Area-Efficient Static and Incremental Graph Drawings *

Therese C. Biedl[1] Michael Kaufmann[2]

[1] RUTCOR, Rutgers Univ., P.O. Box 5062, New Brunswick, NJ 08903, USA
e-mail: therese@rutcor.rutgers.edu.
[2] Wilhelm-Schickard-Institut, Universität Tübingen, Sand 13,
72076 Tübingen, Germany
e-mail: mk@informatik.uni-tuebingen.de.

Abstract. In this paper, we present algorithms to produce orthogonal drawings of arbitrary graphs. As opposed to most known algorithms, we do not restrict ourselves to graphs with maximum degree 4. The best previous result gave an $(m-1) \times (\frac{m}{2}+1)$-grid for graphs with n nodes and m edges.

We present algorithms for two scenarios. In the static scenario, the graph is given completely in advance. We produce a drawing on a grid of size at most $\frac{m+n}{2} \times \frac{m+n}{2}$, or on a larger grid where the aspect ratio of the nodes is bounded. Furthermore, we give upper and lower bounds for drawings of the complete graph K_n in the underlying model. In the incremental scenario, the graph is given one node at a time, and the placement of previous nodes can not be changed for later nodes. We then come close to the bounds achieved in the static case and get at most an $(\frac{m}{2}+n) \times (\frac{2}{3}m+n)$-grid. In both algorithms, every edge gets at most one bend, thus, the total number of bends is at most m.

Then we focus on planar graphs and outer-planar graphs. We obtain planar drawings in an $(m-n+1) \times \min\{\frac{m}{2}, m-n+1\}$-grid with $m-n$ bends for planar triconnected graphs. The best previous result here was an $m \times m$-grid and m bends, if the boxes of the nodes are constrained to be small.

All algorithms work in linear time.

1 Background

In recent years, the subject of graph drawings has created intense interest, due to numerous applications. Different drawing styles have been investigated (see [3] for an overview). One possible drawing technique is to produce orthogonal graph drawings, where only horizontal and vertical lines are employed. For example, in networking and data base applications, graph drawings serve as a tool to help display large diagrams efficiently. Specific uses of orthogonal graph drawings include Data Flow Diagrams and Entity Relationship Diagrams. The goal is to obtain an aesthetically pleasing drawing, and common objectives are small area, few bends, and few crossings.

* The research was partly funded by the NIST Advanced Technology Program Award No. 70NANB5H1162" and by the German Research Society, Grant DFG Ka/4-2.

For graphs with maximum degree 4, the usual definition of an orthogonal drawing is an embedding in the plane with nodes drawn as points, and edges drawn as sequences of horizontal and vertical line segments. For graphs with higher maximum degree, it is not possible to drawn the nodes as points, since no overlap among edges is allowed. Several attempts to generalize the known results for graphs with maximal degree 4 have been made. In Giotto [15], the high-degree nodes are split into several 'small' nodes and the previous techniques could be applied. Unfortunately, no theoretical bounds have been achieved and even worse, the final boxes of the nodes might be stretched unrelated to the degree.

In this paper, we forbid that nodes may be stretched far, which can be described in one sentence as "the nodes are not bigger than they need to be in order to accommodate all incident edges", see also [1]. For our presentation, we use the simpler constraint that the half-perimeter of the box of each node is at most $deg(v)$. In the same paper, a generic scheme was presented how to create orthogonal drawings of graphs by placing first nodes, then bends, and then ports. We will describe our algorithms using this scheme to simplify our presentation.

Most of our drawings will be in the so-called *Kandinsky-model* introduced by Fößmeier and Kaufmann [7]. In such a drawing, there are two different types of grid-lines. The grid-lines of a coarse grid are used to place the nodes. The grid-lines of a finer grid are added to allow more than one edge to attach on each side of a node. A set of neighboring fine grid-lines, called a *slot*, is assigned to each coarse grid-line. Any two slots are disjoint. Additionally, the Kandinsky-model imposes the *bend-or-end property.* If e is an edge attaching at the top of v, and if w is another node placed in the same column as v, and above v, then either e must have the other endpoint w, or e must be drawn with a bend *below* the lowest row of w. The same holds for the other directions analogously.

Fößmeier and Kaufmann presented an algorithm that computes an orthogonal drawing in the Kandinsky-model with the minimum number of bends [7]. However, this algorithm works only for planar graphs, the nodes all have the same sizes and the running time is $\mathcal{O}(n^2 \log n)$. The required area has not been analyzed. In a subsequent paper, Fößmeier, Kant and Kaufmann gave a linear-time heuristic to achieve, in the same model, an $m \times m$-grid and one bend per edge for planar triconnected graphs [6]. Another recent result by Papakostas and Tollis draws biconnected graphs with width $m-1$ and height $\frac{m}{2}+2$ using a less restrictive model [13].

We develop a new algorithm, which yields an $\frac{m+n}{2} \times \frac{m+n}{2}$-grid, and one bend per edge, and improves on the previous algorithms in various ways. It works for any simple graph, without demands on the connectivity. It takes only linear time. It improves the grid-size, apart from differences in the chosen model, by a factor of close to 2, under the reasonable assumption that m is significantly larger than n. Furthermore, the size of the box of each node v is bounded by $\frac{deg(v)}{2}$. However, the aspect ratio of each node can be unbounded, since the box of the node may appear to have size $1 \times \frac{deg(v)}{2}$. A variation of our algorithm achieves an aspect ratio of at most 1:2 for each node, while the grid-size is at most $(\frac{3}{4}m + \frac{n}{2}) \times (\frac{3}{4}m + \frac{n}{2})$.

To explore the Kandinsky-model more thoroughly, we study the special case of the complete graph. We give a construction in a grid of width and height $\frac{m}{2} + \frac{3}{8}n$ with $m - \frac{n}{2}$ bends. Furthermore, we show that any drawing of the complete graph in this model must have $m - n$ bends, thus we are close to optimality.

In incremental scenarios the graph is given one node at a time, and the next node has to be inserted into a fixed previous drawing. This incremental scenario is a first important step towards full interactive scheme and has been considered for 4-graphs in [12]. The critical point for the interactivity is that insertion of new nodes should not change the previous drawing, or at least we can modify it only in a very restricted way. In [15] as well as in [1] interactive schemes have been presented. The second paper yields an $(m+n) \times (m+n)$-grid and m bends. We modify our static algorithm and get area bounds which are only about a factor of 4/3 away from our results for the static scenario.

In the third part of the paper, we consider static algorithms for drawing planar graphs. For triconnected planar graphs we produce drawings in a grid of size of at most $(m - n + 1) \times \min\{\frac{m}{2}, m - n + 1\}$, where the number of bends is $m - n$. The height and width of each box is at most $deg(v)$, which distinguishes these drawings from k-visibility representations (e.g. in [16]), where we have less bends and a smaller area, but in exchange the nodes are bigger.

2 The static scenario

Assume that the full graph G is given in advance. We present one generic algorithm, which works for any node order and edge orientation. Using special implementations, we obtain two different results on high-degree drawings.

2.1 A generic algorithm

Assume some arbitrary node order $\{v_1, \ldots, v_n\}$ and some arbitrary edge orientation of G is given. An edge directed from v_i to v_j is called *good* if $i < j$ and *bad* otherwise. A *predecessor (successor)* of v_j is a neighbor v_i where the edge (v_i, v_j) is incoming (outgoing) at v_j. A predecessor is *good* if the according edge is good. We denote the number of incoming edges of v_j as $indeg(v_j)$, and the number of good and bad incoming edges of v_j as $indeg^{good}(v_j)$, and $indeg^{bad}(v_j)$, respectively. Similarly we define $outdeg(v)$, $outdeg^{good}(v)$ and $outdeg^{bad}(v)$.

We create the drawing by first computing the coarse grid-lines for the nodes, this corresponds to the "node placement" phase introduced in [1]. We assign one row for each node, and one column for each node. No two nodes will be placed in the same row or the same column. These rows and columns will later be expanded into horizontal and vertical slots.

We compute rows for the nodes by processing them in forward order. Consider v_i, $i = 1$ to n. If it has no good predecessor, then we create a new row at an arbitrary place. Otherwise, we add a row close to the median of the rows of the good predecessors. Precisely, let $r_1, \ldots, r_s$ be the rows of the good predecessors of v_i. If s is odd, add a row before or after $r_{\frac{s+1}{2}}$. If s is even, add a row somewhere between $r_{\frac{s}{2}}$ and $r_{\frac{s}{2}+1}$. Place v_i in this row.

We compute a column for each node similarly, but this time in backward order. For v_i, $i = n$ down to 1, place v_i in a new column. This new column is created near the median of the columns of the good successors of v_i, if v_i has good successors, and at an arbitrary place otherwise.

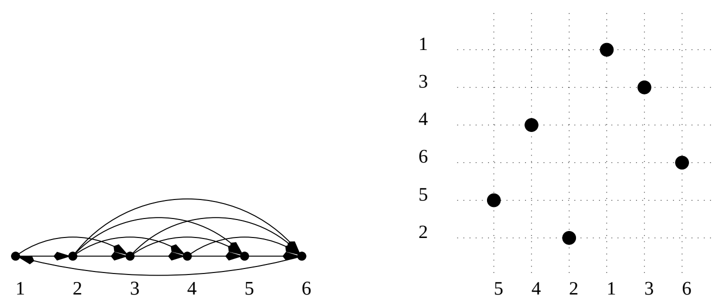

Fig. 1. An example of the node placement.

Next, we assign an approximate place for each bend, which corresponds to the "edge routing" phase in [1]. If $e = (v_i, v_j)$ is an edge directed from v_i to v_j, then we place a temporary bend in the row of v_i and the column of v_j. This edge routing does not yield a feasible drawing, but it gives a first sketch, enough to analyze the drawing. For each node v, let $b(v)$ be the number of edges that attach at the bottom of v. Similarly, define $l(v)$, $r(v)$ and $t(v)$ as the number of incident edges at the left, right, and top side of v.

Lemma 1. *For each node,* $\lfloor \text{outdeg}(v)/2 \rfloor \leq r(v), l(v) \leq \lceil \text{outdeg}^{good}(v)/2 \rceil + \text{outdeg}^{bad}(v)$, *and* $\lfloor \text{indeg}(v)/2 \rfloor \leq t(v), b(v) \leq \lceil \text{indeg}^{good}(v)/2 \rceil + \text{indeg}^{bad}(v)$.

Proof. Consider $b(v)$. By the bend-placement, any bend at the bottom of v belongs to an incoming edge of v. By the node placement, at most half (rounded up) and at least half (rounded down) of the good predecessors are below v. The bad predecessors can be, but need not be, below v. The result follows for $b(v)$, and is similar for the other three sides.

As described in [1], we can get a feasible drawing from this sketch easily. Consider a row r. In this row, there is one node v, and some number of bends. We add $max\{r(v), l(v), 1\} - 1$ rows above the row of v. Then, we distribute the bends among these rows such that no two edges on one side cross, as demonstrated in Fig. 2.

This algorithm can be implementing in $\mathcal{O}(m+n)$ time, using the data structure by Dietz and Sleator [4], the linear-time median-finding algorithm and bucket sort (see for example [2]).

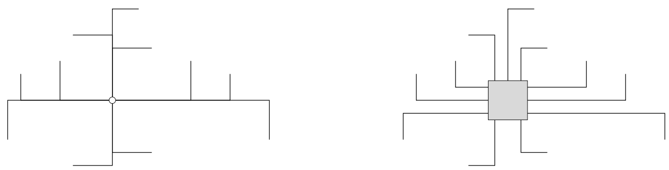

Fig. 2. We assign edges to newly added rows and columns in such a way that there are no crossings between edges from the same side.

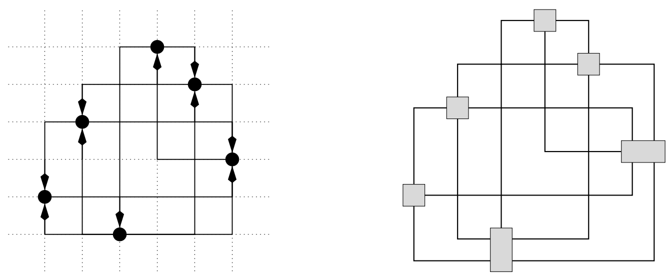

Fig. 3. Continuing the example of Fig. 1, we show the edge routing and how the edges are distributed in the final drawing.

2.2 A small grid-size

In this section, we show how to achieve a small grid-size by choosing a special node order and orientation.

Definition 2. A node order together with an edge orientation is called *polar-free almost-acyclic*, if $indeg(v) \geq 1$ and $outdeg(v) \geq 1$ for all $v \in V$. Furthermore,
(a) $indeg^{bad}(v) \leq 1$, if $indeg^{good}(v) > 0$ then $indeg^{bad}(v) = 0$, and
(b) $outdeg^{bad}(v) \leq 1$, if $outdeg^{good}(v) > 0$ then $outdeg^{bad}(v) = 0$.

Lemma 3. *Let G be a simple graph without nodes with degree ≤ 1. Then G has a polar-free almost-acyclic order and orientation. It can be found in $\mathcal{O}(m)$ time.*

Proof (Sketch). If G is biconnected, compute an *st*-order [11] of it, such that the nodes v_1 and v_n are adjacent. Direct the edges according to it, and reverse edge (v_1, v_n). This can be done in $\mathcal{O}(m)$ time [5].

If G is not biconnected, compute an *st*-order for every biconnected component B of G. If B contains at least two cut-nodes, choose two cut-nodes as first and last node. Otherwise, choose two adjacent nodes in B that are not cut-nodes, and reverse the edge between them. Merge all these orderings such that the order in each component stays unchanged.

Applying the generic algorithm with a polar-free almost-acyclic order and orientation leads to good worst-case bounds on the grid-size.

Theorem 4. *Let G be a simple graph without nodes of degree ≤ 1. Then G has an orthogonal drawing in an $\frac{m+n}{2} \times \frac{m+n}{2}$-grid with one bend per edge. The box size of each node v is at most $\frac{\deg(v)}{2} \times \frac{\deg(v)}{2}$. It can be found in $\mathcal{O}(m)$ time.*

Proof. We will only prove the claim on the height, the claim on the width is similar. After the node placement, we had n rows. For each node v, we add $max\{r(v), l(v), 1\} - 1$ rows. Thus, the height is $\sum_{v \in V} max\{1, r(v), l(v)\}$. By Lemma 1 and the conditions of the polar-free almost-acyclic ordering, we have $r(v), l(v) \leq \lceil \frac{outdeg(v)}{2} \rceil$. Furthermore, since $outdeg(v) \geq 1$ for all nodes, we also have $1 \leq \lceil \frac{outdeg(v)}{2} \rceil$. Therefore, the height is at most $\sum_{v \in V} \lceil \frac{outdeg(v)}{2} \rceil \leq \sum_{v \in V} \frac{outdeg(v)+1}{2} = \frac{m+n}{2}$. The height of the box of node v is $max\{1, r(v), l(v)\} \leq \lceil \frac{outdeg(v)}{2} \rceil \leq \lceil \frac{deg(v)-1}{2} \rceil \leq \frac{deg(v)}{2}$, since $indeg(v) \geq 1$.

A remark here on the condition of "no nodes of degree ≤ 1". Such nodes should be pre-processed and removed from the graph. They can later be re-inserted, by adding only one grid-line and no bend per node. We skip the details here, and only mention that we can achieve a width and height of $\lceil \frac{m+n}{2} \rceil$ for the grid and $\lceil \frac{deg(v)+1}{2} \rceil$ for each node.

2.3 Nodes with bounded aspect ratio

In the previous algorithm, the aspect ratio of a node may be unbounded, since the box of node v may appear as a $1 \times deg(v)/2$ box. In this section, we add the requirement to the model that the nodes should have a bounded aspect ratio. This can be ensured by an orientation via eulerian circuits. Therefore, we make the graph first eulerian by adding new edges between pairs of nodes with odd degree. Then we compute the eulerian circuits which determine the orientation of the edges. For the resulting orientation, we have $indeg(v), outdeg(v) \leq \lceil deg(v)/2 \rceil$.

Now we want a node order $\{v_1, \ldots, v_n\}$ that minimizes the number of bad edges. This problem is $\mathcal{NP}$-complete, since it is the feedback arc problem [8]. But we can always find a node order such that there are at most $m/2$ bad edges.

Applying the generic algorithm with this node order and edge orientation, we obtain good bounds on the aspect ratio of each node. Precisely, one can see from Lemma 1 that the height of v is at most $indeg(v) \leq outdeg(v) + 1$, and the width is at least $(outdeg(v) + 1)/2$, therefore the aspect ratio of v is at most 1:2.

The area of the resulting drawing is determined by the bad edges. If m_g and m_b is the number of good and bad edges, respectively, then the width of the grid is at most $\frac{m_g}{2} + \frac{n}{2} + m_b \leq \frac{3}{4}m + \frac{n}{2}$.

Theorem 5. *Let G be a simple graph without nodes of degree ≤ 1. Then G has an orthogonal drawing in an $(\frac{3}{4}m + \frac{n}{2}) \times (\frac{3}{4}m + \frac{n}{2})$-grid with one bend per edge where each node has aspect ratio at most 1:2. It can be found in $\mathcal{O}(m)$ time.*

We expect that with a suitable choice of a heuristic to determine the node order, the expected area of the drawing can be improved tremendously.

3 The complete graph in the Kandinsky-model

In this section, we study the behavior of the Kandinsky-model for graphs with many edges. We derive upper and lower bounds for the drawing of K_n, the complete graph with n nodes.

Theorem 6. *If n is divisible by 8, then K_n can be embedded in the Kandinsky-model in a grid of width and height $\frac{m}{2} + \frac{3}{8}n$ with $m - \frac{n}{2}$ bends.*

Proof. We divide the nodes into four groups of equal size. Enumerate the nodes in each group as $\{v_1^{(i)}, \ldots, v_{n/4}^{(i)}\}$. We place the nodes "as a diamond," i.e.,on a coarse $\frac{n}{2} \times \frac{n}{2}$-grid such that node $v_k^{(i)}$ is placed in the ith quadrant and such that the absolute value of the coordinates is $(k, \frac{n}{4} + 1 - k)$.

For each $1 \leq k \leq \frac{n}{4}$, we define the *$k$-square* to be the four nodes $v_k^{(i)}$, $i = 1, \ldots 4$. These nodes induce a K_4 with 6 edges. Four of these edges are drawn straight, the other two edges are drawn with two bends each. We use two columns for these bends if $k \leq \frac{n}{8}$, and two rows otherwise.

For any $1 \leq i \leq \frac{n}{4}$, any $j > i$, and $k = 1, \ldots, 4$, we define the *kth i,j-cross* as the group of nodes $\{v_i^{(k)}, v_j^{(k)}, v_i^{(k-1)}, v_j^{(k+1)}\}$, where all additions are modulo 4. In an i,j-cross there are four edges that were not in a k-square. We draw these four edges with one bend each, using two rows and two columns. This is possible since by $j > i$ the ith vertical and the jth horizontal coarse grid-line cross inside the diamond. See also Fig. 4.

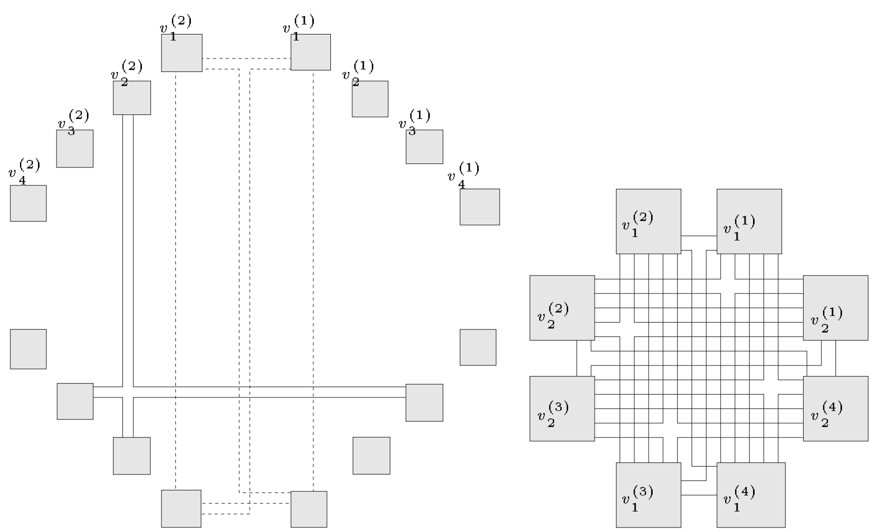

Fig. 4. The construction in the Kandinsky-model. We show the 1-square (dashed), and the third 2,3-cross (solid), and the completed drawing of K_8.

We have $4\sum_{i=1}^{n/4} i(\frac{n}{4} - i - 1) = \frac{n^2}{8} - \frac{n}{2}$ many crosses, each accounts for 4 edges and 4 bends and uses two rows and columns. We have $\frac{n}{4}$ squares, each accounts for 6 edges and 4 bends. Half of the squares use 4 columns and 5 rows, the other half uses 5 columns and 4 rows. Since $4(\frac{n^2}{8} - \frac{n}{2}) + 6\frac{n}{4} = \frac{n^2}{2} - \frac{n}{2} = m$, all edges are either in a square or in a cross. Thus, the total number of bends is $4(\frac{n^2}{8} - \frac{n}{2}) + 4\frac{n}{4} = m - \frac{n}{2}$. The width and height each is $2(\frac{n^2}{8} - \frac{n}{2}) + 4\frac{n}{8} + 5\frac{n}{8} = \frac{n^2}{4} + \frac{n}{8} = \frac{m}{2} + \frac{3}{8}n$.

Variants of this technique lead to other drawings in other models. If we drop the bend-or-end property, but still keep the dimensions of the nodes limited, then we can improve the bounds to a grid of width and height $\frac{m}{2} + \frac{n}{4} - 1$ with $m - 2n + 2$ bends. If we also drop the constraint on the size of the nodes but let them grow arbitrarily (similar as in 2-visibility representations), we can even prove that the grid has width and height $\frac{m}{2} - \frac{3}{4}n + 3$ with $m - 6n + 20$ bends. This is optimal in the number of bends, since any orthogonal drawing has at most $6n - 20$ edges drawn as straight lines [9].

Now we prove a lower bound on the number of bends.

Theorem 7. *Any drawing of the K_n in the Kandinsky-model has at least $m - n$ bends.*

Proof. Assume we have a drawing Γ of K_n. Let A be the number of vertical slots that contain nodes. Let B be the number of horizontal slots that contain nodes. The number of nodes in the ith vertical slot is denoted a_i, while the number of nodes in the jth vertical slot is denoted b_i, so $\sum_{i=1}^{A} a_i = \sum_{i=1}^{B} b_i = n$.

The edges split into three groups. E_a are the edges where the two endpoints are in the same vertical slot, E_b are the edges where the endpoints are in the same horizontal slot. We have $E_a \cap E_b = \emptyset$, by property of the Kandinsky-model. E_c are the remaining edges.

We have $|E_a| = \sum_{i=1}^{A} \binom{a_i}{2}$. Of these edges, $\sum_{i=1}^{A}(a_i - 1)$ can be drawn without bends. On the other hand, all remaining edges in E_a must have at least two bends by the bend-or-end property. Therefore, the number of bends in E_a is at least $2\sum_{i=1}^{A}[\binom{a_i}{2} - (a_i - 1)]$. Similarly, the number of bends in E_b is at least $2\sum_{i=1}^{B}[\binom{b_i}{2} - (b_i - 1)]$.

Every edge in E_c has at least one bend. So the total number of bends is

$$m - \sum_{i=1}^{A}\binom{a_i}{2} - \sum_{i=1}^{B}\binom{b_i}{2} + 2\sum_{i=1}^{A}\left[\binom{a_i}{2} - (a_i - 1)\right] + 2\sum_{i=1}^{B}\left[\binom{b_i}{2} - (b_i - 1)\right]$$
$$= m + \sum_{i=1}^{A} a_i^2/2 + \sum_{i=1}^{B} b_i^2/2 - 5n + 2A + 2B.$$

Given fixed values of A, B, the minimum of this expression is achieved if all a_i's, respectively all b_i's, are the same; so $a_i = \frac{n}{A}$ and $b_i = \frac{n}{B}$. The number of bends then is $m + n^2/2A + n^2/2B - 5n + 2A + 2B$. Minimizing this for A and B, we arrive at $A, B = \frac{n}{2}$, therefore the number of bends is at least $m + 2n - 5n + 2n = m - n$.

4 Incremental drawing of graphs with high degrees

We now study the *incremental scenario* where the nodes are given one by one, but we have to fix one placement of a node before the next is given. An insertion of a new row or column is allowed only at those positions where such an operation does not stretch any node box unnecessarily large. We naturally generalize the relative-coordinates scenario introduced in [12].

Assume the order of the nodes given is $\{v_1, \ldots, v_n\}$. Since the static algorithm also worked with a node order, it is an obvious idea to try to use the same algorithm with the user-defined node order and the induced edge orientation. Two main differences occur: We have no information about the node order, and in particular it is possible that the in-degree or out-degree of a node is 0. Secondly, we have no information about the successors of node v_i when placing v_i, and therefore have to find a different column-choice strategy.

We need to maintain a valid drawing, i.e.,a drawing where nodes are boxes and no two edges overlap. Thus, we will represent the nodes as boxes throughout the algorithm and keep the invariant that for any two boxes, the x-intervals of the boxes as well as the y-intervals are disjoint.

So assume nodes $v_1, \ldots, v_{i-1}$ are drawn in this way. Compute all predecessors of node v_i, and find their median as in the static scenario. Add a new row at this median place, such that it does not intersect any existing node (this is possible since the y-intervals are disjoint).

Add $w_i = max\{1, \lceil \frac{indeg(v_i)}{2} \rceil\}$ new columns for v_i, and add these either at the extreme left or at the extreme right. Clearly, in a practical implementation, one would allow to place a new node at any new column in the middle of the drawing, but here we analyze only the situation where placements to the right or left hand side of the drawing is allowed. Thus, these columns do not intersect any existing node. v_i will be drawn as $w_i \times 1$-box in the beginning, and will increase in height later, when we add more outgoing edges.

To route an edge (u, v_i), we may have space left at the correct side of u, or we may have to increase the height of u to make space for the edge. We can increase u by adding a new row, this will not intersect any other node. We make sure that this new row is between the upward-bending and the downward-bending edges on either side. The edge (u, v_i) will leave u on the right or left (depending on where we placed v_i), and enter v_i at the top or bottom side of the box of v_i. It bends exactly once.

Let n_s be the number of nodes with in-degree 0. Since for each node v_i the width of the box is $max\{1, \lceil \frac{indeg(v_i)}{2} \rceil\}$, the total width now is $n_s + (m + n - n_s)/2 = (m + n + n_s)/2 \leq \frac{m}{2} + n$.

Since we choose a column without knowledge about the successors, any number of outgoing edges may attach on the right side or on the left side of the box. Thus, we can estimate the height of node v_i only by $h_i = max\{1, outdeg(v_i)\}$. So if n_t is the number of nodes with out-degree 0, then the total height may be as much as $n_t + m$.

We reduce this bound by choosing the column for v_i wisely. At a fixed time, if a node v has $r(v)$ incident edges on the right side and $l(v)$ edges on its left side,

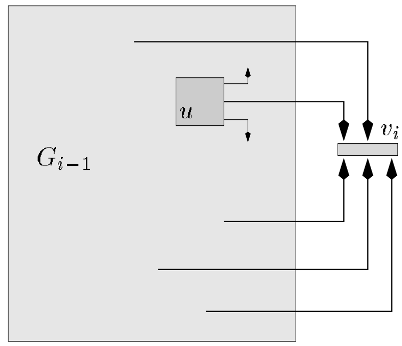

Fig. 5. v_i is inserted at the extreme left or right end, and at the median of the rows of its predecessor.

and if $r(v) > l(v)$ then we say $l(v)$ edges on each side are *mated*, and $r(v) - l(v)$ edges are *(right) free*. Obviously, the height of v is $r(v)$. We call v *right-weighted* since $r(v) > l(v)$.

If f is the number of left or right free edges at the end, then $m - f$ is the number of mated edges. The total height of the drawing is n_t plus half of the mated edges plus the number of free edges, which amounts to $n_t + \frac{m-f}{2} + f = n_t + \frac{m}{2} + \frac{f}{2}$. So we want to minimize the number of free edges.

Our approach is greedy and chooses that side for the placement of node v_i which appears better with respect to the number of free edges. If the number of right-weighted predecessors of v_i is smaller than the number of left-weighted predecessors, then we place v_i on the right hand side. Therefore, the left-weighted predecessors loose one free edge each. Otherwise, we place v_i on the left side, and the right-weighted predecessors loose one free edge each.

We estimate the number f of free edges. There are only $n - n_t$ nodes with outgoing edges, and only those may have free edges. So the number of first free edges is at most $n - n_t$. Consider a right free edge $e = (v, w)$ which is not the first on its node v. Edge e was inserted when we placed w on the right side. Hence the number a of left-weighted predecessors of w was at least as big as the number b of right-weighted predecessors. We created $2a$ mated edges (the incoming edges of w, and the edges that caused the predecessors to be left-weighted), and only b right-free edges. So we can assign two mated edges to every free edge that was not the first on its node. This gives $f \leq n - n_t + \frac{1}{2}(m - f)$, or $f \leq \frac{2}{3}(n - n_t) + \frac{1}{3}m$. Therefore, the total height is at most $\frac{1}{3}n + \frac{2}{3}n_t + \frac{2}{3}m \leq \frac{2}{3}m + n$.

Theorem 8. *Assume G is given incrementally. Then we can achieve an $(\frac{m}{2} + n) \times (\frac{2}{3}m + n)$-grid and 1 bend per edge.*

Next, we show that the analysis above is tight (at least for this algorithm). We define a graph with $n + 1$ nodes, where n is divisible by 3, as follows:

Insert nodes v_1, v_2, v_3 without any edges.
For $i = 1$ to $n/3 - 1$ **do**
  let $j = 3 \cdot i$;
  insert v_{j+1} adjacent to $v_{j/3+1}, ..., v_j$; (* inserted to the left hand side *)
  insert v_{j+2} adjacent to $v_1, ..., v_{2j/3}$; (* inserted to the right *)
  insert v_{j+3} adjacent to $v_1, ..., v_{j/3}, v_{2j/3+1}, ..., v_{3j}$; (* inserted to the right *)
od;
insert v_{n+1} adjacent to $v_{n/3}, ..., v_{n-1}$; (* inserted to the right *)

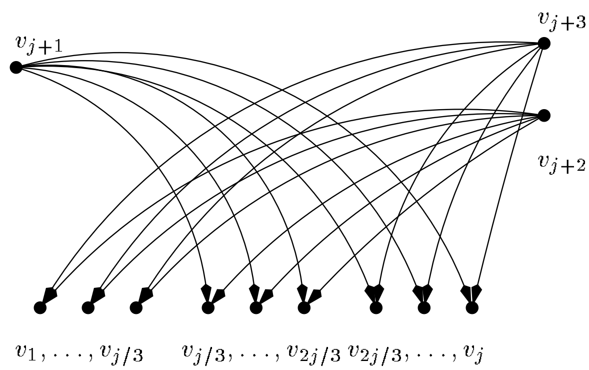

Fig. 6. A graph where incremental drawing performs badly.

We can prove for this graph with $n + 1$ nodes, and a total number of edges of $m = n^2/3 - n/3$, a bound for the height of the drawing of $1 + \frac{2}{3}m + \frac{2}{9}n$. This almost closes the gap between the guaranteed behavior of the greedy algorithm and its behavior on a specific example.

5 Planar graphs

We now present a new linear-time heuristic that works for triconnected planar graphs, and that gives an $(m-n+1) \times \min\{\frac{m}{2}, m-n+1\}$-grid and $m-n$ bends. Every edge has at most one bend. Thus, we improve the grid-size by a factor of 2, and we decrease the number of bends by n, compared to the best previous bounds of an $m \times m$-grid and m bends [6].

5.1 The canonical ordering

Assume from now on that G is a triconnected, simple, planar graph. For such graphs, we can use the *canonical ordering* as introduced by Kant [10]. For a node ordering $\{v_1, \ldots, v_n\}$, let $G(i)$ be the graph induced by $v_1, \ldots, v_i$, in the planar embedding as induced by G.

Lemma 9. *[10] Let G be a planar simple triconnected graph with a fixed planar embedding. Then G has a node ordering $V = \{v_1, \ldots, v_n\}$, called a* canonical ordering, *such that the following holds:*

- *(v_1, v_2) is an edge and belongs to the outer-face, with v_1 clockwise after v_2 on the outer-face.*
- *v_n belongs to the outer-face and has at least three neighbors.*
- *For $3 \leq j \leq n-1$, v_j is in the outer-face of $G(j-1)$, and one of the following holds:*
  - *Either "v_j is a new single node", i.e.,v_j has at least three neighbors in $G(j-1)$ and at least one neighbor in $G - G(j)$. $G(j)$ is biconnected.*
  - *Or "v_j is part of a new chain", i.e.,there exists i, k, $i \leq j \leq k$, such that for all $i < l < k$ v_l is adjacent to v_{l-1} and v_{l+1}, has no other neighbor in $G(k)$, and at least one neighbor in $G - G(k)$. Furthermore, v_i and v_k have each exactly one neighbor in $G(i-1)$ and at least one neighbor in $G - G(k)$. $G(k)$ is biconnected.*

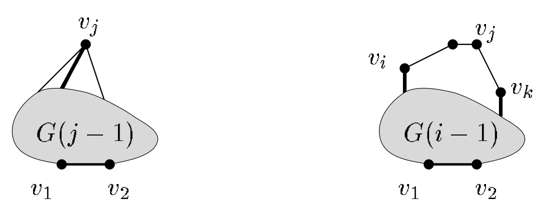

Fig. 7. The two different possibilities for v_j.

Given an element other than the first one of the ordering, let the *left foot-point* be the clockwise first node after v_1 on the outer-face connected to a node in the new element. We define the *right foot-point* similarly.

We call a node on the outer-face of G_i *open* if it still has neighbors in $G - G_i$, otherwise we call it *closed*. We distinguish the chain-elements further. A chain is *left-free* if its left foot-point is closed after adding the chain. Or, in other words, the highest outgoing edge of the left foot-point, which is the edge leading to the node with the highest number in the ordering, is the one that is incoming to the chain. Similarly a chain is *right-free* if its right foot-point is closed after adding the chain. A chain is *free* if it is either right-free or left-free, and *non-free* otherwise.

5.2 The placement

We add the nodes following the elements of the canonical ordering. Throughout the algorithm, we maintain the invariant that the open nodes have disjoint intervals, and they are sorted from left to right when going around the outer-face from v_1 to v_2 in clockwise direction.

We start with the **placement of** v_1 **and** v_2. We add one row and two columns for these two nodes, and draw the edge between them as a straight line.

Assume we want to **add a new non-free chain** $v_k, \ldots, v_l$ with left and right foot-point c_α respectively c_β. Let r_α be the highest row used by any incident edge of c_α on the right side of c_α. Similarly, let r_β be the highest row used by any incident edge of c_β on the left side. We will embed the chain in the row above $max\{r_\alpha, r_\beta\}$ (we add a new row on top if there was no such row yet). Call this row r_c. Add $l-k+1$ new columns between the columns of c_α and c_β. Place $v_k, \ldots, v_l$ in these columns and in r_c. For lack of space, we skip the proof that this placement does not create any overlap.

If $r_c = r_\alpha + 1$, then, if necessary, we increase the height of the node c_α by 1 unit, so that it now overlaps r_c as well. In this case, the edge (c_α, v_k) is routed as horizontal line. Otherwise, we increase the width of c_α by one, and add a new column between the left-continuing and right-continuing outgoing edges of c_α. We route the edge (c_α, v_k) using this column with a bend above c_α. Similarly we proceed for the edge (c_β, v_l). All edges (v_i, v_{i+1}), $i = k, \ldots, l-1$ are routed horizontally along r_c.

Assume we next want to **place a left-free chain** $v_k, \ldots, v_l$ (the placement of a right-free chain is symmetric). Let the foot-points be again c_α and c_β. Add a new row r_c on top of the drawing. Add $k-l$ columns between the columns of c_α and c_β. All nodes of the chain will be placed in r_c. Place v_k in the column of c_α (this does not violate the invariant, since c_α is closed after adding the chain). Place $v_{k+1}, \ldots, v_l$ in the newly created columns. The edge (c_α, v_k) is routed vertically. All edges (v_i, v_{i+1}), $i = k, \ldots, l-1$ are routed horizontally. The edge (v_l, c_β) is routed with a bend above c_β, this adds a new column to c_β.

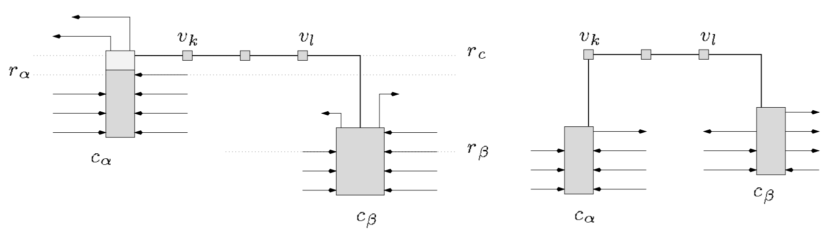

Fig. 8. Placement of a non-free chain and left-free chain, respectively.

Finally, assume we next want to **place a new node** v_i. Let $w_1, \ldots, w_d$ be the predecessors of v_i, sorted from left to right. Nodes with in-degree 2 are chain-elements, so $d \geq 3$. Add $\lceil \frac{d-1}{2} \rceil$ rows on top of the existing drawings. v_i will overlap all these rows. Add a new column to each predecessor of v_i. Place v_i in the new column of $w^* = w_{\lceil \frac{d}{2} \rceil}$, which is closed after adding v_i since $d \geq 3$.

Route the edge (w^*, v_i) vertically, while all other edges (w_j, v_i) are routed with a bend above w_j. This adds a new column to c_α and c_β. Assign rows to the incoming edges of v_i such that there is no crossing among them.

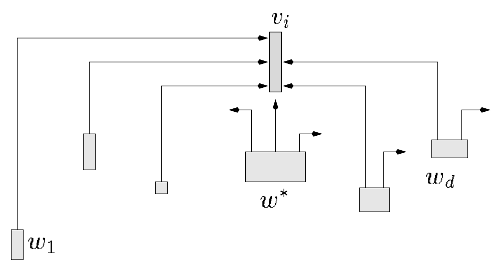

Fig. 9. Placement of new node.

5.3 Bounds

Let us now consider the height of the obtained drawing. Placing v_1 and v_2 requires one row. Placing a chain $v_k, \ldots, v_l$ (free or not) requires $1 = \frac{indeg(v_l)}{2} \leq \sum_{i=k}^{l} \frac{indeg(v_i)}{2}$ rows. Placing a node requires $\lceil \frac{indeg(v_i)-1}{2} \rceil$ rows. Since $indeg(v_2) = 1$, the total height is therefore at most $\frac{1}{2} + \sum_{v \in V} \frac{indeg(v)}{2} = \frac{m+1}{2}$. With a clever choice of v_n, we can shave off this $\frac{1}{2}$-term, and get a bound of $\frac{m}{2}$. Another estimation on the height can be obtained as follows: For v_1 and v_2 we use $1 = 2 + \sum_{i=1}^{2}(indeg(v_i) - 1)$ rows. For every chain we use $1 = \sum_{i=k}^{l}(indeg(v_i) - 1)$ rows. For placing a node we use $\lceil \frac{indeg(v)-1}{2} \rceil \leq indeg(v) - 2$ rows, since the in-degree is at most 3. Since we have at least one node-element in v_n, the total number of rows therefore is at most $1 + \sum_{v \in V}(indeg(v) - 1) = m - n + 1$.

Now let us consider the width. We will count added columns when we route incoming edges. Thus, to place v_1 and v_2 we do not count any of the used columns – they will be accounted for when placing the highest outgoing edges of v_1 and v_2. Similarly, we do not count any columns when adding a non-free chain, and only one column when adding a free chain.

Finally, when adding a node v with in-degree d, we count the $d-1$ columns of the predecessors that are not the median predecessor w^*. The column of w^* will be accounted for when placing the last outgoing edge of v. The only exception to this is the case $v = v_n$, in which case we must count d columns. Since the second element of the ordering is always a non-free chain, the total number of columns is at most $\sum_{v \in V} indeg(v) - 1 + 1 = m - n + 1$. Similarly one can estimate the number of bends as $m - n$. We increase the height or width of a node only if we add a new incident edge to it. Every node has an incident horizontally attaching edge (one of the incoming edges), and an incident vertically attaching edge (the last outgoing edge). Therefore, the half-perimeter of each node is at most $deg(v)$.

It is quite straightforward to show that the algorithm can be implemented in linear time.

Theorem 10. *There exists a linear-time heuristic to draw a planar triconnected graph orthogonally in an* $(m-n+1) \times \min\{m-n+1, \frac{m}{2}\}$*-grid with* $m-n$ *bends. Every edge has at most one bend. The half-perimeter of the box of each node is at most* $\deg(v)$*.*

5.4 Variations of the algorithm

We now show how to achieve related results with slight variations of the algorithms. For lack of space, we have to skip all proofs.

A graph is called *outer-planar* if we can add a dummy-node v^* connected to all nodes and the graph stays planar. If we choose this dummy-nodes as last node, then we can show that we get an $(n-1) \times n$-drawing where all edges are routed horizontally. So the produced drawing is a 1-visibility representation. As opposed to all previous algorithms (e.g. [16,14,9,6]), in our drawings there are known bounds on the height of a node.

Theorem 11. *Let G be an outer-planar graph. Then G has a 1-visibility representation in an $(n-1) \times n$-grid. Every node v has height at most* $\deg(v)$.

In our drawings of planar graphs, the half-perimeter of each node is at most $deg(v)$, but the drawing is not necessarily in the Kandinsky-model, since we may violate the bend-or-end property when adding a non-free chain. By changing the placement of non-free chains, we can get a drawing in the Kandinsky-model at the cost of introducing more bends.

Theorem 12. *There exists a linear-time heuristic to draw a planar simple triconnected graph orthogonally planar in the Kandinsky-model in an $(m-1) \times \min\{m-n+1, \frac{m}{2}\}$-grid with $m-2$ bends. Every edge has at most one bend.*

A slight modification of our algorithm produces 2-visibility drawings of small area, where the best previous bound was a half-perimeter of $2n$ [6].

Theorem 13. *There exists a linear-time heuristic to draw a planar simple graph without bends or crossings as a 2-visibility drawing in an $(n-1) \times (n-1)$-grid.*

6 Conclusion

In this paper, we presented an algorithm to compute an orthogonal drawing of a simple graph with arbitrary degrees. We achieved a grid-size of $\frac{m+n}{2} \times \frac{m+n}{2}$. Furthermore, every edge has exactly one bend, thus the number of bends is m. This result improves previous results by a factor that approaches 2 as m gets large relative to n. We also, for the first time, managed to show a non-trivial bound on the box size of at most $\frac{deg(v)}{2}$ for node v. For the incremental scenario, where the drawing has to be produced as the nodes are given in a sequence, we achieve a grid-size of $(\frac{m}{2}+n) \times (\frac{2}{3}m+n)$, which is surprisingly close to the results for the static case.

Many open problems remain:

- In the static scenario, we would like to remove the "$\frac{n}{2}$" terms. It arises if the in-degree or out-degree of a node is odd, since we need to add a $\frac{1}{2}$ as a correction term for rounding. At least one of the two terms cannot be avoided for nodes with odd degrees. But can we reduce it if there are many nodes with even degree?

- It is not clear how much improvement can be achieved in other models. If we drop the restrictions on the size of boxes, does the grid-size get smaller?
- Our algorithm for the static case does not consider planarity of the graph, and in fact, may create $\mathcal{O}(n^2)$ crossings for a planar graph. Can a grid-size of (roughly) $\frac{m}{2}$ in both directions be achieved for planar drawings as well?

References

1. T. Biedl, B. Madden, and I. Tollis. The Three-Phase Method: A Unified Approach to Orthogonal Graph Drawing. Tech. Report, University of Texas at Dallas, UT-DCS 03-97, 1997.
2. T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* MIT Press, McGraw-Hill Book Company, 1990.
3. G. Di Battista, P. Eades, R. Tamassia, and I. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry: Theory and Applications*, 4(5):235–282, 1994.
4. P.F. Dietz and D.D. Sleator. Two algorithms for maintaining order in a list. In *Proc. 19th Annual ACM Symp. Theory of Computing*, pp. 365–372, 1987.
5. S. Even and R.E. Tarjan. Computing an *st*-numbering. *Theoretical Computer Science*, 2:436–441, 1976.
6. U. Fößmeier, G. Kant, and M. Kaufmann. 2-visibility drawings of planar graphs. In S. North, editor, *Symp. on Graph Drawing, GD 96*, volume 1190 of *Lecture Notes in Computer Science*, pp. 155-168. Springer Verlag, 1997.
7. U. Fößmeier and M. Kaufmann. Drawing high degree graphs with low bend numbers. In F. Brandenburg, editor, *Symp. on Graph Drawing, GD 95*, volume 1027 of *Lecture Notes in Computer Science*, pp. 254–266. Springer Verlag, 1996.
8. M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, 1979.
9. J. Hutchinson, T. Shermer, and A. Vince. On representation of some thickness-two graphs. In F. Brandenburg, editor, *Symp. on Graph Drawing, GD 95*, volume 1027 of *Lecture Notes in Computer Science*, pp. 324–332. Springer Verlag, 1996.
10. G. Kant. *Algorithms for Drawing Planar Graphs.* PhD thesis, Univ. Utrecht, 1993.
11. A. Lempel, S. Even, and I. Cederbaum. An algorithm for planarity testing of graphs. In *Proc. of Theory of Graphs, Int. Symp. Rome*, pp. 215–232. 1966.
12. A. Papakostas and I. Tollis. Issues in interactive orthogonal graph drawing. In F. Brandenburg, editor, *Symp. on Graph Drawing, GD 95*, volume 1027 of *Lecture Notes in Computer Science*, pp. 419–430. Springer Verlag, 1996.
13. A. Papakostas and I. Tollis. High-degree orthogonal drawings with small grid-size and few bends. *Workshop on Algorithms and Data Structures, WADS 97*, Lecture Notes in Computer Science. To appear.
14. P. Rosenstiehl and E. Tarjan. Rectilinear Planar Layouts and Bipolar Orientations of Planar Graphs. *Discrete Computational Geometry*, 1:343–353, 1986.
15. R. Tamassia, G. Di Battista, and C. Batini. Automatic graph drawing and readability of diagrams. *IEEE Trans. on Systems, Man and Cybernetics*, 18(1), 1988.
16. R. Tamassia and I. Tollis. A unified approach to visibility representations of planar graphs. *Discrete Computational Geometry*, 1:321–341, 1986.