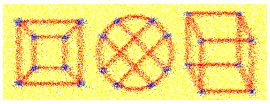

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 19, no. 2, pp. 657–680 (2015)
DOI: 10.7155/jgaa.00369

Planar Octilinear Drawings with One Bend Per Edge

M. A. Bekos [1] *M. Gronemann* [2] *M. Kaufmann* [1] *R. Krug* [1]

[1]Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Germany
[2]Institut für Informatik, Universität zu Köln, Germany

Abstract

In *octilinear drawings* of planar graphs, every edge is drawn as a sequence of horizontal, vertical and diagonal (45°) line segments. In this paper, we study octilinear drawings of low edge complexity, i.e., with few bends per edge. A k-planar graph is a planar graph in which each vertex has degree at most k. In particular, we prove that every 4-planar graph admits a planar octilinear drawing with at most one bend per edge on an integer grid of size $O(n^2) \times O(n)$. For 5-planar graphs, we prove that one bend per edge still suffices in order to construct planar octilinear drawings, but in super-polynomial area. However, for 6-planar graphs we give a class of graphs whose planar octilinear drawings require at least two bends per edge for some edges.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
October 2014	March 2015	April 2015	September 2015	September 2015

Published: November 2015

Article type: Regular paper

Communicated by: C. Duncan and A. Symvonis

This work has been supported by DFG grant Ka812/17-1. The article is based on the preliminary version [M. A. Bekos, M. Gronemann, M. Kaufmann, R. Krug, Planar Octilinear Drawings with One Bend Per Edge, In C. Duncan and A. Symvonis editors, Proc. of 22nd International Symposium on Graph Drawing (GD2014), LNCS 8871, pp. 331-342, 2014].

E-mail addresses: bekos@informatik.uni-tuebingen.de (M. A. Bekos) gronemann@informatik.uni-koeln.de (M. Gronemann) mk@informatik.uni-tuebingen.de (M. Kaufmann) krug@informatik.uni-tuebingen.de (R. Krug)

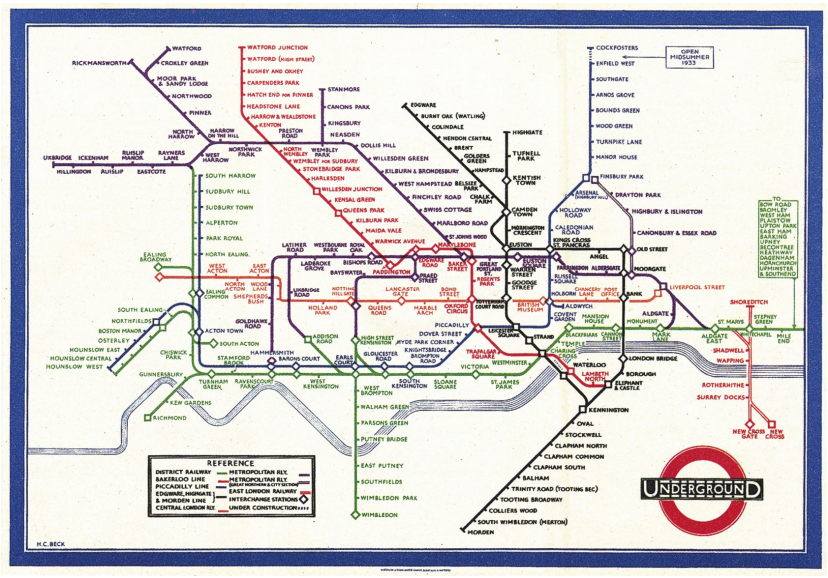


Figure 1: Henry Beck Tube Map (first edition), 1933. Printed at Waterlow & Sons Ltd., London.

1 Motivation

Drawing edges as octilinear paths plays a central role in the design of metro-maps (see e.g., [8, 17, 18]), which dates back to the 1930's when Henry Beck, an engineering draftsman, designed the first schematic map of London Underground using mostly horizontal, vertical and diagonal segments; see Fig.1. Laying out networks in such a way is called *octilinear graph drawing*. More precisely, an *octilinear drawing* of a (planar) graph $G = (V, E)$ of maximum degree 8 is a (planar) drawing $\Gamma(G)$ of G in which each vertex occupies a point on the integer grid and each edge is drawn as a sequence of horizontal, vertical and diagonal ($45°$) line segments. For example, see Fig. 4, 7 and 10a.

For drawings of (planar) graphs to be readable, special care is needed to keep the number of bends small. However, the problem of determining whether a given embedded 8-planar graph (that is, a planar graph of maximum degree 8 with given combinatorial embedding) admits a bendless octilinear drawing is NP-hard [16]. This negative result motivated us to study octilinear drawings of low *edge complexity*, that is, with few bends per edge. Surprisingly enough, very few results relevant to this problem were known, even if the octilinear model has been well-studied in the context of metro-map visualization and map schematization (see e.g. [20]). As an immediate byproduct of a result of Keszegh et al. [12], it turns out that every d-planar graph, with $3 \leq d \leq 8$, admits a planar octilinear drawing with at most two bends per edge; see Section 2. On the other hand, every 3-planar graph on five or more vertices admits a planar octilinear drawing in which all edges are bendless [6, 11].

In this paper, we bridge the gap between the two aforementioned results. In particular, we prove that every 4-planar graph admits a planar octilinear drawing with at most one bend per edge in cubic area (see Section 4). We further show that every 5-planar graph also admits a planar octilinear drawing with at most one bend per edge, but our construction may require super-polynomial area (see Section 5). Hence, we have made no effort in proving a concrete bound. We complement our results by demonstrating an infinite class of 6-planar graphs whose planar octilinear drawings require at least two bends per edge on some edges (see Section 6).

2 Related Work

The research on the (planar) *slope number of graphs* focuses on minimizing the number of used slopes (see e.g., [9, 12, 13, 14, 15]). Octilinear drawings can be seen as a special case thereof, since only 4 slopes (horizontal, vertical and the two diagonals) are used. In a related work, Keszegh et al. [12] showed that any d-planar graph admits a planar drawing with one bend per edge, in which all edge-segments have at most $2d$ different slopes. So, for $d = 4$ and for $d = 5$, we significantly reduce the number of different slopes from 8 and 10, resp., to 4. They also proved that d-planar graphs, with $d \geq 3$, admit planar drawings with two bends per edge that require at most $\lceil \frac{d}{2} \rceil$ different slopes. It is not difficult to transfer this technique to the octilinear model and show that any d-planar graph, with $3 \leq d \leq 8$, admits a planar octilinear drawing with two bends per edge. However, for $d = 3$, Di Giacomo et al. [6] recently proved that any 3-planar graph with $n \geq 5$ vertices has a bendless planar drawing with at most 4 different slopes and angular resolution $\pi/4$ (see also [11]); their approach also yields octilinear drawings.

Octilinear drawings can be considered as an extension of *orthogonal drawings*, which allow only horizontal and vertical segments (i.e., graphs of maximum degree 4 admit such drawings). Tamassia [19] showed that one can efficiently minimize the total number of bends in orthogonal drawings of embedded 4-planar graphs. However, minimizing the number of bends over all embeddings of a 4-planar graph is NP-hard [5]. Note that the core of Tamassia's approach is a min-cost flow algorithm that first computes the angles and the bends of the drawing, producing an *orthogonal representation*, and then based on this representation computes the actual drawing by specifying the exact coordinates for the vertices and the bends of the edges. It is known that Tamassia's algorithm can be employed to produce a bend-minimum octilinear representation for any given embedded 8-planar graph. However, a bend-minimum octilinear representation may not be realizable by a corresponding planar octilinear drawing [3]. Furthermore, the number of bends on a single edge might be very high, but can easily be bounded by applying appropriate capacity constraints to the flow-network.

Biedl and Kant [1] showed that any 4-planar graph except the octahedron admits a planar orthogonal drawing with at most two bends per edge on an $O(n^2)$ integer grid. Hence, the octilinear drawing model allows us to reduce the number of bends per edge at the cost of an increased area. On the other hand, not all 4-planar graphs admit orthogonal drawings with one bend per edge; however, testing whether a 4-planar graph admits such a drawing can be done in polynomial time [2]. In the context of metro-map

visualization, several approaches have been proposed to produce metro-maps using octilinear or nearly-octilinear polylines, such as force-driven algorithms [8], hill climbing multi-criteria optimization techniques [18] and mixed-integer linear programs [17]. However, the problem of laying out a metro-map in an octilinear fashion is significantly more difficult than the octilinear graph drawing problem, as several metro-lines may connect the same pair of stations and the positions of the vertices have to reflect geographical coordinates of the stations.

3 Preliminaries

In our algorithms, we incrementally construct the drawings similar to the method of Kant [10]. We first employ the canonical order to cope with triconnected graphs. Then, we extend them to biconnected graphs using the SPQR-tree [4] and to simply-connected graphs using the BC-tree. In this section we briefly recall them; however we assume basic familiarity.

Definition 1 (Canonical order [10]) *For a given triconnected plane graph $G = (V, E)$ let $\Pi = (P_0, \ldots, P_m)$ be a partition of V into paths such that $P_0 = \{v_1, v_2\}$, $P_m = \{v_n\}$ and $v_2 \rightarrow v_1 \rightarrow v_n$ is a path on the outer face of G. For $k = 0, \ldots, m$ let G_k be the subgraph induced by $\bigcup_{i=0}^{k} P_i$ and assume it inherits its embedding from G. Partition Π is a canonical order of G if for each $k = 1, \ldots, m-1$ the following hold: (i) G_k is biconnected, (ii) all neighbors of P_k in G_{k-1} are on the outer face of G_{k-1}, (iii) all vertices of P_k have at least one neighbor in P_j for some $j > k$, (iv) if $|P_k| > 1$, then $deg_{G_k}(v) = 2\ \forall v \in P_k$ and P_k is called chain; otherwise P_k is called singleton.*

Definition 2 (BC-tree) *The* BC-tree $\mathcal{B}$ *of a connected graph G has a B-node for each biconnected component of G and a C-node for each cutvertex of G. Each B-node is connected with the C-nodes that are part of its biconnected component.*

An SPQR-tree [4] provides information about the decomposition of a biconnected graph into its triconnected components. It can be computed in linear time and space [7]. Every triconnected component is associated with a node μ in the SPQR-tree $\mathcal{T}$. The triconnected component itself is referred to as the *skeleton* of μ, denoted by $G_\mu^{skel} = (V_\mu^{skel}, E_\mu^{skel})$. We refer to the degree of a vertex $v \in V_\mu^{skel}$ in G_μ^{skel} as $deg_\mu^{skel}(v)$. We say that μ is an *R-node*, if G_μ^{skel} is a simple triconnected graph. A bundle of at least three parallel edges classifies μ as a *P-node*, while a simple cycle of length at least three classifies μ as an *S-node*. By construction R-nodes are the only nodes of the same type that are allowed to be adjacent in $\mathcal{T}$. The leaves of $\mathcal{T}$ are formed by the *Q-nodes*. Their skeleton consists of two parallel edges; one of them corresponds to an edge of G and is referred to as *real edge*. The skeleton edges that are not real are referred to as *virtual edges*. A virtual edge e in G_μ^{skel} corresponds to a tree node μ' that is adjacent to μ in $\mathcal{T}$, more precisely, to another virtual edge e' in $G_{\mu'}^{skel}$. We assume that $\mathcal{T}$ is rooted at a Q-node. Hence, every skeleton (except the one of the root) contains exactly one virtual edge $e = (s, t)$ that has a counterpart in the skeleton of the parent node. We call this edge the *reference edge* of μ denoted by *ref*(μ). Its endpoints, s and t, are named the

poles of μ denoted by $\mathcal{P}_\mu = \{s, t\}$. Every subtree rooted at a node μ of $\mathcal{T}$ induces a subgraph of G called the *pertinent graph* of μ that we denote by $G_\mu^{pert} = (V_\mu^{pert}, E_\mu^{pert})$. We abbreviate the degree of a node v in G_μ^{pert} with $deg_\mu^{pert}(v)$. The pertinent graph is the subgraph of G for which the subtree describes the decomposition.

Now let G be a simple, biconnected k-planar graph, whose SPQR-tree $\mathcal{T}$ is given. Additionally, we may assume that $\mathcal{T}$ is rooted at a Q-node that is adjacent to an S- or R-node. Notice that at least one such node exists since the graph does not contain any multi-edges, which would be the case if only a P-node existed. Biconnectivity and maximum degree of k yield basic bounds for the graph degree of a node $v \in V$, i.e., $2 \leq deg(v) \leq k$. By construction the pertinent graph of a tree node μ is a (connected) subgraph of G; thus $1 \leq deg_\mu^{pert}(v) \leq deg(v)$. For the degrees in a skeleton graph G_μ^{skel}, we obtain bounds based on the type of the corresponding node. Skeletons of Q-nodes are cycles of length two, whereas S-nodes are by definition simple cycles of length at least three; hence, $deg_\mu^{skel}(v) = 2$. For P- and R-nodes the degree can be bounded by $3 \leq deg_\mu^{skel}(v) \leq k$, since the skeleton of the former is at least a bundle of three parallel virtual edges and the latter's skeleton is triconnected by definition. The upper bound is derived from the relation between skeleton and graph degrees: A virtual edge $e = (s, t)$ hides at least one incident edge of each node (not necessarily an (s, t)-edge). This observation can be easily proven by induction on the tree. Hence, $2 \leq deg_\mu^{skel}(v) \leq deg(v)$.

Next, we use this observation to derive bounds for the pertinent degree by distinguishing two cases depending on whether v is a pole or not. Recall that G_μ^{pert} is a subgraph of G that is obtained by recursively replacing virtual edges by the skeletons of the corresponding children. In the first case when v is an internal node in G_μ^{pert}, i.e., $v \notin \mathcal{P}_\mu$, v is not incident to the reference edge in G_μ^{skel}. Thus, every edge of G hidden by a virtual edge in G_μ^{skel} is in G_μ^{pert}. Hence, $deg_\mu^{skel}(v) \leq deg_\mu^{pert}(v) \leq k$. In the other case, i.e., $v \in \mathcal{P}_\mu$, at least one edge that is hidden by the reference edge, is not part of G_μ^{pert}, thus, $deg_\mu^{skel}(v) - 1 \leq deg_\mu^{pert}(v) \leq k - 1$. Notice that the lower bounds depend on the skeleton degree, which in turn depends on the type of node, unlike the upper bounds that hold for all tree nodes. The next lemma tightens these bounds based on the type of the parent node.

Lemma 1 *Let μ be a tree node that is not the root in the SPQR-tree $\mathcal{T}$ of a simple, biconnected, k-planar graph G and let μ' be its parent in $\mathcal{T}$. For $v \in \mathcal{P}_\mu$, it holds that $deg_\mu^{pert}(v) \leq k - 2$, if μ' is a P- or an R-node and $deg_\mu^{pert}(v) \leq k - 1$ otherwise, i.e. μ' is an S- or a Q-node.*

Proof: Notice that every skeleton edge hides at least one edge. Therefore, every edge incident to a vertex v may hide at most $k + 1 - deg_\mu^{skel}(v)$ edges. Since for the poles of P- and R-nodes, $deg_\mu^{skel}(v) \geq 3$ holds and for S-nodes $deg_\mu^{skel}(v) = 2$, the claim follows. □

Lemma 2 *Let $\mathcal{T}$ be the SPQR-tree of a planar biconnected graph $G = (V, E)$ with $\deg(v) \geq 3$ for every $v \in V$. There exists at least one Q-node in $\mathcal{T}$ that is adjacent to a P- or an R-node.*

Proof: Assume to the contrary that all Q-nodes are adjacent to S-nodes only. We pick such a Q-node and root $\mathcal{T}$ at it. Let μ be an S-node with poles $\mathcal{P}_\mu = \{s, t\}$ such that there is no other S-node in the subtree of μ. By definition of an S-node, μ has at least two children. If all of them are Q-nodes, then there exists a $v \in V_\mu^{skel}$ with $s \neq v \neq t$ and $\deg(v) = 2$; a contradiction. Hence, there is at least one child μ' that is a P- or an R-node. However, since the leaves of $\mathcal{T}$ are Q-nodes and those are not allowed to be children of P- and R-nodes by our assumption, there exists at least one other S-node in the subtree of μ' and therefore in the subtree of μ which contradicts our choice of μ. □

4 Octilinear Drawings of 4-Planar Graphs

In this section, we focus on planar octilinear drawings of 4-planar graphs. We first consider the case of triconnected 4-planar graphs and then we extend our approach first to biconnected and then to simply-connected graphs. Central in our approach is the port assignment; by the *port* of a vertex we refer to the side of the vertex an edge is connected to. The different ports on a vertex are distinguished by the cardinal directions.

4.1 The Triconnected Case

Let $G = (V, E)$ be a triconnected 4-planar graph and $\Pi = \{P_0, \ldots, P_m\}$ be a canonical order of G. We momentarily neglect the edge (v_1, v_2) of the first partition P_0 of Π and we start by placing the second partition, say a chain $P_1 = \{v_3, \ldots, v_{|P_1|+2}\}$, on a horizontal line from left to right. Since by definition of Π, v_3 and $v_{|P_1|+2}$ are adjacent to the two vertices, v_1 and v_2, of the first partition P_0, we place v_1 to the left of v_3 and v_2 to the right of $v_{|P_1|+2}$. So, they form a single chain where all edges are drawn using horizontal line segments that are attached to the east and west port at their endpoints. The case where P_1 is a singleton is analogous. Having laid out the base of our drawing, we now place in an incremental manner the remaining partitions. Assume that we have already constructed a drawing for G_{k-1} and we now have to place P_k, for some $k = 2, \ldots, m-1$.

In case where $P_k = \{v_i, \ldots, v_j\}$ is a chain of $j - i + 1$ vertices, we draw them from left to right along a horizontal line one unit above G_{k-1}. Since v_i and v_j are the only vertices that are adjacent to vertices in G_{k-1}, both only to one vertex, we place the chain between those two as in Fig.2a. The port used at the endpoints of P_k in G_{k-1} depends on the following rule: Let v_i' (v_j', resp.) be the neighbor of v_i (v_j, resp.) in G_{k-1}. If the edge (v_i, v_i') ((v_j, v_j'), resp.) is the last to be attached to vertex v_i' (v_j', resp.), i.e., there is no vertex v in $P_l \in \Pi$, $l > k$ such that $(v_i', v) \in E$ ($(v_j', v) \in E$, resp.), then we use the northern port of v_i' (v_j', resp.). Otherwise, we choose the north-east port for (v_i, v_i') and the north-west port for (v_j, v_j').

In case of a singleton $P_k = \{v_i\}$, we can apply the previous rule if the singleton is of degree 3, as the third neighbor of v_i must belong to a partition Π_j for some $j > k$. However, in case where v_i is of degree 4 we may have to deal with an additional third edge (v_i, v) that connects v_i with G_{k-1}. By the placement so far, we may assume that v lies between the other two endpoints, thus, we place v_i such that $x(v_i) = x(v)$. This enables us to draw (v_i, v) as a vertical line segment; see Fig.2b.

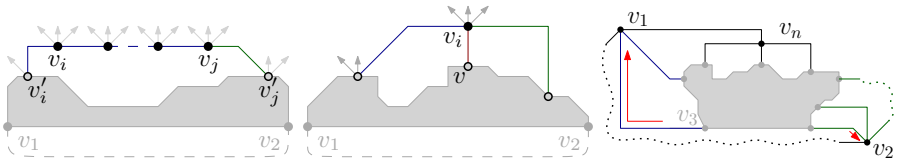


Figure 2: (a) Horizontal placement of a chain $P_k = \{v_i, \ldots, v_j\}$. (b) Placement of a singleton $P_k = \{v_i\}$ with degree 4. (c) Final layout after repositioning v_1 and v_2 (the shape of the dotted edges can be obtained by extending the stubs until they intersect).

The above procedure is able to handle all chains and singletons except the last partition P_m, because v_n may have four edges pointing downwards. One of these edges is (v_n, v_1), by definition of Π. We exclude (v_n, v_1) and draw v_n as an ordinary singleton. Then, we shift v_1 to the left and up as in Fig.2c. This enables us to draw (v_1, v_n) as a combination of a horizontal and a vertical segment. For (v_1, v_2), we move v_2 one unit to the right and down. We free the west port of v_2 by redrawing its incident edges as in Fig.2c and attach (v_1, v_2) to it. Edge (v_1, v_2) will be drawn as a diagonal segment with positive slope connected to v_1 and a horizontal segment connected to v_2, which requires one bend. Let (v_2, v_i) be the other incomplete edge according to Fig.2c. It will be drawn using a diagonal segment with positive slope connected to v_2 and a horizontal segment connected to v_i, again requiring one bend.

So far, we have specified a valid port assignment and the y-coordinates of the vertices. However, we have not fully specified their x-coordinates. Notice that by construction every edge, except the ones drawn as vertical line segments, contains exactly one horizontal segment. This enables us to stretch the drawing horizontally by employing appropriate cuts. A *cut*, for us, is a y-monotone continuous curve that crosses only horizontal segments and divides the current drawing into a left and a right part; see Fig.3a. It is not difficult to see that we can shift the right part of the drawing that is defined by the cut further to the right while keeping the left part of the drawing on place and the result remains a valid octilinear drawing.

To compute the x-coordinates, we proceed as follows. We first assign consecutive x-coordinates to the first two partitions and from there on we may have to stretch the drawing in two cases. The first one appears when we introduce a chain, say P_k, as it may not fit into the gap defined by its two adjacent vertices in G_{k-1} (see Fig.3a). In this case, we horizontally stretch the drawing between its two adjacent vertices in G_{k-1} to ensure that their horizontal distance is at least $|P_k|+1$. The other case appears when an edge that contains a diagonal segment is to be drawn. Such an edge requires a horizontal distance between its endpoints that is at least the height it bridges. We also have to prevent it from intersecting any horizontal-vertical combinations in the face below it. We can cope with both cases by horizontally stretching the drawing by a factor that is bounded by the current height of the drawing. Since the height of the resulting drawing is bounded by $|\Pi| = O(n)$, it follows that in the worst case its width is $O(n^2)$. Based on the above, we are now ready to state the main theorem of this subsection.

Algorithm 1: 4-Planar Triconnected

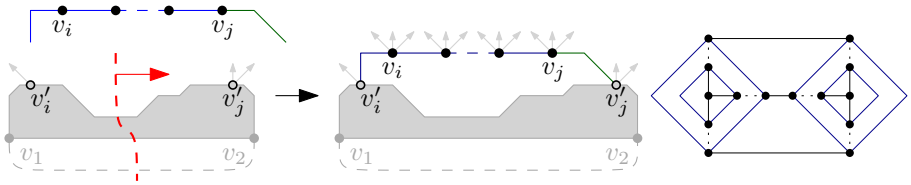
Input : A 4-planar triconnected graph $G = (V, E)$ and a canonical order $\Pi = \{P_0, \ldots, P_m\}$ of G, where $P_0 = \{v_1, v_2\}$ and $P_m = \{v_n\}$
Output: An octilinear drawing $\Gamma(G)$ of G

$(x(v_1), y(v_1)) \leftarrow (0, 0)$;
Place $P_1 = \{v_3, \ldots, v_{|P_1|+2}\}$ at $(n, 0), \ldots, (n \cdot |P_1|, 0)$;
$(x(v_2), y(v_2)) \leftarrow (n \cdot (|P_1| + 1), 0)$;

for $k \leftarrow 2$ **to** $m - 1$ **do**

- Let $P_k = \{v_i, \ldots, v_j\}$;
- $v'_i \leftarrow$ leftmost neighbor of P_k in G_{k-1};
- $v'_j \leftarrow$ rightmost neighbor of P_k in G_{k-1};
- `//Ensure that` P_k `fits in between` v'_i `and` v'_j
- **if** $(x(v'_j) - x(v'_i) < n \cdot (|P_k| + 1))$ **then**
  - Stretch $\Gamma(G)$ horizontally such that $x(v'_j) - x(v'_i) = n \cdot (|P_k| + 1)$;
- `//Compute y-coordinates`
- **for** $v \in P_k$ **do** $y(v) \leftarrow k - 1$;
- `//Compute x-coordinates`
- **if** $|P_k| = 1$ **and** $deg_{G_k}(v_i) = 3$ **then** `//Deg-3 Singleton Case`
  - $v \leftarrow$ neighbor of v_i in G_{k-1} s.t. $v \notin \{v'_i, v'_j\}$;
  - $x(v_i) \leftarrow x(v)$;
  - Draw (v, v_i) as vertical segment;
- **else** `//Chain & Deg-2 Singleton Cases`
  - **for** $t := 0$ **to** $j - i$ **do**
    - $x(v_{i+t}) \leftarrow x(v'_i) + n \cdot (t + 1)$;
  - Draw the interior edges of P_k as straight line segments;
- `//Route edges to` v'_i `and` v'_j
- **if** *the north-east port at* v'_i *is occupied* **or** $deg(v'_i) = 3$ **then**
  - Draw (v'_i, v_i) as horizontal-vertical combination;
- **else**
  - Draw (v'_i, v_i) as horizontal-diagonal combination;
- **if** *the north-west port at* v'_j *is occupied* **or** $deg(v'_j) = 3$ **then**
  - Draw (v'_j, v_j) as horizontal-vertical combination;
- **else**
  - Draw (v'_j, v_j) as horizontal-diagonal combination;

Place $P_m = \{v_n\}$ and reposition v_1 and v_2 as in Fig.2c.



(a) (b)

Figure 3: (a) Illustration of a cut, in case of a chain. (b) Nested separating triangles each requiring one bend.

Theorem 1 *Given a triconnected* 4*-planar graph* G*, we can compute in* $O(n)$ *time an octilinear drawing of* G *with at most one bend per edge on an* $O(n^2) \times O(n)$ *grid.*

Proof: In order to keep the time complexity of our algorithm linear, we employ a simple trick. We assume that any two adjacent points of the underlying integer grid are by n units apart in the horizontal direction and by one unit in the vertical direction. This a priori ensures that all edges that contain a diagonal segment will not be involved in crossings and simultaneously does not affect the total area of the drawing, which asymptotically remains cubic. On the other hand, the advantage of this approach is that we can use the shifting method of Kant [10] to cope with the introduction of chains in the drawing, that needs $O(n)$ time in total by keeping relative coordinates that can be efficiently updated and computing the absolute values only at the last step. □

An outline for a simple reference implementation that employs the idea of keeping the vertices at a horizontal distance of n is given in Algorithm 1. The result is a rather simple algorithm which may result in wider drawings than necessary. With some more effort and by shifting the vertices more carefully, one may considerably improve the actual drawing area in practice, while keeping the running time linear. An example produced by a more sophisticated implementation is given in Fig. 4.

Note that our algorithm produces drawings that have linear number of bends in total (in particular, exactly $2|\Pi| = O(n)$ bends). In the following, we prove that this bound is asymptotically tight.

Theorem 2 *There exists an infinite class of* 4*-planar graphs which do not admit bendless octilinear drawings and if they are drawn with at most one bend per edge, then a linear number of bends is required.*

Proof: Based on the simple fact that in an orthogonal drawing a triangle requires at least one bend, we describe an example that translates this idea to the octilinear model (see Fig.3b). While a triangle can easily be drawn bendless with the additional ports available, we will occupy those to enforce the creation of a bend as in the orthogonal model. Our construction is heavily based on the so-called *separating triangle*, i.e., a three-cycle whose removal disconnects the graph. Each vertex of such a triangle has degree 4. Any triangle which is drawn bendless has a $45°$ angle inside. But since the triangles are nested and have incident edges going inside of the triangles, this is impossible. Note that the graph of our construction is triconnected. Hence, its embedding is fixed up to the choice of the outer face. □

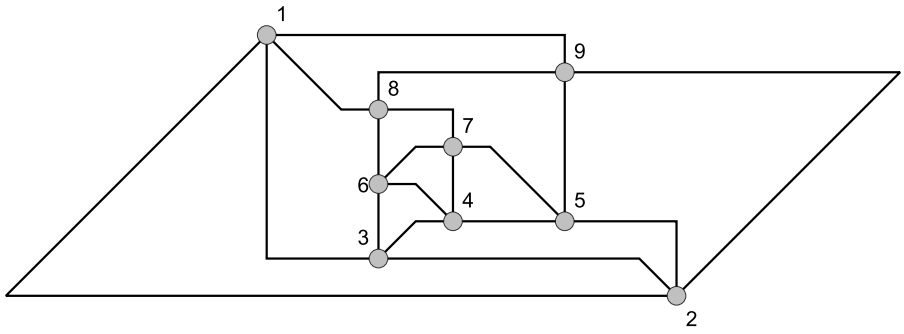


Figure 4: A sample drawing of a triconnected 4-planar graph produced by an implementation of our algorithm. Vertices are labeled according to their numbering in the canonical ordering.

4.2 The Biconnected Case

Following standard practice, we employ a rooted SPQR-tree and assume for a tree node that the pertinent graphs of its children are drawn in a pre-specified way. Consider a node μ in $\mathcal{T}$ with poles $\mathcal{P}_\mu = \{s, t\}$. In the drawing of G_μ^{pert}, s should be located at the upper-left and t at the lower-right corner of the drawing's bounding box with a port assignment as in Fig.5a. In general, we assume that the edges incident to s (t, resp.) use the western (eastern, resp.) port at their other endpoint, except of the northern-most (southern-most, resp.) edge which may use the north (south, resp.) port instead. In that case we refer to s and t as *fixed*; see $\overline{e}_s, \overline{e}_t$ in Fig.5a. More specifically, we maintain the following invariants:

IP-1: The width (height) of the drawing of μ is quadratic (linear) in the size of G_μ^{pert}. s is located at the upper-left and t at the lower-right corner of the drawing's bounding box.

IP-2: If $deg_\mu^{pert}(s) \geq 2$, s is fixed; t is fixed if $deg_\mu^{pert}(t) = 3$ and μ's parent is not the root of $\mathcal{T}$.

IP-3: The edges that are incident to s and t in G_μ^{pert} use the south, south-east and east ports at s and the north, north-west and west port at t, resp. If s or t is not fixed, incident edges are attached at their other endpoints via the west and east port, respectively. If s or t is fixed, the northern-most edge at s and the southern-most edge at t may use the north (south, resp.) port at its other endpoint.

Notice that the port assignment, i.e. IP-3, guarantees the ability to stretch the drawing horizontally even in the case where both poles are fixed. Furthermore, IP-2 is *interchangeable* in the following sense: If $deg_\mu^{pert}(s) = 2$ and $deg_\mu^{pert}(t) = 1$, then s is fixed but t is not. But, if we relabel s and t such that $t' = s$ and $s' = t$, then $deg_\mu^{pert}(s') = 1$ and $deg_\mu^{pert}(t') = 2$.

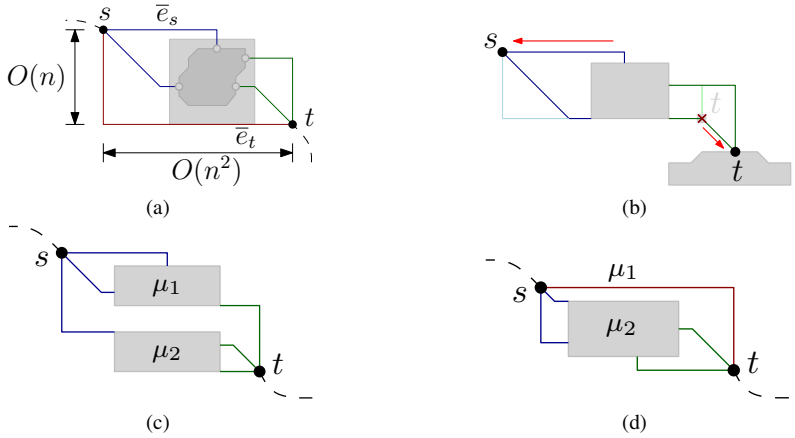


Figure 5: (a) Schematic view of the layout requirements. (b) Creating a nose at t. (c) First P-node subcase without an (s,t)-edge but s might be fixed in a child μ_1. (d) Second P-node subcase with an (s,t)-edge where t might get fixed in a child μ_2.

By IP-2, we can create a drawing where both s' and t' are not fixed and located in the upper-left and lower-right corner of the drawing's bounding box. Afterwards, we mirror the resulting layout vertically and horizontally to obtain one where s and t are in their respective corners and not fixed. Notice that in general the property of being fixed is not symmetric, e.g., when $deg_{\mu}^{pert}(s) = 3$ and $deg_{\mu}^{pert}(t) = 2$ holds, s remains fixed while t becomes fixed as well. For a non-fixed vertex, we introduce an operation that is referred to as forming or creating a *nose*: We say that a pair of edges pointing downwards to a common vertex w form a nose if one is using the north port of w while the other one uses either the north-west or the north-east port of w; see Fig.5b, where t has been moved downwards at the cost of a bend. As a result, the west port of t is free.

P-node case: Let μ be a P-node. By Lemma 1, for a child μ' of μ, it holds that $deg_{\mu'}^{pert}(s) \leq 2$ and $deg_{\mu'}^{pert}(t) \leq 2$. So, t can form a nose in μ', while s might be fixed in the case where $deg_{\mu'}^{pert}(s) = 2$. Notice that there exists at most one such child due to the degree restriction. We distinguish two cases based on the existence of an (s,t)-edge.

In the first case, assume that there is no (s,t)-edge. We draw the children of μ from top to bottom such that a possible child in which s is fixed, is drawn topmost (see μ_1 in Fig.5c). In the second case, we draw the (s,t)-edge at the top and afterwards the remaining children (see Fig.5d). Of course, this works only if s is not fixed in any of the other children. Let μ' be such a potential child where s is fixed, i.e., $deg_{\mu'}^{pert}(s) = 2$, and thus, the only child that remains to be drawn. Here, we use the property of interchangeability to "unfix" s in μ'. As a result s can form a nose, whereas t may now be fixed in μ' when $deg_{\mu'}^{pert}(t) = 2$ holds, as in Fig.5d. However, then $deg_{\mu}^{pert}(t) = 3$ follows. Notice that the presence of an (s,t)-edge implies that the parent of μ is not the root of $\mathcal{T}$, since this would induce a pair of parallel edges. Hence, by IP-2 we are allowed to fix t in μ. Port assignment and area requirements comply in both cases with our invariant properties.

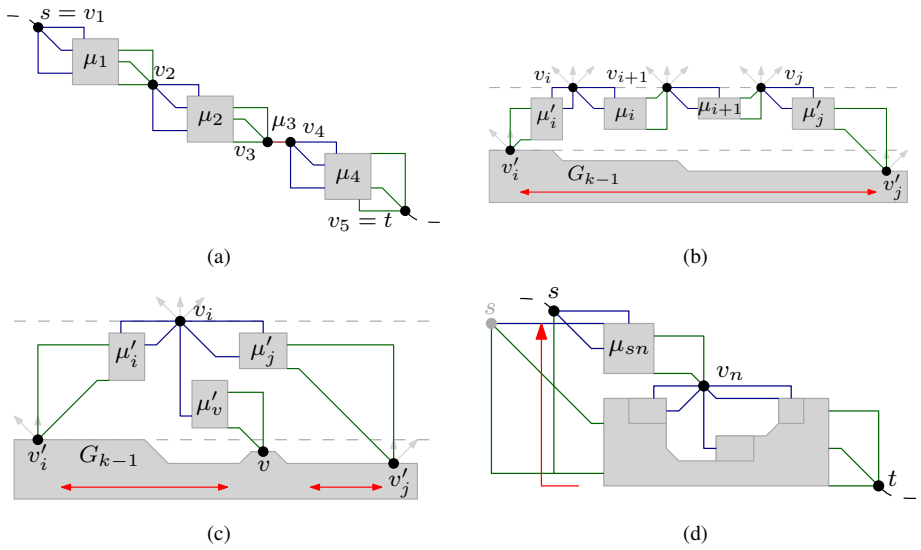


Figure 6: (a) S-node with children $\mu_1, \ldots, \mu_4$; μ_3 is a Q-node representing the edge (v_3, v_4). Optional edges are drawn dotted. (b) Example for a chain $v_i, \ldots, v_j$ with virtual edges representing $\mu_i, \ldots, \mu_{j-1}$ in the R-node case. (c) Singleton v_i with possibly three incident virtual edges representing μ'_i, μ'_v, μ'_j. (d) Placing v_n and moving up s which might be fixed in μ_{sn}.

S-node case: We place the drawings of the children, say $\mu_1, \ldots, \mu_\ell$, of an S-node μ in a "diagonal manner" such that their corners touch as in Fig.6a. In case of Q-nodes being involved, we draw their edges as horizontal segments (see, e.g., edge (v_3, v_4) in Fig.6a that corresponds to Q-node μ_3). Observe that s and t inherit their port assignment and pertinent degree from μ_1 and μ_ℓ, respectively, i.e., $deg^{pert}_{\mu}(s) = deg^{pert}_{\mu_1}(s)$ and $deg^{pert}_{\mu}(t) = deg^{pert}_{\mu_\ell}(t)$. So, we may assume that s is fixed in μ, if s is fixed in μ_1. Similarly, t is fixed in μ, if t is fixed in μ_ℓ. By IP-2, t is not allowed to be fixed in the case where the parent of μ is the root of $\mathcal{T}$. However, Lemma 2 states that we can choose the root such that t is not fixed in that case, and thus, complies with IP-2. Since we only concatenated the drawings of the children, IP-1 and IP-3 are satisfied.

R-node case: For the case where μ is an R-node with poles $\mathcal{P}_\mu = \{s, t\}$, we follow the basic idea of the triconnected algorithm of the previous section and describe the modifications necessary to handle the drawing of the children of μ. To do so, we assume the worst case where no child of μ is a Q-node. Let μ_{uv} denote the child that is represented by the virtual edge $(u, v) \in E^{skel}_{\mu}$. Notice that due to Lemma 1, $deg^{pert}_{\mu_{uv}}(u) \leq 2$ and $deg^{pert}_{\mu_{uv}}(v) \leq 2$ holds. Hence, with IP-2 we may assume that at most one out of u and v is fixed in μ_{uv}. We choose the first partition in the canonical ordering to be $P_0 = \{s, t\}$ and distinguish again between whether the partition to be placed next is a chain or a singleton.

In case of a chain, say $P_k = \{v_i, \ldots, v_j\}$ with two neighbors v'_i and v'_j in G_{k-1},

we have to replace two types of edges with the drawings of the corresponding children: the edges $(v_i, v_{i+1}), \ldots, (v_{j-1}, v_j)$ representing the children $\mu_i, \ldots, \mu_{j-1}$ and (v'_i, v_i) ((v_j, v'_j) resp.) representing μ'_i (μ'_j resp.). We place the vertices of P_k on a horizontal line high enough above G_{k-1} such that every drawing may fit in-between it and G_{k-1}. Then, we insert the drawings aligned below the horizontal line and choose for $i \leq l < j$, v_l to be the fixed node in μ_l, whereas in μ'_i (μ'_j resp.), we set v_i (v_j resp.) to be fixed. Hence, for $i \leq l < j$, v_{l+1} may form a nose in μ_l pointing upwards while v'_i and v'_j form each one downwards as depicted in Fig.6b. By stretching the drawing horizontally, we increase its width, which allows us to further increase its height if necessary.

For the case where $P_k = \{v_i\}$ and $i \neq n$ is a singleton, we only outline the difference which is a possible third edge (v_i, v) to G_{k-1} representing say μ'_v. While the other two involved children, say μ'_i and μ'_j, are handled as in the chain-case, μ'_v requires extra height now and we may place v_i such that μ'_v fits below μ'_j as in Fig.6c. Notice that $deg^{pert}_{\mu'_v}(v_i) = 1$ holds and therefore by IP-2 both v_i and v are not fixed in μ'_v. Hence, forming a nose at v_i and v as in Fig.6c is feasible.

It remains to describe the special case where the last singleton $P_k = \{v_n\}$ is placed. Since $s, t \in P_0$, both have not been fixed yet. We proceed as in the triconnected algorithm and move $s = v_1$ above v_n as depicted in Fig.6d, high enough to accommodate the drawing of the child μ_{sn} represented by the edge (s, v_n). Since we may require v_n to form a nose in μ_{sn} as in Fig.6d, we choose s to be fixed in μ_{sn}. However, we are allowed by IP-2 to fix s since t remains unfixed. For the area constraints of IP-1, we argue as follows: Although some diagonal segments may force us to stretch the whole drawing by its height, the height of the drawing has been kept linear in the size of G^{pert}_{μ}. Since we increase the width by the height a constant number of times per step, the resulting width remains quadratic.

Root case: For the root of $\mathcal{T}$ we distinguish two cases: In the first case, there exists a vertex $v \in V$ with $\deg(v) \leq 3$. Then, we choose as root a Q-node μ that represents one of its three incident edges and orient the poles $\{s, t\}$ such that $t = v$. Hence, for the child μ' of μ follows $deg^{pert}_{\mu'}(t) \leq 2$. In the other case, i.e., for every $v \in V$ we have $\deg(v) = 4$, we choose a Q-node that is not adjacent to an S-node, whose existence is guaranteed by Lemma 2. In both cases, we may form a nose with t pointing downwards and draw the edge as in the triconnected algorithm.

Theorem 3 *Given a biconnected 4-planar graph G, we can compute in $O(n)$ time an octilinear drawing of G with at most one bend per edge on an $O(n^2) \times O(n)$ grid.*

Proof: The SPQR-tree $\mathcal{T}$ can be computed in $O(n)$-time and its size is linear to the size of G [7]. The pertinent degrees of the poles at every node can be pre-computed by a bottom-up traversal of $\mathcal{T}$. Drawing a P-node requires constant time; S- and R-nodes require time linear to the size of the skeleton. However, the sum over all skeleton edges is linear, as every virtual edge corresponds to a tree node. □

An example illustrating our approach is given in Fig.7.

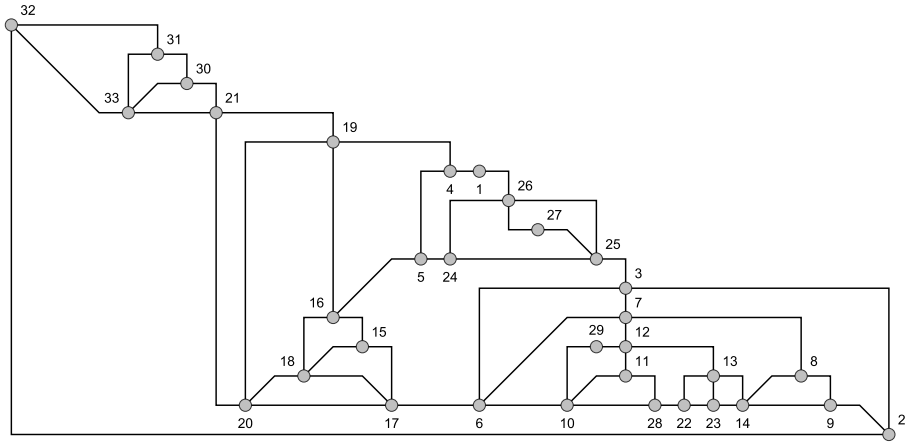


Figure 7: Example layout of a biconnected 4-planar graph. Vertices are labeled by their indices. The corresponding SPQR-tree $\mathcal{T}$ has been rooted at a Q-node representing the edge (v_{32}, v_2) with the only child being an S-node whose skeleton is the simple cycle v_{32}, v_{21}, v_2. It has two R-nodes as children, a smaller in the upper left (with poles $\{v_{32}, v_{21}\}$) and a larger one (with poles $\{v_{21}, v_2\}$) occupying most of the drawing area. The latter one contains two smaller S-nodes (with poles $\{v_{10}, v_{12}\}$ and $\{v_4, v_{26}\}$) and a P-node (with poles $\{v_{26}, v_{25}\}$) that has two children. One of them being an (s, t)-edge, the other one an S-node.

4.3 The Simply-Connected Case

In the following, we turn our attention to the connected case. We start by computing the BC-tree of G and root it at some arbitrary B-node. Every B-node, except the root, contains a designated cut vertex that links it to the parent. A *bridge* for a biconnected component consists only of a single edge. Similar to the biconnected case, we define an invariant for the drawing of a subtree: The cut vertex that links the subtree to the parent is located in the upper left corner of the drawing's bounding box.

Any subgraph, say G_b, induced by a non-bridge biconnected component can be laid out using the biconnected algorithm. However, to construct a drawing that satisfies our invariant we have to consider two cases. First, the cut vertex, say v_b, that links G_b to the parent, has to be drawn in the upper-left corner of the subtrees drawing. Second, there may be other cut vertices of G in G_b to which we have to attach their corresponding subtrees. We describe in detail both cases below:

Parent cut vertex v_b: We start by describing how to root the SPQR-tree $\mathcal{T}_b$ for G_b so that v_b is located in the upper-left corner. There are at least two Q-nodes having v_b as a pole (as G_b is biconnected) and the degree of v_b in G_b is at most 3. In the biconnected case, we distinguished for the root of the tree between whether there exists $v \in V$ with $\deg(v) \leq 3$ or not. Hence, we may choose for the root of $\mathcal{T}_b$ a Q-node having v_b as a pole and orient it such that $v_b = t$, thus, satisfying $\deg(t) \leq 3$. Then, we flip the final drawing of G_b such that t is in the upper left corner (see Fig.8a).

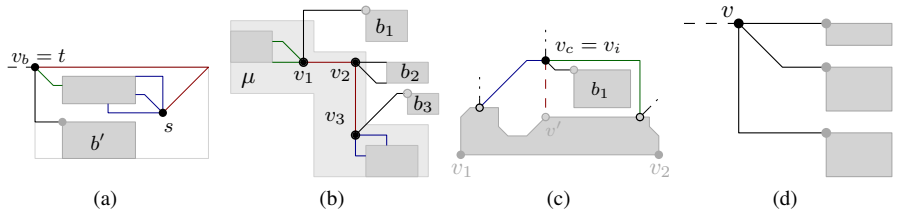


Figure 8: (a) Rooting the SPQR-tree such that v_b is in the upper-left corner. (b) All possible situations at an S-node μ. For attaching b_2 to v_2, the layout had to be modified. (c) Attaching a subtree via a bridge to a cut vertex v_c in an R-node. The dashed edge (v_i, v') may only be present if $v_i = v_n$. (d) A cut vertex where all of its children are attached via bridges.

Child cut vertex v_c: Let v_c be a cut vertex in G_b that is not the link to the parent. If v_c has degree 3 in G_b, then it may occur in the pertinent graph of every node. However, in this case we only have to attach a subtree of the BC-tree that is connected via a bridge. This poses no problem, as there are enough free ports available at v_c and we can afford a bend at the bridge. We only consider S- and R- nodes here since the poles of P-nodes occur in the pertinent graphs of the first two. For R-nodes we assume that the south east port at v_c is free. So, we attach the drawing via the bridge by creating a bend as in Fig.8c. In the diagonal drawing of an S-node, the north-east port is free. So, we can proceed similarly; see Fig.8b.

If v_c has degree 2 in G_b, it only occurs in the pertinent graph of an S-node; see v_3 in Fig.8b. However, we may no longer assume that the bridge is available. As a result, we cannot afford a bend and have to deal with two incident edges instead of one. We modify the drawing by exploiting the two real edges incident to v_c in the S-nodes layout to free the east and south east port; see v_2 in Fig.8b. This enables us to attach the subtrees drawing without modifying it. We finish this section by dealing with the most simple case where there are only bridges attached to a cut vertex. The idea is illustrated in Fig.8d and matches our layout specification.

Theorem 4 *Given a connected 4-planar graph G, we can compute in $O(n)$ time an octilinear drawing of G with at most one bend per edge on an $O(n^2) \times O(n)$ grid.*

Proof: Decomposing a connected graph into its biconnected components takes linear time. It remains to discuss the area requirement. Inserting a subtree with n vertices and the given dimensions into the drawing of an R- or S-node clearly increases the width of the drawing by at most $O(n^2)$ and the height by at most $O(n)$. Hence, the total drawing area is cubic, as desired. □

5 Octilinear Drawings of 5-Planar Graphs

In this section, we focus on planar octilinear drawings of 5-planar graphs. As in Section 4, we first consider the case of triconnected 5-planar graphs and then we extend our approach first to biconnected and then to the simply-connected graphs.

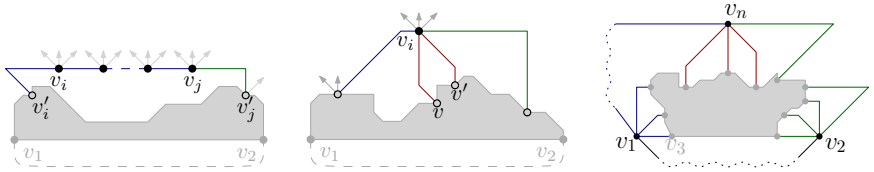


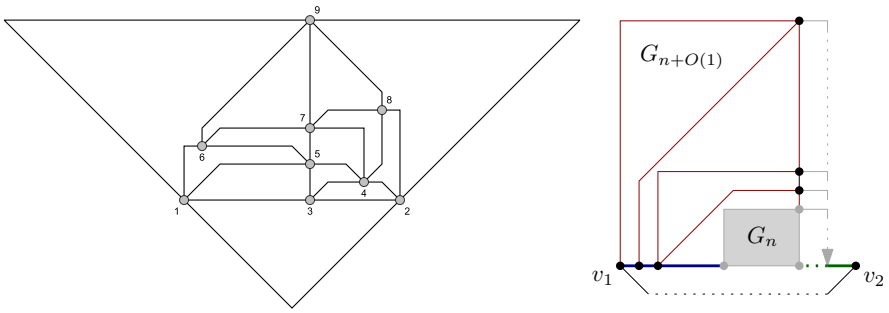
Figure 9: (a) Horizontal placement of a chain $P_k = \{v_i, \ldots, v_j\}$. (b) Placement of a singleton $P_k = \{v_i\}$ of degree 5. (c) Final layout (the shape of the dotted edges can be obtained by extending the stubs until they intersect).

5.1 The Triconnected Case

Let $G = (V, E)$ be a triconnected 5-planar graph and $\Pi = \{P_0, \ldots, P_m\}$ be a canonical order of G. We place the first two partitions P_0 and P_1 of Π, similar to the case of 4-planar graphs. Again, we assume that we have already constructed a drawing for G_{k-1} and now we have to place P_k, for some $k = 2, \ldots, m-1$. We further assume that the x- and y-coordinates are computed simultaneously so that the drawing of G_{k-1} is planar and horizontally stretchable in the following sense: If $e \in E(G_{k-1})$ is an edge incident to the outer face of G_{k-1}, then there is always a cut which crosses e and can be utilized to horizontally stretch the drawing of G_{k-1}. This is guaranteed by our construction which makes sure that in each step the edges incident to the outer face have a horizontal segment. In other words, one can define a cut through every edge incident to the outer face of G_{k-1} (*stretchability invariant*).

If $P_k = \{v_i, \ldots, v_j\}$ is a chain, it is placed exactly as in the case of 4-planar graphs, but with different port assignment. Recall that by v'_i (v'_j, resp.) we denote the neighbor of v_i (v_j, resp.) in G_{k-1}. Among the available northern ports of vertex v'_i (v'_j, resp.), edge (v_i, v'_i) ((v_j, v'_j), resp.) uses the eastern-most unoccupied port of v'_i (western-most unoccupied port of v'_j, resp.); see Fig.9a. If P_k does not fit into the gap between its two adjacent vertices v'_i and v'_j in G_{k-1}, then we horizontally stretch G_{k-1} between v'_i and v'_j to ensure that the horizontal distance between v'_i and v'_j is at least $|P_k| + 1$. This can always be accomplished due to the stretchability invariant, as both v'_i and v'_j are on the outer face of G_{k-1}. Potential crossings introduced by edges of P_k containing diagonal segments can be eliminated by employing similar cuts to the ones presented in the case of 4-planar graphs. So, we may assume that G_k is plane. Also, G_k complies with the stretchability invariant, as one can define a cut that crosses any of the newly inserted edges of P_k and then follows one of the cuts of G_{k-1} that crosses an edge between v'_i and v'_j.

In case of a singleton $P_k = \{v_i\}$ of degree 3 or 4, our approach is very similar to the one of the case of 4-planar graphs. Here, we mostly focus on the case where v_i is of degree 5. In this case, we have to deal with two additional edges (called *nested*) that connect v_i with G_{k-1}, say (v_i, v) and (v_i, v'); see Fig.9b. Such a pair of edges does not always allow vertex v_i to be placed along the next available horizontal grid line; v_i's position is more or less prescribed, as each of v and v' may have only one northern port unoccupied. However, a careful case analysis on the type of ports (i.e., north-west,



(a) (b)

Figure 10: (a) A sample drawing of a triconnected 5-planar graph produced by an implementation of our algorithm. (b) A recursive construction of an infinite class of 5-planar graphs requiring super-polynomial drawing area.

north or north-east) that are unoccupied at v and v' in conjunction with the fact that G_{k-1} is horizontally stretchable shows that we can always find a feasible placement for v_i (usually far apart from G_{k-1}); see e.g. Fig.9b. Potential crossings due to the remaining edges incident to v_i are eliminated by employing similar cuts to the ones presented in the case of 4-planar graphs. So, we may assume that G_k is planar. Similar to the case of a chain, we prove that G_k complies with the stretchability invariant. In this case special attention should be paid to avoid crossings with the nested edges of v_i, as a nested edge may contain no horizontal segment. Note that the case of the last partition $P_m = \{v_n\}$ is treated in the same way, even if v_n is potentially incident to three nested edges; see Fig.9c.

To complete the description of our approach it remains to describe how edge (v_1, v_2) is drawn. By construction both v_1 and v_2 are on a horizontal line. So, (v_1, v_2) can be drawn using two diagonal segments that form a bend pointing downwards; see Fig.9c. For a sample drawing produced by an implementation of our algorithm see Fig. 10a.

Theorem 5 *Given a triconnected* 5*-planar graph* G*, we can compute in* $O(n)$ *time an octilinear drawing of* G *with at most one bend per edge.*

Proof: In contrast to the corresponding proof for 4-planar graphs (see Theorem 1), in the case of 5-planar graphs the x and y-coordinates are not independent. However, since the y-coordinates of the vertices that have been placed already do not change afterwards, we can still use the shifting method of Kant and keep the running time of our algorithm linear. More specifically, in order to determine the y-coordinate of a singleton vertex, we only use the x-distances between its neighbors that have already been drawn. The x-distances can be computed in time proportional to the length of the path connecting them on the outer face of the graph in the tree structure of Kant. Since each such computation will not involve the same set of vertices more than once, the total time needed is linear (with respect to the number of the graph's vertices). Note that the aforementioned procedure is not necessary for chains, as they do not impose restriction on the drawing's height. □

Recall that when placing a singleton $P_k = \{v_i\}$ that has four edges to G_{k-1}, the height of G_k is determined by the horizontal distance of its neighbors along the outer face of G_{k-1}, which is bounded by the actual width of the drawing of G_{k-1}. On the other hand, when placing a chain P_k the amount of horizontal stretching required in order to avoid potential crossings is delimited by the height of the drawing of G_{k-1}. Unfortunately, this connection implies that for some input triconnected 5-planar graphs our drawing algorithm may result in drawings of super-polynomial area, as the following theorem states.

Theorem 6 *There exist infinitely many triconnected* 5*-planar graphs for which our drawing algorithm produces drawings of super-polynomial area.*

Proof: Fig.10b illustrates a recursive construction of an infinite class of 5-planar triconnected graphs with this property. The base of the construction is a "long chain" connecting v_1 and v_2 (refer to the bold drawn edges of Fig.10b). Each next member, say $G_{n+O(1)}$, of this class is constructed by adding a constant number of vertices (colored black in Fig.10b) to its immediate predecessor member, say G_n, of this class, as illustrated in Fig.10b. If W_n and H_n is the width and the height of G_n, respectively, then it is not difficult to show that $W_{n+O(1)} > 2W_n$ and $H_{n+O(1)} > 2H_n$, which implies that the required area is asymptotically exponential. □

5.2 The Biconnected Case

For the 4-planar case we defined invariants in order to keep the area of the resulting drawings polynomial. Since we drop this requirement now we can define a (simpler) new invariant for the biconnected 5-planar case. When considering a node μ in $\mathcal{T}$ and its poles $\mathcal{P}_\mu = \{s, t\}$, then in the drawing of G_μ^{pert}, s and t are horizontally aligned at the bottom of the drawing's bounding box as in Fig.11a. If an (s, t)-edge is present, it can be drawn at the bottom. An (s, t)-edge only occurs in the pertinent graph of a P-node (and Q-node). Again, we use the term *fixed* for a pole-node that is not allowed to form a nose. We maintain the following properties through the recursive construction process: In S- and R- nodes, s and t are not fixed. In P- and Q-nodes, only one of them is fixed, say s. But similar to the 4-planar biconnected case, we may swap their roles.

P-node case: Let μ be a P-node. It is not difficult to see that μ has at most 4 children; one of them might be a Q-node, i.e., an (s, t)-edge, which can be drawn at the bottom as a horizontal segment. Since P-nodes are not adjacent to each other in $\mathcal{T}$, the remaining

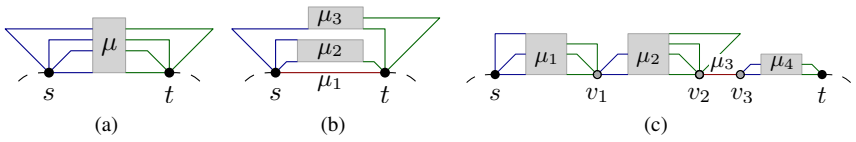


Figure 11: (a) Layout specification; s and t are located at the bottom. (b) P-node with an (s, t)-edge from a Q-node μ_1. s and t form a nose in μ_2, μ_3. (c) S-node example with four children $\mu_1, \ldots, \mu_4$.

children are S- or R-nodes. By our invariant we may form noses enabling us to stack them as in Fig.11b, as s and t are not fixed in them.

S-node case: Let μ be an S-node with children $\mu_1, \ldots, \mu_l$. Instead of the diagonal layout used earlier, we now align the drawings horizontally; see Fig.11c. In the S-node case, the poles inherit their pertinent degree from the children and the same holds for the property of being fixed. However, by our new invariant this is forbidden, as it clearly states that s and t are not fixed. It is easy to see that when μ_1 is a P-node, s is fixed by the invariant in μ_1. In this case, we swap the roles of the poles in μ_1 such that s is not fixed. However, the other pole of μ_1, say v_1, is fixed now. Since the skeleton of an S-node is a cycle of length at least three, $v_1 \neq t$ holds. As a result, both s and t are not fixed in the resulting drawing.

R-node case: To compute a layout of an R-node, we employ the triconnected algorithm (with $s = v_1$ and $t = v_2$). So, let μ be an R-node and μ_e a child of μ that corresponds to the virtual edge $e = (u, v)$ in G_μ^{skel}. Then, $deg_{\mu_e}^{pert}(u) \leq 3$ and $deg_{\mu_e}^{pert}(v) \leq 3$ holds. When inserting the drawing of $G_{\mu_e}^{pert}$, we require at most three consecutive ports at u and v for the additional edges. As the triconnected algorithm assigns ports in a consecutive manner based on the relative position of the endpoints, we modify the port assignment so that an edge may have more than one port assigned. To do so, we assign each edge $e = (u, v)$ in G_μ^{skel} a pair $(deg_{\mu_e}^{pert}(u), deg_{\mu_e}^{pert}(v)) \in \{1, 2, 3\}^2$ that reflects the number of ports required by this edge at its endpoints. Then, we extend the triconnected algorithm such that when a port of u is assigned to an edge $e = (u, v)$, $deg_{\mu_e}^{pert}(u) - 1$ additional consecutive ports in clockwise or counterclockwise order are reserved. The direction depends on the different types of edges that we will discuss next.

The simplest type of edges are the ones among consecutive vertices v_i, v_{i+1} of a chain. Recall that $P_0 = \{v_1, v_2\}$ is a special case and the edge (v_1, v_2) is drawn differently. Also, the edges from P_0 to P_1 are drawn as horizontal segments; see Fig.9c. For each such edge we reserve the additional ports at v_i in counter-clockwise order and at v_{i+1} in clockwise order; see Fig.12a. So, we can later plug the drawings of the children into the layout as in Fig.12b without forming noses. The second type of edges are the ones that connect $P_k = \{v_i, \ldots, v_j\}$ to v_i' and v_j' in G_{k-1}. No matter if P_k is a singleton or a chain, we proceed by reserving the ports as in the previous case, i.e., at v_i clockwise, (v_j counter-clockwise, resp.) and at v_i' counter-clockwise (v_j' clockwise); see Fig.12c. In case where (v_i, v_i') or (v_j, v_j') is a virtual edge, we choose the poles such that v_i (v_j resp.) is fixed in $\mu_{(v_i, v_i')}$ ($\mu_{(v_j, v_j')}$ resp.). Thus, we can create a nose with v_i' (v_j' resp.). Having exactly the ports required at both endpoints, we insert the drawing by replacing the bend with a nose as in Fig.12d. The remaining edges from P_k to G_{k-1} in case of a singleton $P_k = \{v_i\}$ can be handled similarly; see Fig.12. Notice that during the replacement of the edges, the fixed vertex is always the upper one. The only exception are the horizontally drawn edges of a chain. There, it does not matter which one is fixed, as none of the poles has to form a nose.

Root case: We root $\mathcal{T}$ at an arbitrarily chosen Q-node representing a real edge (s, t). By our invariant we may construct a drawing with s and t at the bottom of the drawing's bounding box, hence, we draw the edge (s, t) below the bounding box with a ninety degree bend using the south east port at s and south west port at t.

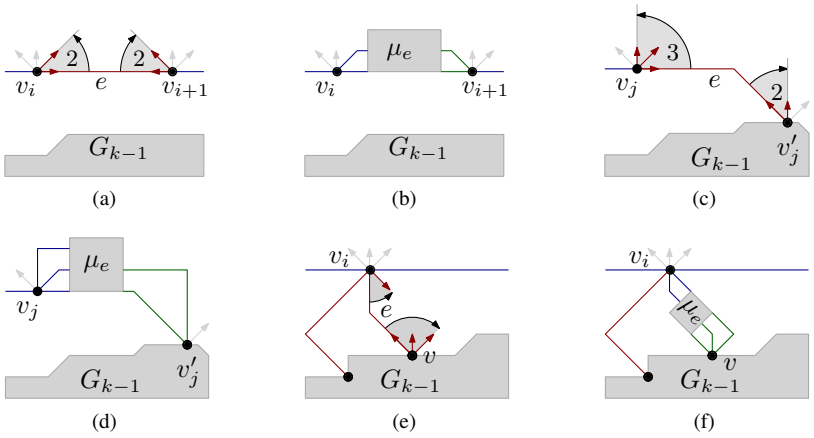


Figure 12: (a) Virtual edge $e = (v_i, v_{i+1})$ connecting two consecutive vertices of a chain. At both endpoints the drawing of μ_e requires two ports. (b) Replacing e in (a) with the corresponding drawing of the child μ_e. (c) Example of an edge $e = (v_j, v'_j)$ that requires three ports at v_j and two at v'_j. (d) Inserting the drawing of μ_e into (c) with v_j being fixed and v'_j forming a nose. (e) Reserving ports for the nested edges. A single port for a real edge is reserved and then two ports for the virtual edge e = (v_i, v). (f) Final layout after inserting the drawing of μ_e.

Theorem 7 *Given a biconnected* 5*-planar graph* G*, we can compute in* $O(n)$ *time an octilinear drawing of* G *with at most one bend per edge.*

Proof: We have shown that the ability to rotate and scale suffices to extend the result from 4-planar to 5-planar at the expense of the area. Similar to the 4-planar case, computing $\mathcal{T}$ takes linear time. Hence, the overall running time is governed by the triconnected algorithm. □

5.3 The Simply-Connected Case

In the following, we only outline the differences in comparison with the corresponding 4-planar case. As an invariant, the drawing of every subtree should conform to the layout depicted in Fig.13a. For a single biconnected component b, let v_c refer to the cut vertex linking it to the parent. As root for the SPQR-tree $\mathcal{T}_b$ of G_b, we again choose a Q-node μ_r whose real edge is incident to v_c; see Fig.13b. Hence, the layout generated by the biconnected approach matches this scheme.

It remains to show that we can attach the children. Since we are able to scale and rotate, we keep things simple and look for suitable spots to attach them. Notice that in the drawings of S-nodes and chains in R-nodes all southern ports are free. Hence, we may rotate the drawings of the subtrees and attach the at most three (two for a chain) edges to v_c there (refer to Fig.13c for an example of a chain). The only exception are the singletons. Assume that v_i is an ordinary singleton that has one nested edge attached. Hence, it has degree 4, leaving us with a single bridge to attach the component; Fig.13d.

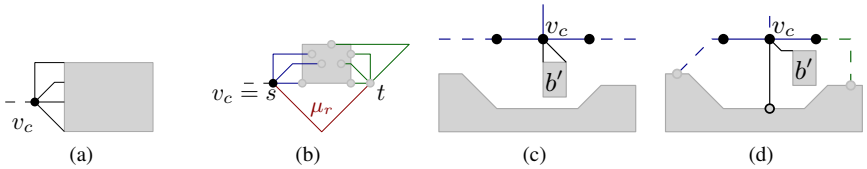


Figure 13: (a) Layout scheme for a BC-subtree rooted at v_c. (b) Rooting $\mathcal{T}_b$ at a Q-node μ_r. (c) Attaching a subtree at a chain and in (d) at a singleton inside an R-node.

However, this does not hold in case $v_i = v_n$. Consider the case where v_n has a nested edge and we have to attach a subtree that requires two ports. As a result v_n has degree 3 in G_b and, thus, all northern ports are free.

Theorem 8 *Given a connected 5-planar graph G, we can compute in $O(n)$ time an octilinear drawing of G with at most one bend per edge.*

Proof: We described how to attach any subtree to cut vertices inside a biconnected component. Furthermore, the component itself complies with the layout scheme. In addition, this scheme enables us to compose such drawings at a cut vertex using rotations. The running time follows from the fact that the decomposition of a connected graph into its biconnected components takes linear time. □

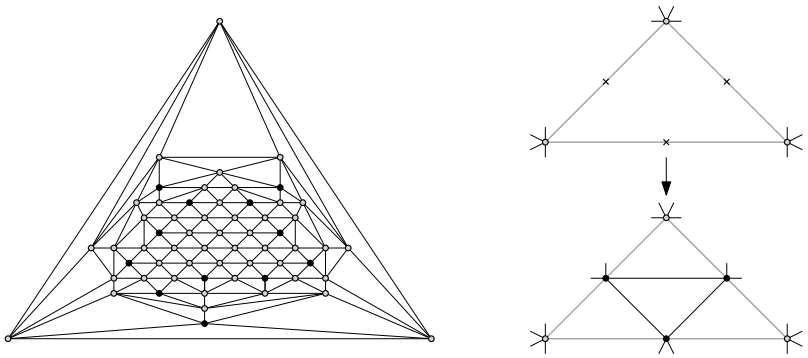
6 A Note on Octilinear Drawings of 6-Planar Graphs

In this section, we present an infinite class of 6-planar graphs that do not admit planar octilinear drawings with at most one bend per edge.

Theorem 9 *There exists an infinite class of 6-planar graphs which do not admit planar octilinear drawings with at most one bend per edge.*

Proof: Our proof is heavily based on the following simple observation: If the outer face $\mathcal{F}(\Gamma(G))$ of a given planar octilinear drawing $\Gamma(G)$ consists of exactly three vertices, say v, v' and v'', that have the so-called *outerdegree-property*, i.e., $deg(v) = deg(v') = 6$ and $5 \leq deg(v'') \leq 6$, then it is not feasible to draw all edges delimiting $\mathcal{F}(\Gamma(G))$ with at most one bend per edge; one of them has to be drawn with (at least) two bends in $\Gamma(G)$. Next, we construct a specific maximal 6-planar graph, in which each face has at most one vertex of degree 5 and at least two vertices of degree 6; see Fig.14a. This specific graph does not admit a planar octilinear drawing with at most one bend, as its outer face is always bounded by three vertices that have the outerdegree-property.

To obtain an infinite class, say $\mathcal{C}$, of 6-planar graphs with this property, we give the following iterative construction. The first member of $\mathcal{C}$ is the graph of Fig.14a. For $i = 1, 2, \ldots$, the i-th member of $\mathcal{C}$ is constructed by first subdividing all edges of the $(i-1)$-th member of $\mathcal{C}$. This yields a new planar graph, in which every face consists of three vertices of degree 5 or 6 and three vertices of degree 2. We connect the three vertices of degree 2 of each face of the implied graph such that they form a K_3; see



(a) (b)

Figure 14: (a) A maximal 6-planar graph in which each face has at most one vertex of degree 5 (black-colored vertices) and at least two vertices of degree 6 (gray-colored vertices). From Euler's formula for maximal planar graphs, it follows that any graph with this property must have at least 12 vertices of degree 5. Hence, this is a smallest graph with this property. (b) Illustration of the recursive construction.

Fig.14b. Observe that in this way all new vertices are of degree 6. This implies that the result is a 6-planar fully-triangulated graph which contains the same number of degree 5 vertices as the graph of Fig.14a and has the outerdegree-property. □

7 Conclusions

Motivated by their significance in map schematization, we presented algorithms for creating planar octilinear drawings with at most one bend per edge for 4- and 5-planar graphs. We improved the known bounds on the required number of slopes for 4- and 5-planar drawings from 8 and 10 ([12]) to 4. Our work raises several open problems:

- Is it possible to construct planar octilinear drawings of 4-planar (5-planar, resp.) graphs with at most one bend per edge in $o(n^3)$ (polynomial, resp.) area?
- Does any triangle-free 6-planar graph admit a planar octilinear drawing with at most one bend per edge?
- What is the complexity to determine whether a 6-planar graph admits a planar octilinear drawing with at most one bend per edge?
- What is the number of necessary slopes for bendless drawings of 4-planar graphs?
- Is it possible to extend our methods to non-planar graphs?

Acknowledgement: The authors would like to thank Findan Eisenhut who implemented the algorithms for the triconnected 4- and 5-planar graphs and reduced the running time of the 5-planar triconnected algorithm from $O(n^2)$ to $O(n)$.

References

[1] T. C. Biedl and G. Kant. A better heuristic for orthogonal graph drawings. *Comput. Geom.*, 9(3):159–180, 1998. doi:10.1016/S0925-7721(97)00026-6.

[2] T. Bläsius, M. Krug, I. Rutter, and D. Wagner. Orthogonal graph drawing with flexibility constraints. *Algorithmica*, 68(4):859–885, 2014. doi:10.1007/s00453-012-9705-8.

[3] H. L. Bodlaender and G. Tel. A note on rectilinearity and angular resolution. *J. Graph Algorithms Appl.*, 8:89–94, 2004. doi:10.7155/jgaa.00083.

[4] G. Di Battista and R. Tamassia. On-line graph algorithms with SPQR-trees. In M. Paterson, editor, *Automata, Languages and Programming*, volume 443 of *LNCS*, pages 598–611. Springer Berlin Heidelberg, 1990. doi:10.1007/BFb0032061.

[5] 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.

[6] E. D. Giacomo, G. Liotta, and F. Montecchiani. The planar slope number of subcubic graphs. In A. Pardo and A. Viola, editors, *LATIN*, volume 8392 of *Lecture Notes in Computer Science*, pages 132–143. Springer, 2014. doi:10.1007/978-3-642-54423-1_12.

[7] C. Gutwenger and P. Mutzel. A linear time implementation of spqr-trees. In J. Marks, editor, *Graph Drawing*, volume 1984 of *Lecture Notes in Computer Science*, pages 77–90. Springer, 2000. doi:10.1007/3-540-44541-2_8.

[8] S. Hong, D. Merrick, and H. A. D. do Nascimento. Automatic visualisation of metro maps. *J. Vis. Lang. Comput.*, 17(3):203–224, 2006. doi:10.1016/j.jvlc.2005.09.001.

[9] V. Jelínek, E. Jelínková, J. Kratochvíl, B. Lidický, M. Tesar, and T. Vyskocil. The planar slope number of planar partial 3-trees of bounded degree. *Graphs and Combinatorics*, 29(4):981–1005, 2013. doi:10.1007/s00373-012-1157-z.

[10] G. Kant. Drawing planar graphs using the lmc-ordering (extended abstract). In *FOCS*, pages 101–110. IEEE Computer Society, 1992. doi:10.1109/SFCS.1992.267814.

[11] G. Kant. Hexagonal grid drawings. In E. W. Mayr, editor, *WG*, volume 657 of *Lecture Notes in Computer Science*, pages 263–276. Springer, 1992. doi:10.1007/3-540-56402-0_53.

[12] B. Keszegh, J. Pach, and D. Pálvölgyi. Drawing planar graphs of bounded degree with few slopes. *SIAM Journal of Discrete Mathematics*, 27(2):1171–1183, 2013. doi:10.1137/100815001.

[13] B. Keszegh, J. Pach, D. Pálvölgyi, and G. Tóth. Drawing cubic graphs with at most five slopes. *Computational Geometry*, 40(2):138–147, 2008. doi:10.1016/j.comgeo.2007.05.003.

[14] W. Lenhart, G. Liotta, D. Mondal, and R. I. Nishat. Planar and plane slope number of partial 2-trees. In S. K. Wismath and A. Wolff, editors, *GD*, volume 8242 of *Lecture Notes in Computer Science*, pages 412–423. Springer, 2013. doi:10.1007/978-3-319-03841-4_36.

[15] P. Mukkamala and D. Pálvölgyi. Drawing cubic graphs with the four basic slopes. In M. J. van Kreveld and B. Speckmann, editors, *Graph Drawing*, volume 7034 of *Lecture Notes in Computer Science*, pages 254–265. Springer, 2011. doi:10.1007/978-3-642-25878-7_25.

[16] M. Nöllenburg. Automated drawings of metro maps. Technical Report 2005-25, Fakultät für Informatik, Universität Karlsruhe, 2005.

[17] M. Nöllenburg and A. Wolff. Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):626–641, 2011. doi:10.1109/TVCG.2010.81.

[18] J. M. Stott, P. Rodgers, J. C. Martinez-Ovando, and S. G. Walker. Automatic metro map layout using multicriteria optimization. *IEEE Trans. Vis. Comput. Graph.*, 17(1):101–114, 2011. doi:10.1109/TVCG.2010.24.

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

[20] A. Wolff. Graph drawing and cartography. In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*, chapter 23, pages 697–736. CRC Press, 2013.