Planar Polyline Drawings with Good Angular Resolution*

Carsten Gutwenger[1] and Petra Mutzel[2]

[1] Max-Planck-Institut für Informatik
Saarbrücken, Germany, gutwenge@mpi-sb.mpg.de
[2] Max-Planck-Institut für Informatik
Saarbrücken, Germany, mutzel@mpi-sb.mpg.de

Abstract. We present a linear time algorithm that constructs a planar polyline grid drawing of any plane graph with n vertices and maximum degree d on a $(2n-5) \times (\frac{3}{2}n - \frac{7}{2})$ grid with at most $5n-15$ bends and minimum angle $> \frac{2}{d}$. In the constructed drawings, every edge has at most three bends and length $O(n)$. To our best knowledge, this algorithm achieves the best simultaneous bounds concerning the grid size, angular resolution, and number of bends for planar grid drawings of high-degree planar graphs. Besides the nice theoretical features, the practical drawings are aesthetically very pleasing. An implementation of our algorithm is available with the AGD-Library (Algorithms for Graph Drawing) [2, 1]. Our algorithm is based on ideas by Kant for polyline grid drawings for triconnected plane graphs [23]. In particular, our algorithm significantly improves upon his bounds on the angular resolution and the grid size for non-triconnected plane graphs. In this case, Kant could show an angular resolution of $\frac{4}{3d+7}$ and a grid size of $(2n-5) \times (3n-6)$, only.

1 Introduction

In automatic graph drawing, the task is to visualize discrete structures so that they are easy to read and to understand. Applications include drawing of entity-relationship diagrams, PERT-diagrams, and flow-diagrams (see, e.g., [31, 22]). For a survey on graph drawing, see e.g., [10, 12].

Important aesthetic criteria for nice drawings are: a small number of edge crossings, small area, few bends, and a good angular resolution (see, e.g., [27]). Unfortunately, the problem of optimizing the above criteria is NP-hard, even if we restrict ourselves to subsets of the criteria (see, e.g., [15, 25, 7, 29, 17, 16]). E.g., it is NP-complete to decide whether a graph can be embedded in a grid of prescribed area [25] even if we restrict ouselves to planar orthogonal grid drawings of trees [7]. Furthermore, it is NP-hard to compute a drawing with the minimal number of crossings [15]. However, it is possible to test whether a graph can be drawn without crossings in linear time [21]. In this paper, we restrict our attention to planar graphs.

* Partially supported by DFG-Grant Mu 1129/3-1, Forschungsschwerpunkt "Effiziente Algorithmen für diskrete Probleme und ihre Anwendungen".

S.H. Whitesides (Ed.): GD'98, LNCS 1547, pp. 167–182, 1998.

It is well-known that planar n-vertex graphs can be drawn without any bends on a grid of size $(n-2) \times (n-2)$ [9, 23, 8]. However, the quality of these planar straight-line drawings is not sufficient in practice. One reason for this is that the angles of the drawings get too small. Indeed, they achieve size $\Omega(\frac{1}{n^2})$.

The term *angular resolution* denotes the size of the minimum angle formed by any two edge segments sharing a common point in a drawing. Unfortunately, one can show that a good angular resolution in planar straight-line drawings can be achieved only at the cost of the area taken up by the drawing. Garg and Tamassia [18] have shown that a class of planar graphs exists that require exponential area in any planar straight-line drawing with optimal $\Omega(1)$ angular resolution.

Drawings with good angular resolution and small grid size are the so-called orthogonal (grid) drawings. Here, only horizontal and vertical line segments are used for representing the edges. Hence, only k-multiples of 90°-degree angles for $k \in \{1, 2, 3, 4\}$ occur. While minimizing the number of bends in an orthogonal drawing of a 4-planar graph G is NP-hard [17], the problem can be solved in polynomial time when the embedding of G is fixed. Tamassia has presented an algorithm that constructs a planar orthogonal grid drawing of an embedded planar graph with maximum degree four with the minimum number of bends [30]. It can be shown that the size of the drawing is bounded by $(n+1) \times (n+1)$ [3] and the number of bends by $2n + 4$ if the graph is biconnected. However, in practice, the algorithm by Tamassia produces drawings of better quality on average [11]. Its running time is $O(n^{\frac{7}{4}}\sqrt{\log n})$ [19].

Good bounds can also be achieved by the linear time algorithms suggested in [5, 32, 33]. More precisely, for biconnected plane graphs, the constructed drawings need a $n \times n$ grid and have at most $2n + 4$ bends [32, 33]. For non-biconnected plane graphs, the bounds on the grid size and the number of bends are $1.2n \times 1.2n$ and $2.4n + 2$, respectively [3]. This can be improved to $n \times n$ and $2n$, respectively, if change is permitted to the embedding of the graph [5].

Unfortunately, all these algorithms work for planar graphs with maximum degree four only. For planar graphs with higher degree, several extensions have been suggested:

(1) drawing the nodes as boxes [31, 6],
(2) introducing two grids: a coarse grid for the nodes and a finer grid for the line segments (Kandinsky model) [14], and
(3) allowing locally non-orthogonal line segments [24, 23].

The disadvantage of approach (1) in [31] is that the sizes of the nodes may grow very large, independently of their degree. Biedl [4] suggested a proportional-growth model, in which an increase in the sizes of the boxes is allowed only according to the degree of the corresponding node. She also presents a linear time algorithm that constructs a planar orthogonal drawing in the proportional-growth model for any triconnected plane graph. In these drawings, the grid size is bounded by $(2n-5) \times (1.5n-3)$ and the number of bends by $2n-6$ [4].

If the user prefers equally sized nodes, approaches (2) and (3) are the method of choice. However, the resolution in these drawings is not always sufficient.

While in drawings of approach (2), line segments occur very close to each other, in the drawings of approach (3), the angles may get too small. Biedl [4] has given a linear time algorithm that constructs a planar orthogonal drawing in the Kandinsky model of any planar graph in a grid of size $(3n-7) \times (1.5n-3)$ with at most $3n-8$ bends. It can be shown that any drawing constructed in the Kandinsky model can be transformed into the proportional-growth model without introducing additional bends [14].

Kant [23] has suggested a linear time algorithm that constructs a planar polyline grid drawing on a $(2n-5) \times (3n-6)$ grid with at most $5n-15$ bends for a triconnected plane (simple) graph $G=(V,E)$. The important point here is that the minimum angle is at least $\frac{2}{d}$, where d denotes the maximum degree of a node in V. For connected planar graphs, the size of the smallest angle is bounded by $\frac{4}{3d+7}$ using the result that every plane graph G can be augmented by adding edges to a triconnected plane graph G' so that $d' \leq \lceil \frac{3}{2}d \rceil + 3$, with d and d' the maximum degree of G and G', respectively. In these drawings, every edge has at most three bends and length $\mathrm{O}(n)$. Kant called his algorithm the "mixed model" algorithm, since he used a framework for straight-line planar drawings based on the shelling order (also called canonical ordering) for triconnected plane graphs for constructing the polyline drawings. Instead of placing single vertices as done in the straight-line algorithms, so-called boxes of vertices are placed. A box of a vertex v consists of the vertex itself and the segments representing the first parts of incident edges to v.

Our new algorithm is based on Kant's ideas. Here, we use a shelling order for biconnected plane graphs introduced in [20] for achieving straight-line drawings of biconnected graphs. Based on this ordering, our definition of the bounding boxes of the vertices is similar to that of Kant. Then we use a placement algorithm in order to place the bounding boxes as in the straight-line drawing algorithms [20]. This gives a linear time and space algorithm that constructs a planar polyline grid drawing of a connected (simple) planar graph with n vertices and maximum degree d on a $(2n-5) \times (\frac{3}{2}n - \frac{7}{2})$ grid with at most $5n-15$ bends and minimum angle $> \frac{2}{d}$, in which every edge has at most three bends.

Note that our algorithm leads to a big improvement concerning the minimum angle and the grid size compared to Kant's work. To our best knowledge, this algorithm achieves the best simultaneous bounds concerning the grid size, angular resolution, and number of bends for planar drawings of high-degree planar graphs. Besides the nice theoretical features, the practical drawings are aesthetically very pleasing. An implementation of our algorithm is available with the AGD-Library (Algorithms for Graph Drawing) [2, 1].

The paper is organized as follows: Section 2 contains mathematical preliminaries. The algorithm is presented in Sect. 3. First, we introduce the shelling order for biconnected plane graphs from [20], then we define the bounding boxes and finally describe the placement algorithm. In Sect. 4, we analyze the algorithm: its correctness, the linear running time, the bounds on the grid size, the size of the minimum angle, and the bounds on the number of bends. Here, we also show that the bounds on the grid size are tight. Some practical aspects are discussed in Sect. 5. In particular, we try to locally save edge bends and some

grid lines. Although, in practice, most of the drawings improve significantly, the theoretical bounds do not improve. In the final section, we give a summary and discuss some open problems.

2 Mathematical Preliminaries

Let $G = (V, E)$ be an undirected graph. We let (v, w) denote an edge connecting vertices v and w, i.e., (v, w) and (w, v) denote the same edge. The vertices v and w are the two *endpoints* of the edge (v, w). We call G *simple* if G contains neither multiple edges nor self loops. Throughout this paper, we only consider simple graphs. A *walk* in G is a sequence $v_0 e_1 v_1 \dots e_k v_k$ of alternating vertices and edges, such that the endpoints of e_i are v_{i-1} and v_i for $1 \leq i \leq k$. We usually write $v_0, \dots, v_k$ for short. A walk is called *circular* if $v_0 = v_k$. A *path* is a walk in which each vertex appears at most once. A path is a *chain* in G if each vertex on the path has degree two in G.

A *planar drawing* of a graph G is a drawing of G in the plane without edge crossings. A graph is called *planar* if it allows for a planar drawing. It is called *plane* if it is associated with a planar drawing. An *embedding* of a planar graph G is the collection of circular orderings of the incident edges (or adjacent vertices) around each vertex in a planar drawing of G. For a vertex v of G, we will refer to the *counter-clockwise circular ordering* around v. A plane graph divides the plane into regions called *faces*. The clockwise boundary of a face f is a circular walk $v_0 e_1 v_1 \dots e_k v_k$ along the edges bounding f such that v_{i+1} is the successor of v_{i-1} in the circular counter-clockwise ordering around v_i. The *counter-clockwise boundary* is defined analogously, where v_{i+1} is the predecessor of v_{i-1} in the ordering around v_i. The unbounded face is called the *exterior face*; all other faces are called *interior faces*. Edges and vertices belonging to the exterior face are called *exterior edges* and *exterior vertices*, respectively; all other edges and vertices are called *interior edges* and *interior vertices*. It is well-known that a planar graph with n vertices contains at most $3n - 6$ edges.

Let $W, W' \subseteq V$ be two disjoint sets of vertices of $G = (V, E)$. We denote by $E(W, W')$ the set of edges with one endpoint in W and the other endpoint in W', and by $E(W)$ the set of edges with both endpoints in W. The subgraph of G *induced* by the vertices in $W \subseteq V$ consists of the vertices in W and all edges in $E(W)$. A connected graph G is called *k-connected* if it is still connected after deleting any set of $k - 1$ vertices. For $k = 2$ and $k = 3$, we also say *biconnected* and *triconnected*, respectively.

3 The Algorithm

In this section we describe our new drawing algorithm. Suppose the input graph $G = (V, E)$ is plane and simple. The idea of the algorithm is to assign *inpoints* and *outpoints* to each edge of G. We draw every edge $e = (v, w)$ as polyline that goes from v to the outpoint u_{out} of e. Then, from u_{out} vertically to a point, say

u. From u horizontally to the inpoint u_{in} of e, and finally from u_{in} to w. Thus, each edge gets at most three bends.

Algorithm `Mixed-Model` works in three phases. In the first phase, we compute an ordered partition $\pi = V_1, \ldots, V_N$ of the vertices in G, that is, $V_1 \cup \ldots \cup V_N = V$ and $V_i \cap V_j = \emptyset$ for $i \neq j$. The *rank* of a vertex v is the index i of the set V_i containing v. In the second phase, we determine the positions of the in- and outpoints. For each vertex $v \in V$, we define the inpoints of all edges (v, w) with $\mathit{rank}(w) \leq \mathit{rank}(v)$, and the outpoints of all edges (v, w) with $\mathit{rank}(w) > \mathit{rank}(v)$ relative to v. In the third phase, we finally compute the coordinates of the vertices. We build the drawing incrementally by adding the vertices in $V_1, \ldots, V_N$ in the order given by π. The ordering π guarantees that we always add vertices lying in the exterior face of the current drawing. Once we have the vertex coordinates, the polylines for the edges are given by the in- and outpoints. In the following, we discuss each step in detail.

3.1 Computation of the Ordered Partition

Here, we explain how to compute the ordered partition $\pi = V_1, \ldots, V_N$. Let G_k be the plane subgraph of G induced by $V_1 \cup \ldots \cup V_k$. We want π to have the following properties:

(P1) For each set $V_k = \{z_1, \ldots, z_p\}$ there exist two vertices $\mathit{left}(V_k)$ and $\mathit{right}(V_k)$. Let $E_1(V_k) = \{(z_i, z_{i+1}) \mid 1 \leq i < p\}$ for $k \geq 1$, and $E_2(V_k) = \{(\mathit{left}(V_k), z_1), (z_p, \mathit{right}(V_k))\}$ for $k \geq 2$. Then, the set

$$S = \bigcup_{1 \leq k \leq N} E_1(V_k) \setminus E \cup \bigcup_{2 \leq k \leq N} E_2(V_k) \setminus E$$

can be inserted into the plane graph G without introducing any crossings. We denote with $\widetilde{G} = (V, \widetilde{E})$ the plane graph with $\widetilde{E} = E \cup S$.

(P2) $V_1 = \{v_1, \ldots, v_s\}$ is a path on the clockwise boundary of the exterior face of $\widetilde{G}$, and the vertices in V_k, $k \geq 2$, lie on the exterior face of $\widetilde{G}_k$.

(P3) We define C_1 to be the sequence of vertices $v_1, \ldots, v_s$. Consider a $k \geq 2$ and let $C_{k-1} = c_1, \ldots, c_q$ be already defined. Let $c_\ell = \mathit{left}(V_k)$ and $c_r = \mathit{right}(V_k)$. Then, we define C_k to be the sequence $c_1, \ldots, c_\ell, V_k, c_r, \ldots, c_q$.

(P4) Let $V_k = \{z_1, \ldots, z_p\}$.

(a) $\widetilde{E}(V_k) = \{(z_i, z_{i+1}) \mid 1 \leq i < p\}$

(b) Let $k \geq 2$, $C_{k-1} = c_1, \ldots, c_q$, $\mathit{left}(V_k) = c_\ell$, and $\mathit{right}(V_k) = c_r$. Then, all neighbors of V_k in $\widetilde{G}_{k-1}$ lie in $\{c_\ell, \ldots, c_r\}$. If $p \geq 2$ then V_k has exactly two neighbors in $\widetilde{G}_{k-1}$.

These properties facilitate an incremental approach of building the drawing by inserting step-by-step the vertices in V_k into the exterior face of G_{k-1}, and shifting the vertices right of and including $\mathit{right}(V_k)$ by some grid units to the right if necessary. Kant defined a suitable ordering for triconnected plane graphs in [23]. Figure 1(a) shows that it is not satisfying to augment the given plane

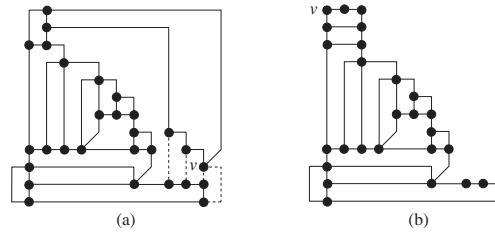

Fig. 1. The same graph drawn with augmentation (a) and with the new shelling order (b)

graph to a triconnected plane graph, and to then use the shelling order given by Kant. The graph depicted has been optimally augmented to a triconnected plane graph by adding the dashed edges. When these edges are removed, the resulting drawing looks strange. The reason is that vertex v has been placed before all of its neighbors. In [20] we have introduced a shelling order for biconnected plane graphs that guarantees that each set V_k, $k \geq 2$, has at least one neighbor in G_{k-1}. Figure 1(b) shows the same graph drawn using the new ordering.

Definition 1. Let $G = (V, E)$ be a biconnected plane graph with at least three vertices. Let $\pi = (V_1, \ldots, V_N)$ be an ordered partition of V, that is, $V_1 \cup \ldots \cup V_N = V$ and $V_i \cap V_j = \emptyset$ for $i \neq j$. Denote by G_k the plane subgraph of G induced by $V_1 \cup \ldots \cup V_k$. We say that π is a *shelling order* of G if the following conditions are satisfied:

(1.1) $V_1 = \{v_1, \ldots, v_s\}$, where $v_1, \ldots, v_s$ is a path on the clockwise boundary of the exterior face of G with $s \geq 2$ and $E(\{v_1, \ldots, v_s\}) = \{(v_i, v_{i+1}) \mid 1 \leq i < s\}$.

(1.2) Each G_k is connected and internally biconnected, that is, removing one interior vertex of G_k does not disconnect it.

(1.3) Denote by C'_k the walk on the counter-clockwise boundary of the exterior face of G_k from v_1 to v_s. For each $2 \leq k \leq N$ one of the following conditions holds, where $C'_k = [v_1 = c_1, \ldots, c_q = v_s]$:
  (a) $V_k = \{z\}$ and z is a vertex on C'_k with at least three neighbors in G_{k-1}.
  (b) $V_k = \{z_1, \ldots, z_p\} \subset C'_k$, $p \geq 1$, and there are vertices c_ℓ, c_r, $\ell < r$, on C'_k such that $c_\ell, z_1, \ldots, z_i, u_1, \ldots, u_j, z_{i+1}, \ldots, z_p, c_r$ is a path on the clockwise boundary of a face f_k of G for some $0 \leq i \leq p$, $j \geq 0$, $u_1, \ldots, u_j$ are vertices in $G \setminus G_k$ and $E(V_k, V_1 \cup \ldots \cup V_{k-1}) \subseteq \{(c_\ell, z_1), (z_p, c_r)\}$.

We will use this shelling order in our algorithm. Since our input graph G is not necessarily biconnected, we first augment it to a biconnected plane graph $G' = (V, E')$. This can be done optimally using the algorithm in [26]. However, if we are allowed to change the embedding, the algorithms by Fialko and Mutzel [13, 26] are the better choice. Then, we compute the shelling order $\pi = V_1, \ldots, V_N$ for G'. Consider a set $V_k = \{z_1, \ldots, z_p\}$ with $k \geq 2$. We need to determine $\mathit{left}(V_k)$ and $\mathit{right}(V_k)$ (see P1). Let $C_{k-1} = c_1, \ldots, c_q$, and $e_1, \ldots, e_a$ be the edges in

G' with $e_\mu = (v_\mu, c_{i_\mu})$ and $v_\mu \in V_k$, such that $i_1 < \cdots < i_a$. If $a \leq 2$, then V_k satisfies case (1.3)(b) and we set $\mathit{left}(V_k) := c_\ell$ and $\mathit{right}(V_k) := c_r$, where c_ℓ and c_r are the vertices that exist according to (1.3)(b).

If $a \geq 3$, we set $\mathit{left}(V_k) := c_\ell$ and $\mathit{right}(V_k) := c_r$ and determine ℓ and r as follows. Let $N = \{c_{i_\mu} \mid e_\mu \in E\}$. If $N = \emptyset$ we set $\ell := i_1$ and $r := i_a$. If $N = \{c_t\}$ we set

$$\ell := \left\{ \begin{array}{ll} t & \text{if } t \neq i_a \\ i_1 & \text{otherwise} \end{array} \right\} \text{ and } r := \left\{ \begin{array}{ll} t & \text{if } t \neq \ell \\ i_a & \text{otherwise} \end{array} \right\}$$

If $|N| \geq 2$ we set $\ell := \min\{\mu \mid c_\mu \in N\}$ and $\ell := \max\{\mu \mid c_\mu \in N\}$.

Note that we only needed the graph G' in order to compute a feasible ordered partition of G. In the following, we consider the given plane graph G together with the ordered partition π.

3.2 Assignment of In- and Outpoints

Recall that we assign an inpoint and an outpoint to each edge $e = (v, w)$. In order to compute the absolute coordinates of an in- or outpoint of e efficiently, we only determine coordinates which are relative to either v or w. Therefore, for each vertex v there is a set of inpoints and a set of outpoints whose coordinates are relative to v. These in- and outpoints form a so-called *bounding box* around v, which we define in the following.

We call the edges in $\{(v, w) \mid \mathit{rank}(w) \leq \mathit{rank}(v)\}$ the *incoming edges* of v, and the edges in $\{(v, w) \mid \mathit{rank}(w) > \mathit{rank}(v)\}$ the *outgoing edges* of v. We denote with $\mathit{in}(v)$ the number of incoming edges of v, and with $\mathit{out}(v)$ the number of outgoing edges of v. For a vertex v, we define the coordinates of the inpoints of all incoming edges of v, and the outpoints of all outgoing edges of v relative to the position of v. We point out that an edge (v, w) with $\mathit{rank}(v) = \mathit{rank}(w)$ is an incoming edge of v and w, and thus is assigned two inpoints and no outpoint. But in this case, the edge is drawn as one horizontal line segment and we will not refer to its in- or outpoint in the algorithm.

Let $V_k = \{z_1, \ldots, z_p\}$, $v = z_i$, and set $z_0 := \mathit{left}(V_k)$ and $z_{p+1} := \mathit{right}(V_k)$. We define $\mathit{out}_\ell(v)$, $\mathit{out}_r(v)$, δ_ℓ and δ_r according to the following table.

$\mathit{out}_\ell(v)$	δ_ℓ	$\mathit{out}_r(v)$	δ_r	**if**
$\mathit{out}^-(v)$	1	$\mathit{out}^+(v)$	1	$\mathit{in}(v) \geq 2$
$\mathit{out}^+(v)$	0	$\mathit{out}^-(v)$	1	$\mathit{in}(v) = 1$ and $(z_{i-1}, z_i) \notin E$
$\mathit{out}^-(v)$	1	$\mathit{out}^+(v)$	0	$\mathit{in}(v) = 1$ and $(z_i, z_{i+1}) \notin E$
$\mathit{out}^-(v)$	0	$\mathit{out}^+(v)$	0	$\mathit{in}(v) = 0$

where $\mathit{out}^+(v) = \left\lceil \frac{\mathit{out}(v)-1}{2} \right\rceil$ and $\mathit{out}^-(v) = \max\left(0, \left\lfloor \frac{\mathit{out}(v)-1}{2} \right\rfloor\right)$. If $\mathit{out}(v) \geq 1$, we place the outpoints on the following points:

- $\mathit{out}_\ell(v)$ outpoints on the points $(-\mathit{out}_\ell(v), \delta_\ell), \ldots, (-1, \mathit{out}_\ell(v) + \delta_\ell - 1)$, which lie on a straight line with slope $+1$.
- One outpoint on point $(0, \max\{\mathit{out}_\ell(v) + \delta_\ell - 1, \mathit{out}_r(v) + \delta_r - 1\})$.

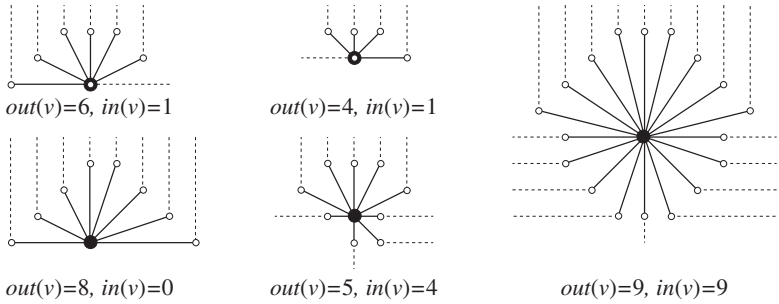

Fig. 2. Examples of bounding boxes

– $out_r(v)$ outpoints on the points $(1, out_r(v) + \delta_r - 1), \ldots, (out_r(v), \delta_r)$, which lie on a straight line with slope -1.

We set $in_\ell(v) := \max\left(0, \left\lfloor \frac{in(v)-3}{2} \right\rfloor\right)$ and $in_r(v) := \max\left(0, \left\lceil \frac{in(v)-3}{2} \right\rceil\right)$. If $in(v) \leq 3$, we set all inpoints on $(0, 0)$. Otherwise, we place the inpoints on the following points:

– One inpoint on point $(-in_\ell(v), 0)$.
– $in_\ell(v)$ inpoints on the points $(-in_\ell(v), -1), \ldots, (-1, -in_\ell(v))$, which lie on a straight line with slope -1.
– One inpoint on point $(0, -in_r(v))$.
– $in_r(v)$ inpoints on the points $(1, -in_r(v)), \ldots, (in_r(v), -1)$, which lie on a straight line with slope $+1$.
– One inpoint on point $(in_r(v), 0)$.

Figure 2 shows several examples of bounding boxes. The dashed lines indicate how vertical or horizontal segments are attached to in- and outpoints. We denote with $outpoint(e)$ or $outpoint(v, w)$ the outpoint belonging to edge $e = (v, w)$. For an outpoint u, we denote with $u.dx$ and $u.dy$ the relative x- and y-coordinate of u, respectively.

3.3 Computation of Coordinates

Next, we show how to compute the coordinates of the vertices. Within the algorithm, we maintain absolute y-coordinates and relative x-coordinates. Let $C = c_1, \ldots, c_q$ be the current contour as introduced in Sect. 3.1, i.e, $C = C_k$ after the kth step. For the x-coordinates

$$x(v) = \begin{cases} x_{abs}(v) & \text{if } v = c_1 \\ x_{abs}(c_i) - x_{abs}(c_{i-1}) & \text{if } v = c_i \text{ with } i \geq 2 \\ x_{abs}(v) - x_{abs}(father(v)) & \text{if } v \notin C \end{cases} \tag{1}$$

holds where x_{abs} denotes the absolute x-coordinate. If we have to shift a vertex c_i on C to the right, we also have to shift all vertices $c_{i+1}, \ldots, c_q$ and some vertices not on C. We denote with $M(c_i)$ the set of vertices that must be shifted if c_i

is shifted. We represent all sets $M(c_i)$, $1 \le i \le q$, as a forest F of all vertices, where the roots are exactly the vertices on C. Thus, $M(c_i)$ contains the vertices in the tree with root c_i. We denote with $\mathit{father}(v)$ the unique predecessor of v in F, and guarantee that $\mathit{rank}(\mathit{father}(v)) > \mathit{rank}(v)$ holds. We can retrieve the absolute x-coordinates using (1) by first traversing the vertices on C from left to right, and then traversing all other vertices by decreasing rank.

Suppose we are in step k of the algorithm, that is, we have already placed $V_1, \ldots, V_{k-1}$, and we want to place V_k. We consider a vertex v that is already placed, i.e., $v \in G_{k-1}$. An outgoing edge (v, w) of v is called *free* if $w \in G \setminus G_k$, otherwise it is called *closed*. Let $e_1, \ldots, e_m$ be the outgoing edges of v sorted by increasing x-coordinate of their outpoints. For an edge e_μ, $1 \le \mu \le m$, let w_μ be the endpoint $\neq v$ of e_μ, and denote with r_μ the rank of w_μ. Note, that the definition of π implies that there are indices $0 \le a < b \le m+1$, such that the edges $e_1, \ldots, e_a$ are the closed edges e_μ with $\mathit{right}(V_{r_\mu}) = v$, and $e_b, \ldots, e_m$ are the closed edges e_μ with $\mathit{left}(V_{r_\mu}) = v$. We define $\mathit{next\_left}^k(v)$, such that $\mathit{next\_left}^k(v) < \mathit{outpoint}(e_\mu).dx$ for $a < \mu \le m$, and $\mathit{next\_right}^k(v)$, such that $\mathit{outpoint}(e_\mu).dx < \mathit{next\_right}^k(v)$ for $1 \le \mu < b$:

$$\mathit{next\_left}^k(v) = \begin{cases} \mathit{outpoint}(e_a).dx & \text{if } 1 \le a \le m \\ 0 & \text{if } m = 0 \\ -\mathit{out}_\ell(v) - 1 & \text{if } m > 0 \text{ and } a = 0 \end{cases}$$

$$\mathit{next\_right}^k(v) = \begin{cases} \mathit{outpoint}(e_b).dx & \text{if } 1 \le b \le m \\ 0 & \text{if } m = 0 \\ \mathit{out}_r(v) + 1 & \text{if } m > 0 \text{ and } b = m+1 \end{cases}$$

Algorithm Mixed-Model-Placement

Input: A plane graph $G = (V, E)$, an ordered partition $\pi = V_1, \ldots, V_N$ of G satisfying (P1)-(P4), and in- and outpoints as defined in Sect. 3.2.

Output: Absolute y- and relative x-coordinates according to equation (1) for each vertex in G.

Initialization: We place the set $V_1 = \{v_1, \ldots, v_s\}$. We set

$y(v_i) := 0$ for $1 \le i \le s$

$$x(v_i) := \begin{cases} \mathit{out}_\ell(v_i) & \text{if } i = 1 \\ \mathit{out}_r(v_{i-1}) + \mathit{out}_\ell(v_i) + 1 & \text{if } 2 \le i \le s \end{cases}$$

$C := v_1, \ldots, v_s$

for $k := 2$ **to** N **do**

Let $C = c_1, \ldots, c_q$, $V_k = \{z_1, \ldots, z_p\}$, $c_\ell = \mathit{left}(V_k)$ and $c_r = \mathit{right}(V_k)$.

For the y-coordinate, we simply set

$$y(z_i) := \mathit{in}_r(z_1) + \max_{\ell \le i \le r} y(c_i) + 1 \quad \text{for } 1 \le i \le p$$

For the x-coordinate, we temporarily compute the x-coordinates of $c_{\ell+1}, \ldots, c_r$ relative to c_ℓ. Therefore, we set

$$x(c_i) := \sum_{j=\ell+1}^{i} x(c_j) \quad \text{for } \ell + 1 \le i \le r.$$

We set dx_ℓ such that $outpoint(e).dx < dx_\ell$ for all free edges e of c_ℓ, and dx_r such that $dx_r < outpoint(e).dx$ for all free edges e of c_r:

$dx_\ell := next\_right^k(c_\ell)$ and $dx_r := next\_left^k(c_r)$

if $in(z_1) \geq 3$ **then**

We place a single vertex (see Fig. 3), i.e., $p = 1$. Let $c_{i_1}, \ldots, c_{i_{in(z_1)}}$ be the neighbors of z_1 on C. We set $t := i_{in_\ell(z_1)+2}$. Let $u_t = outpoint(c_t, z_1)$ and $dx_t = u_t.dx$. We want to place z_1 directly above u_t. Therefore,

$$x(z_1) := \max\{x(c_t) + dx_t, dx_\ell + out_\ell(z_1)\}$$

Let $\delta = x(z_1) - (x(c_t) + dx_t)$. If $\delta > 0$, then z_1 would lie to the right of u_t. We correct this by shifting $c_t, \ldots, c_q$ by δ units to the right. For shifting $c_r, \ldots, c_q$, it is sufficient to set

$$x(c_r) := \max\{x(c_r) + \delta - x(z_1), out_r(z_1) - dx_r\}$$

The vertices $c_{\ell+1}, \ldots, c_{r-1}$ disappear from C in this step, so we set their x-coordinates relative to z_1 and update the forest F:

$$x(c_i) := \begin{cases} x(c_i) - x(z_1) & \text{for } \ell < i < t \\ x(c_i) + \delta - x(z_1) & \text{for } t \leq i < r \end{cases}$$
$$father(c_i) := z_1 \quad \text{for } \ell < i < r$$

else

We place $p \geq 1$ vertices with at most two neighbors on C (see Fig. 4):

$$x(z_i) := \begin{cases} out_\ell(z_1) + dx_\ell & \text{for } i = 1 \\ out_r(z_{i-1}) + out_\ell(z_i) + 1 & \text{for } 2 \leq i \leq p \end{cases}$$
$$x(c_r) := \max\left\{out_r(z_p) - dx_r, x(c_r) - \sum_{j=1}^{p} x(z_j)\right\}$$

The vertices $c_{\ell+1}, \ldots, c_{r-1}$ disappear from C in this step, so we set their x-coordinates relative to z_1 and update the forest F:

$$x(c_i) := x(c_i) - x(z_1)$$
$$father(c_i) := z_1 \quad \text{for } \ell < i < r$$

fi

$C := c_1, \ldots, c_\ell, z_1, \ldots, z_p, c_r, \ldots, c_q$

od

4 The Analysis

In this section we show the correctness of our algorithm and analyze its running time and the bounds on the grid size, angular resolution, and number of bends. Let us analyze the running time of the algorithm. In the case in which the graph G is not biconnected, we can use the simple linear time augmentation algorithm suggested first by [28] in order to biconnect the graph. Note that this algorithm does not change the embedding of the plane graph G. The computation of the bounding boxes can obviously be done in linear time. From the outline of the

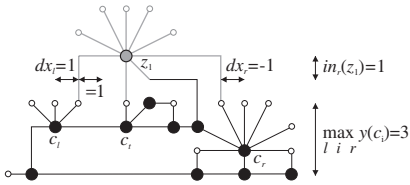

Fig. 3. Placing a single vertex

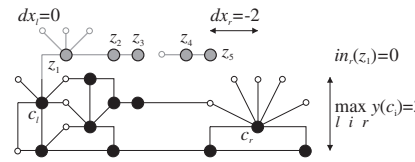

Fig. 4. Placing vertices $z_1, \ldots, z_p$

placement algorithm it is easy to see that this part also runs in linear time. The correctness requires a proof which is omitted in this extended abstract.

Lemma 1. *The algorithm* `Mixed-Model` *correctly computes a planar polyline grid drawing for any given plane graph G in linear time and space.*

Let us now analyze the bounds achieved by the algorithm `Mixed-Model`. For reasons of limited space, we cannot give the proofs of all bounds here.

Theorem 1. *The algorithm* `Mixed-Model` *constructs a planar polyline grid drawing of any plane graph with n vertices and maximum degree d on a $(2n-5) \times (\frac{3}{2}n - \frac{7}{2})$ grid with at most $5n-15$ bends and minimum angle $> \frac{2}{d}$, in which every edge has at most three bends and length* $\mathrm{O}(n)$.

Proof. The proofs of the angular resolution and the number of bends are similar to the ones given in [23]. We will give the proof for the bound on the height. We assume that the edges in the set S (see (P3)) have been inserted into our graph. Obviously, this assumption does not influence the height of the drawing. Note that we then have

$$in_r(w_i) = \max\left(0, \left\lceil \frac{in(v)-3}{2} \right\rceil\right) \leq \frac{1}{2} in(v) - 1.$$

For each node v with $rank(v) \geq 2$ there exists a vertex w with $rank(w) < rank(v)$ and $y(v) = in_r(v) + y(w) + 1$. Let w_t be a node with maximum y-coordinate. Then there exists a sequence of nodes $w_1, \ldots, w_t$ with $w_1 \in V_1$ and $y(w_{i+1}) - y(w_i) = in_r(w_{i+1}) + 1$.

Let m be the number of edges in G. The height of the layout is

$$\begin{aligned} y(w_t) &= \sum_{i=1}^{t-1} (in_r(w_{i+1}) + 1) = (t-1) + \sum_{i=2}^{t} in_r(w_i) \\ &\leq (t-1) + \sum_{i=2}^{t} \left(\frac{in(w_i)}{2} - 1 \right) \\ &= \frac{1}{2} \sum_{i=2}^{t} in(w_i) \leq \frac{1}{2}(m-1) \leq \frac{3}{2}n - \frac{7}{2}. \end{aligned}$$

□

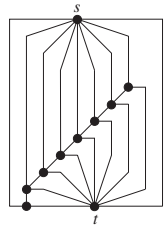

Fig. 5. A drawing of G_{10} with tight height

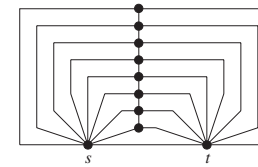

Fig. 6. A drawing of G_{10} with tight width

The bounds for the grid size are tight. We present a graph class for which the bounds for the width and height, respectively, are tight. Let $G_n = (V_n, E_n)$ be the graph defined as $V_n = \{s, t, w_i, \ldots, w_{n-2}\}$ and $E_n = \{(s, w_i), (t, w_i) \mid i = 1, 2, \ldots, n-2\} \cup \{(w_i, w_{i+1}) \mid i = 1, \ldots, n-3\} \cup \{(s, t)\}$. We consider an ordered partition with $V_1 = \{w_1, t\}$ and $V_N = \{s\}$. This fixes the embedding of the graph G_n. The algorithm `Mixed-Model` constructs the drawing shown in Figure 5 of height at least $\frac{3}{2}n - \frac{9}{2}$. The exact bound of $\frac{3}{2}n - \frac{7}{2}$ is achieved by a simple triangle. Now, consider the ordered partition with $V_1 = \{s, t\}$. According to the algorithm, the set V_1 is placed with width $2n - 5$ (see Fig. 6).

Note that Kant's algorithm also needs width $2n - 5$ for this example (and not $2n - 6$ as claimed in [23]). The bounds on the height cannot be achieved by the algorithm in [23].

5 The Algorithm in Practice

In this section we give some ideas how to modify and extent our algorithm in order to improve the grid size and the number of bends in practical instances. However, the worst-case bounds shown in Sect. 4 remain unchanged.

First, we consider vertices with degree one. We extend `Mixed-Model` in order to draw an edge incident to a vertex with degree one as a single short line segment. In a preprocessing step, we remove vertices with degree one from G. Then, we compute the ordered partition π of the resulting graph G'. Let v be a vertex in $G \setminus G'$ and (v, w) the only edge incident to v. When we assign in- and outpoints, we reserve either an in- or an outpoint for (v, w) in the bounding box around w, on which we place v at the end of the placement step. Therefore, this in- or outpoint must not have relative coordinates $(0, 0)$. The placement algorithm needs some modifications in order to cope with the additional in- and outpoints.

After placing a chain $V_k = \{z_1, \ldots, z_p\}$ there may be Δ unused columns to the right of z_p (see Fig. 7). We can reuse these columns in later steps if we need to shift any of the vertices $z_1, \ldots, z_p$ to the right. Hence, we can eventually reduce the width of the final drawing. The placing of a vertex v with $in(v) \geq 3$

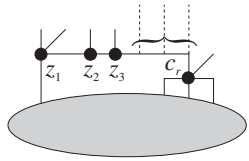

Fig. 7. Reuse of rows

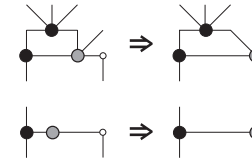

Fig. 8. Postprocessing

can be improved by computing the y-coordinate more carefully. Since not all inpoints of v have relative y-coordinate $in_r(v)$, we can eventually decrease the y-coordinate of v. Moreover, a postprocessing step can reduce the number of bends by local changes of the drawing. Figure 8 shows two examples, where we can move vertices on bend points.

Figure 9 shows several examples produced by `Mixed-Model` using some of the modifications described above. The drawings remain readable even if they contain vertices of high degree.

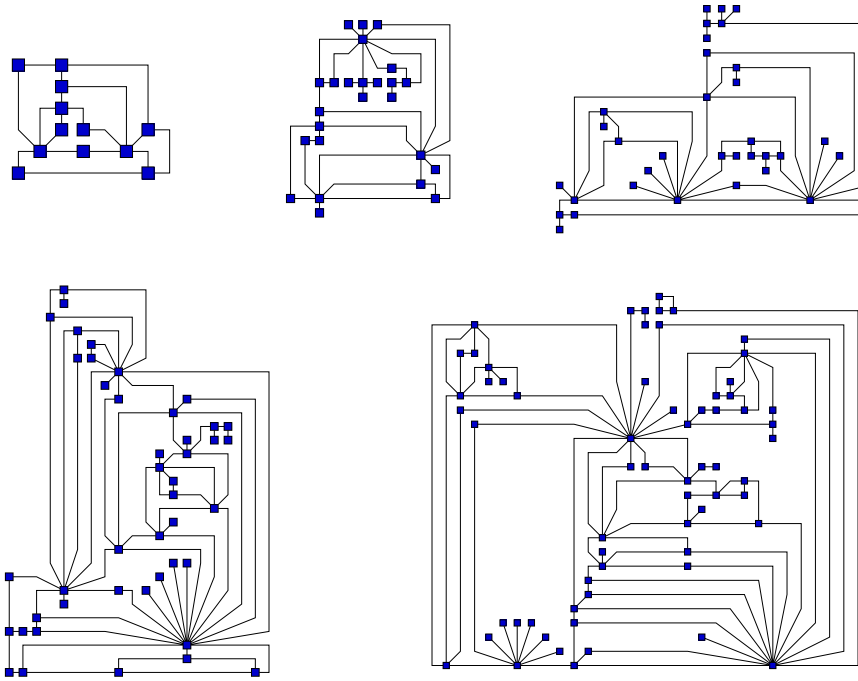

Fig. 9. Example drawings

6 Conclusions

We have presented an algorithm that constructs a planar polyline grid drawing of any plane graph with n vertices and maximum degree d on a $(2n-5) \times (\frac{3}{2}n - \frac{7}{2})$ grid with at most $5n - 15$ bends and minimum angle $> \frac{2}{d}$. To the best of our knowledge, this is the algorithm achieving the best simultaneous bounds concerning the grid size, angular resolution, and number of bends for planar grid drawings of high-degree planar graphs. An implementation of the algorithm is available with the AGD-Library (Algorithms for Graph Drawing) [2, 1]. Practical experience has shown that the drawings produced by our algorithm are aesthetically pleasing.

However, in general, the given graphs are not planar. In practice, the given nonplanar graph G is transformed into a planar graph G', in which each crossing of two edges is substituted by an artificial vertex and four new edges. A possible planarization method is to solve the maximum planar subgraph problem and then to re-insert the deleted edges so that the number of crossings is minimized [31, 12]. Practical experiments have shown that this method produces drawings with a relatively small number of edge crossings [11].

When using the algorithm `Mixed-Model` for such a planarized graph, the drawings may look strange and it is hard to identify the crossings. It is necessary to modify the algorithm so that the artificial crossing vertices are handled differently. It would be desirable to get mixed model drawings in which the edges are not allowed to bend in their crossing vertices.

References

[1] AGD-Library. *The AGD-Algorithms Library User Manual.* Max-Planck-Institut Saarbrücken, Universität Halle, Universität Köln, 1998. Available via "http://www.mpi-sb.mpg.de/AGD/". Partially supported by the DFG-cluster "Effiziente Algorithmen für diskrete Probleme und ihre Anwendungen".

[2] D. Alberts, C. Gutwenger, P. Mutzel, and S. Näher. The design of the AGD-Algorithms Library. In G.F. Italiano and S. Orlando, editors, *Proceedings of the Workshop on Algorithm Engineering (WAE '97)*, 1997. Venice, Italy, Sept. 11-13.

[3] T. Biedl. Optimal orthogonal drawings of connected plane graphs. In *Proc. Canadian Conference Computational Geometry (CCCG'96)*, volume 5 of *International Informatics Series*, pages 306–311. Carleton Press, 1996.

[4] T. Biedl. *Orthogonal Graph Visualization: The Three-Phase Method with Applications.* Ph.D. thesis, Rutgers University, Center for Operations Research, Rutgers, 1997.

[5] T. Biedl and G. Kant. A better heuristic for orthogonal graph drawings. *Computational Geometry: Theory and Applications*, 9:159–180, 1998.

[6] T. Biedl, B. Madden, and I. Tollis. The three-phase method: A unified approach to orthogonal graph drawing. In G. Di Battista, editor, *Graph Drawing (Proc. GD '97)*, volume 1353 of *Lecture Notes in Computer Science*, pages 391–402. Springer-Verlag, 1997.

[7] F. Brandenburg. Nice drawings of graphs and trees are computationally hard. In *Proc. of the 7th Interdisciplinary Workshop on Informatics and Psychology*,

volume 439 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 1988.

[8] M. Chrobak and G. Kant. Convex grid drawings of 3-connected planar graphs. *Interntl. Journal on Computational Geometry and Applications*, 7(3):211–224, 1997.

[9] H. De Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.

[10] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Comput. Geom. Theory Appl.*, 4:235–282, 1994.

[11] G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. *Comput. Geom. Theory Appl.*, 7:303–326, 1997.

[12] P. Eades and P. Mutzel. *Graph Drawing Algorithms*, CRC Handbook of Algorithms and Theory of Computation, Chapter 9, M. Atallah (Ed.). CRC Press, 1998. To appear.

[13] S. Fialko and P. Mutzel. A new approximation algorithm for the planar augmentation problem. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '98)*, pages 260–269, San Francisco, California, 1998. ACM Press.

[14] U. Fößmeier and M. Kaufmann. Algorithms and area bounds for nonplanar orthogonal drawings. In G. Di Battista, editor, *Graph Drawing (Proc. GD '97)*, volume 1353 of *Lecture Notes in Computer Science*, pages 134–145. Springer-Verlag, 1997.

[15] M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM J. Algebraic Discrete Methods*, 4(3):312–316, 1983.

[16] A. Garg. New results on drawing angle graphs. *Comput. Geom. Theory Appl.*, 9(1/2):43–82, 1998.

[17] A. Garg and R. Tamassia. On the computational complexity of upward and rectilinear planarity testing. Report CS-94-10, Comput. Sci. Dept., Brown Univ., Providence, RI, 1994.

[18] A. Garg and R. Tamassia. Planar drawings and angular resolution: Algorithms and bounds. In *Proc. 2nd Annual European Sympos. Algorithms (ESA '94)*, volume 855 of *Lecture Notes in Computer Science*, pages 12–23. Springer-Verlag, 1994.

[19] A. Garg and R. Tamassia. A new minimum cost flow algorithm with applications to graph drawing. In S. North, editor, *Graph Drawing (Proc. GD '96)*, volume 1190 of *Lecture Notes in Computer Science*, pages 201–216. Springer-Verlag, 1997.

[20] C. Gutwenger and P. Mutzel. Grid embeddings of biconnected planar graphs. Extended Abstract, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1997.

[21] J. Hopcroft and R. E. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.

[22] C. Hundack, P. Mutzel, I. Pouchkarev, and S. Thome. *ArchE*: A graph drawing system for archaeology. In G. Di Battista, editor, *Graph Drawing (Proc. GD '97)*, volume 1353 of *Lecture Notes in Computer Science*, pages 297–302. Springer-Verlag, 1997.

[23] G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica, Special Issue on Graph Drawing*, 16(1):4–32, 1996.

[24] G. W. Klau and P. Mutzel. Quasi–orthogonal drawing of planar graphs. Technical Report MPI-I-98-1-013, Max–Planck–Institut für Informatik, Saarbrücken, 1998.

[25] M. R. Kramer and J. van Leeuwen. The complexity of wire-routing and finding minimum area layouts for arbitrary VLSI circuits. In F. P. Preparata, editor, *Advances in Computing Research*, volume 2, pages 129–146. JAI Press, Greenwich, Conn., 1985.

[26] P. Mutzel and S. Fialko. New approximation algorithms for planar augmentation. Extended Abstract, to appear, 1998.

[27] H. Purchase. Which aesthetic has the greatest effect on human understanding? In G. Di Battista, editor, *Graph Drawing (Proc. GD '97)*, volume 1353 of *Lecture Notes in Computer Science*, pages 248–261. Springer-Verlag, 1997.

[28] R. Read. New methods for drawing a planar graph given the cyclic order of the edges at each vertex. *Congr. Numer.*, 56:31–44, 1987.

[29] J. A. Storer. On minimal node-cost planar embeddings. *Networks*, 14:181–212, 1984.

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

[31] R. Tamassia, G. Di Battista, and C. Batini. Automatic graph drawing and readability of diagrams. *IEEE Trans. Syst. Man Cybern.*, SMC-18(1):61–79, 1988.

[32] R. Tamassia and I. G. Tollis. Efficient embedding of planar graphs in linear time. In *Proc. IEEE Internat. Sympos. on Circuits and Systems*, pages 495–498, 1987.

[33] R. Tamassia and I. G. Tollis. Planar grid embedding in linear time. *IEEE Trans. on Circuits and Systems*, CAS-36(9):1230–1234, 1989.