings: The 4m-algorithm. In *Graph Drawing*, volume 1547 of *LNCS*, pages 125–137, 1998. doi:10.1007/3-540-37623-2_10.

[6] U. Fößmeier and M. Kaufmann. Drawing high degree graphs with low bend numbers. In *Graph Drawing*, volume 1027 of *LNCS*, pages 254–266, 1995. doi:10.1007/BFb0021809.

[7] A. Garg and R. Tamassia. On the computational complexity of upward and rectilinear planarity testing. *SIAM J. Comput.*, 31(2):601–625, 2001. doi:10.1137/S0097539794277123.

[8] Gurobi Optimization, Inc. Gurobi Optimizer Reference Manual, 2014. http://www.gurobi.com.

[9] C. E. Leiserson. Area-efficient graph layouts (for VLSI). In *Proc. of IEEE Symposium on Foundations of Computer Science*, volume 1547 of *IEEE*, pages 270–281, 1980. doi:10.1109/SFCS.1980.13.

[10] Y. Liu, A. Morgana, and B. Simeone. A linear algorithm for 2-bend embeddings of planar graphs in the two-dimensional grid. *DAM*, 81(1–3):69–91, 1998. doi:10.1016/S0166-218X(97)00076-0.

[11] M. Nöllenburg and A. Wolff. Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Trans. Vis. Comput. Graph.*, 17(5):626–641, 2011. doi:10.1109/TVCG.2010.81.

[12] A. Papakostas and I. G. Tollis. A pairing technique for area-efficient orthogonal drawings (extended abstract). In *Proc. of 4th International Symposium on Graph Drawing*, volume 1190 of *Lecture Notes in Computer Science*, pages 355–370. Springer Berlin Heidelberg, 1996. doi:10.1007/3-540-62495-3_60.

[13] R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal of Computing*, 16(3):421–444, 1987. doi:10.1137/0216030.

[14] R. Tamassia and I. G. Tollis. Planar grid embedding in linear time. *IEEE Transactions on Circuits and Systems*, 36(9):1230–1234, 1989. doi:10.1109/31.34669.

[15] L. G. Valiant. Universality considerations in VLSI circuits. *IEEE Transaction on Computers*, 30(2):135–140, 1981. doi:10.1109/TC.1981.6312176.