2-Visibility Drawings of Planar Graphs*

Ulrich Fößmeier[1] Goos Kant[2] Michael Kaufmann[1]

[1] Universität Tübingen, Wilhelm-Schickard-Institut, Sand 13, 72076 Tübingen, Germany
[2] Dept. of Comp. Sci., Utrecht University, Padualaan 14, 3584 CH Utrecht, Netherlands.

Abstract. In a 2-visibility drawing the vertices of a given graph are represented by rectangular boxes and the adjacency relations are expressed by horizontal and vertical lines drawn between the boxes. In this paper we want to emphasize this model as a practical alternative to other representations of graphs, and to demonstrate the quality of the produced drawings. We give several approaches, heuristics as well as provably good algorithms, to represent planar graphs within this model. To this, we present a polynomial time algorithm to compute a bend-minimum orthogonal drawing under the restriction that the number of bends at each edge is at most 1.

1 Introduction

Many algorithms for drawing graphs have been developed in the last years. Comparing them is a difficult task, because the quality of a drawing is not clearly defined, and depends highly on the application. So several models for the representation have been worked out to express the different properties of a graph [3].

One of the simplest and therefore most attractive ways of representation is to draw the edges as polygonal chains consisting of horizontal and vertical line segments. It is commonly used in the area of VLSI-design but also in data base schemes and organisational diagrams. Such drawings can be classified in various classes, where the most extreme ones are: a) *orthogonal drawings* and b) *visibility representations*. In orthogonal drawings all vertices are restricted to have a small uniform size and the edges consist of (several) horizontal and vertical segments. If there are vertices with a degree of more than four, several methods have been worked out how to solve this problem [14, 1, 11]; we adopt the model from [5]. The latter algorithm computes a drawing with the minimum number of bends preserving a given embedding. In Figure 1 we examplify the model using a graph which arises in astrophysics and was already used in the PhD. thesis of Mutzel [10]. We use this graph as a running example to distinguish the different models and approaches in this paper.

A visibility representation is a drawing where all edges are restricted to be single orthogonal line segments. No bends arise, but it is only possible to draw graphs in this way when we allow the vertices to have different sizes; it is even not possible to bound the size of the vertices. The advantages of visibility representations are: They yield very readable pictures for human spectators and the vertices have a suitable expansion in horizontal direction to write some text inside. The theory is quite developed [12, 15] for the case when the edges are restricted to be uni-directional, say

* This research was (partially) supported by DFG-Grant Ka812/4-1, "Graphenzeichnen und Animation" and by ESPRIT Long Term Research Project 20244 (project ALCOM IT: *Algorithms and Complexity in Information Technology*).

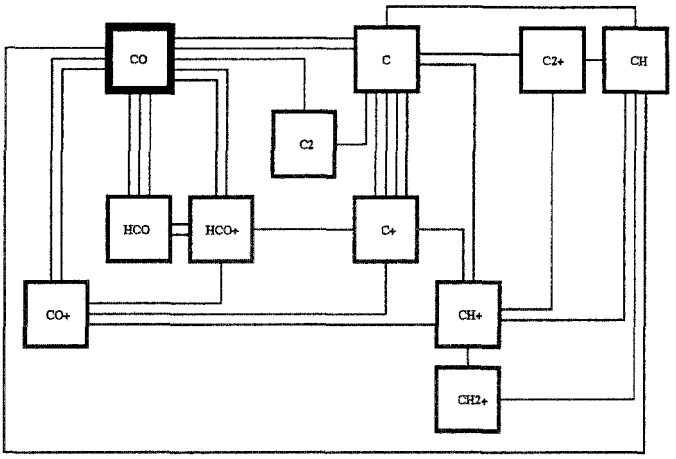


Fig. 1. An orthogonal drawing of the astro-graph

vertical. We call this model the *1-visibility* model. Several distinctions are made: The *strong* visibility model requires that the boxes can see each other if and only if the corresponding vertices are adjacent. In the *weak* visibility model they may see each other even if they are not neighbours. We will only consider the weak model.

Every planar graph can be represented in the weak model. There are efficient algorithms e.g. [7] with good bounds on the required area. 2-visibility representations are a straightforward generalization of the 1-visibility model. Here vertical *and* horizontal edges are allowed. Figure 2 shows a 2-visibility drawing for the graph of Figure 1.

Though the resulting drawings look very promising from a practical point of view, this model has not been considered very often [6, 9, 17, 18] and the known results are very preliminary. Notice that also 2-visibility representations of nonplanar graphs might be possible, though with crossing edges. In this paper we only consider planar graphs and planar representations. The purpose is to introduce this model as a practical alternative to the models used before and to demonstrate the quality of the produced drawings.

We present several heuristics and efficient algorithms to get such representations. The basic idea here is to first draw the graph orthogonally and then stretch the vertices such that they become rectangles and cover all the existing bends. Obviously we have to stretch the vertices in such a way that the final rectangles do not intersect. To make such stretchings possible we produce orthogonal drawings with special properties. Section 2 of this paper is devoted to methods for such orthogonal drawings. In Section 3 we use transform these drawings into visibility representations. Note that Tamassia & Tollis propose just the opposite way in [16]: Starting with a 1-visibility drawing they shrink the vertices and insert the corresponding edges with a number of bends to get nice orthogonal drawings.
We will present the following methods and results concerning the 2-visibility model.

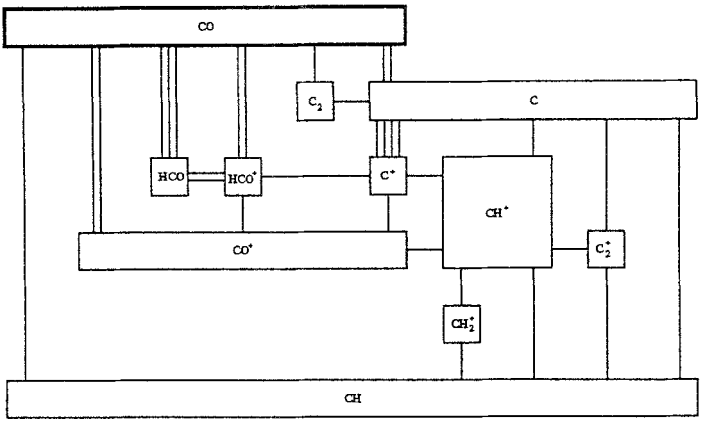


Fig. 2. A 2-visibility drawing of the astro-graph

1. In Section 2.1 we present a linear time algorithm that computes an orthogonal drawing of a planar graph with at most one bend per edge. By applying local transformations this drawing is transformed to a 1-visibility representation or a compact 2-visibility representation (Section 3.1).
2. Two polynomial time algorithms that produce 2-visibility drawings are given in Section 3.2, where the number of horizontal and vertical edges is balanced in some sence. The first algorithm is a modification of the min-cost flow approach of [13], while the latter uses the result developed in Section 2.2. There we show how to compute a minimum-bend orthogonal drawing under the restriction that each edge has at most one bend. In the resulting drawing in Section 3.2, each vertex has a uniform 'small' height.
3. In Section 4 we give an efficient algorithm based on the canonical ordering and prove upper bounds on the used area. We present a class of graphs where the 2-visibility drawing is always nearly as large as described before.
4. Finally, Section 5 contains concluding remarks and directions for further research in this practical field of graph drawings.

2 Orthogonal drawings for high degree planar graphs

In this section we present algorithms for drawing special orthogonal representations of planar graphs without any restriction on the maximum degree. We will use these results in Section 3 to achieve practical 2-visibility representations.

We take the orthogonal drawing model from [5] (notice that this model is different from the usual orthogonal drawing definition for 4-planar graphs). Because of space limitation, we refer to Figure 1 where the properties of the model can clearly be seen, instead of giving formal definitions. Most noticeably, there is at most one straight edge on each side of every vertex. The idea for achieving 2-visibility representations is that the vertices are stretched to rectangles such that the bends on the edges disappear. Note that in general this method does not work on orthogonal drawings when edges with more than one bend exist, cf. Figure 4. The stretching is possible

if every edge is restricted to have at most one bend. The algorithms in our first and third approach use such orthogonal drawings as an intermediate step. The methods are probably interesting on their own.

2.1 Orthogonal drawing with at most one bend per edge

Theorem 1 *For every planar graph $G = (V, E)$ with a planar embedding there is an orthogonal drawing for G preserving the planar embedding with at most one bend on every edge and this drawing can be computed in linear time.*

Proof. At first we triangulate G, i.e., we add dummy edges to G such that every face of G is a triangle. Next we compute a *canonical ordering* for the triangulated graph G' [4]. That means that the vertices are numbered $1, \ldots, n$ such that

- the external face consists of the vertices 1, 2 and n and
- for every $i \geq 3$ there is a vertex v_i on the external face of G_i that has at least two neighbours in G_{i-1} and at least one neighbour in $G \backslash G_i$ and G_i is biconnected. The neighbours of v_i in G_{i-1} form a consecutive sequence on the outerface of the embedding of G_{i-1}. G_i is the subgraph of G' consisting of the vertices $v_1, \ldots, v_i$.

De Fraysseix, Pach & Pollack [4] show how to compute a canonical ordering of a triangulated planar graph in linear time. Our algorithm places vertices v_1 and v_2 at coordinates (0,1) and (1,0) and adds the rest of the vertices in the order of the canonical ordering. The vertices are placed on grid points of an integer grid such that there is only one vertex on every line and on every column. By definition of the canonical ordering, the neighbours of v_i in G_{i-1} form an interval of the external face of G_{i-1}; let v_l be the leftmost and v_r the rightmost vertex in this interval. Insert a new grid line directly below the line of v_l and a new column directly to the left of v_r, place v_i on the intersection point of the new line and the new column and draw edges with one bend per edge such that edge (v_l, v_i) is incident to v_i at its left side and all other incident edges in G_i are incident to v_i at its bottom side (see Figure 3). To ensure that this is always possible without creating crossings between edges or between an edge and a vertex we show the following invariant: The contour of the external face of G_i between v_1 and v_2 (without the edge (v_1, v_2)) is a staircase from the left to the right. It is easy to see that the new edges do not cross any old objects if the invariant holds (Figure 3) and that adding a new vertex does not destroy the invariant. This proves the theorem.

2.2 Bend-minimum drawings with at most one bend per edge

We want to apply the stretching idea to bend-minimum orthogonal drawings hoping that fewer vertices might be stretched and/or vertices are stretched by a smaller amount when we minimized the number of bends before. In this subsection we show how to produce bend-minimum orthogonal drawings under the restriction that each edge has only one bend. In Subsection 3.2 we discuss the properties of the drawing when we stretch the vertices. We also motivate this approach using the bend-minimum orthogonal drawing from Figure 1 and demonstrate in Figure 4 why it is important to restrict the number of bends per edge to be at most 1.

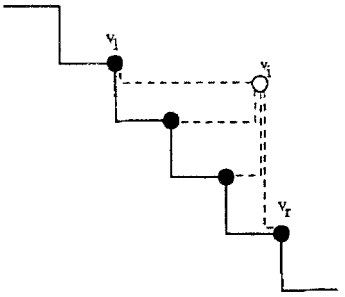


Fig. 3. A new vertex is added to G_{i-1}

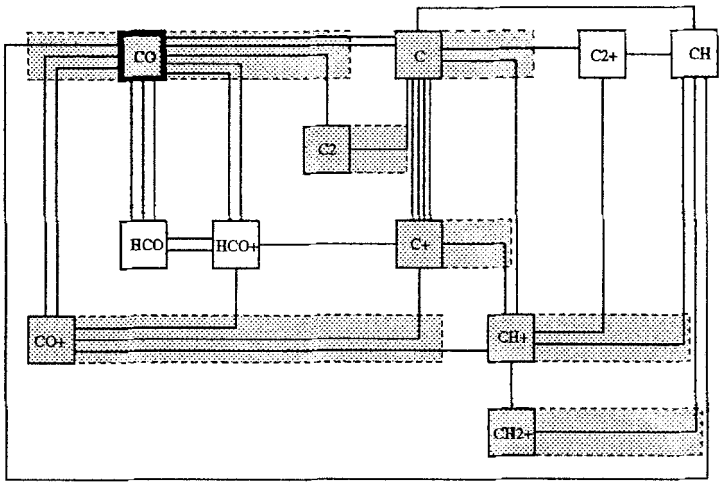


Fig. 4. From an orthogonal drawing to a visibility representation.

Theorem 2 *For every planar graph $G = (V, E)$ with a planar embedding, we can efficiently compute a bend-minimum drawing under the restrictions that the planar embedding is preserved, the area for each face is non-empty and the number of bends on each edge is at most 1.*

Proof. To restrict the drawing such that each edge has only at most one bend is easy in the case of the original model of Tamassia [13] where $0°$-angles are forbidden and the graph is 4-planar. In this case we only need to restrict the capacities of the face-to-face-arcs in the network to be 1. Then at most one bend may happen per edge since only one unit of flow may use this edge.

In the general case of higher-degree vertices we have to change the network considerably. We will motivate and describe only the changes that have to be made with respect to the approach in [5]. Note that we will again require that the faces will be represented by a polygon with a non-empty area.

The key fact again is that for each 0^o-angle there is a unique bend. We extend the number of forbidden configurations, such that e.g. two bends on the same edge corresponding to different 0^o-angles will not arise. Fortunately, the construction of the network becomes simpler than in [5]. We shortly review the original construction from [13]. The solution of such a flow problem leads to a orthogonal representation.

Tamassia defined the network N_H as follows: $N_H = (U, A, s, t, b, c)$ where $b : A \to \mathbb{R}^+$ is a nonnegative capacity function, $c : A \to \mathbb{R}$ is a cost function, U (the *nodes* of the network) $= \{s\} \cup \{t\} \cup U_V \cup U_F$, where s and t are the *source* and the *sink* of the network, U_V contains a node for every vertex of G and U_F contains a node for every face of G, A (the *arcs* of the network) contains

a) arcs from s to nodes v in U_V with cost 0 and capacity $4 - deg(v)$;
b) arcs from s to nodes f in U_F, where f represents an internal face of G with $deg(f) \leq 3$; these arcs have cost 0 and capacity $4 - deg(f)$; $deg(f)$ for a face f always denotes the number of edges in the list $H(f)$;
c) arcs from nodes f in U_F representing the external face or representing internal faces f with $deg(f) \geq 5$ to t; these arcs have cost 0 and capacity $deg(f) - 4$ if f is an internal face and capacity $deg(f) + 4$ for the external face;
d) arcs of cost 0 and capacity ∞ from nodes v in U_V to nodes f in U_F, if v is incident to an edge of $H(f)$;
e) arcs of cost 1 and capacity ∞ from a node f in U_F to a node g in U_F, whenever the faces f and g of G have at least one common edge.

Every flow unit on an arc between two faces stands for a bend on an edge between these faces. The flow on the arcs in d) defines the angles in the drawing: If $x_{v,f}$ is the flow from the node $v \in U_V$ to the node $f \in U_F$ then the angle at vertex v in face f is $(x_{v,f} + 1) \cdot 90°$. Every feasible flow of value $\Sigma_u b(s,u) = \Sigma_w b(w,t)$ with cost B leads to an orthogonal representation with exactly B bends. Thus the cost minimum solution of the flow problem corresponds to the bend minimum drawing.

Allowing 0^o-angles is easy. We extend the rules as follows:

According to the formula above such an angle corresponds to a flow of value -1 from some $v \in U_V$ to some $f \in U_F$. We interprete this as a flow of value +1 in the opposite direction, from f to v. Thus, in the network there are some additional arcs:

f) arcs of cost 0 and capacity $deg(v) - 4$ from nodes v in U_V to t, if $deg(v) \geq 5$; and
g) arcs of cost 0 and capacity 1 from a node f in U_F to a node v in U_V, whenever there is an arc of type d) from v to f.

In [5], we solved the problem that certain configurations in the network should not happen by some quite complicated modifications. Now, we also have to modify the network such that there are no two units of flow crossing one single edge. Therefore we replace the rules e) and f) by the construction shown in Figure 5.

Note that all capacities are 1 and all costs not indicated are 0. By the trick to punish the use of an arc first by costs $2c + 1$ and then to pay c costs twice back we make sure that each edge is crossed only once. A cost of 1 remains, corresponding to a single bend as before. The additional use of the nodes H_f and H_{f_e} ensures that the forbidden configurations already discussed in [5] will not occur. The introduction of nodes H_{f_e} is necessary since the flow into the face-node f across e can only go directly into the vertex-nodes adjacent to e or into the face-node f via the arc (H_{f_e}, f).

Choosing the cost parameter c sufficiently large and solving the min-cost-flow problem as usually leads to a bend-minimum orthogonal drawing of the graph.

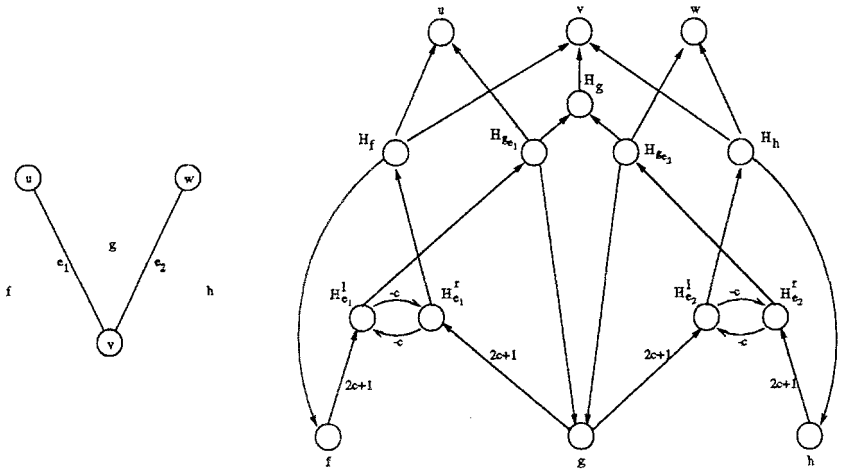


Fig. 5. Restricting the flow across each single edge

3 Three approaches to 2-visibility drawings

3.1 A first heuristic using local improvements

As a first result in this section we give an algorithm that computes a 1-visibility representation of a given planar graph preserving a given planar embedding. The idea is to start with a 1-bend orthogonal drawing as described in Section 2.1 and to delete bends by expanding vertices. In particular, the algorithm chooses a direction (w.l.o.g. horizontal), stretches the vertices horizontally such that any two neighboured edges being incident to the same vertex at its bottom side have distance at least 1 from each other; The vertex of the visibility drawing gets the y-coordinate of the vertex of the orthogonal drawing and will be extended in horizontal direction such that it covers all horizontally incident bends. The remaining edges are the vertical segments of the edges in the orthogonal drawing. It is clear that this method does not create any crossings.

This leads to a 1-visibility drawing with an area being slightly larger than the corresponding orthogonal drawing because of the first stretching of the vertices.

Our first approach to get a 'real' 2-visibility drawing (every 1-visibility drawing is a special case of a 2-visibility drawing) is to change locally the orthogonal drawing obtained by the algorithm described in Section 2.1 such that we save unnecessary bends. The resulting straight edges remain unchanged by the stretching algorithm and may run horizontally or vertically. The vertical segment of every edge that our algorithm from 2.1 creates is incident to some vertex at its bottom side; so there are two kinds of edges: Edges of type (i) are incident to the other vertex at its left side and edges of type (ii) are incident to the other vertex at its right side. So there are four ways to save a bend which are shown in Figure 6; operations (a) and (b) concern edges of type (i) and the other operations concern edges of type (ii).

A bend-saving operation can only be applied if the graph (locally) fulfills some conditions; so there are some rules to decide which operation can be realized.

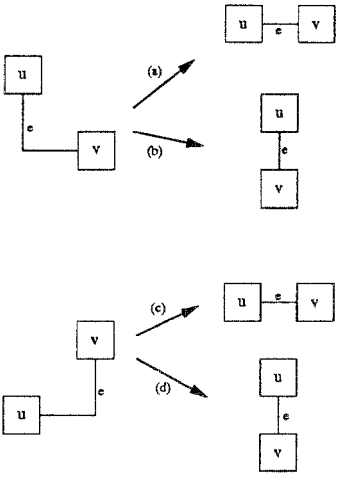


Fig. 6. How to save bends

Operation (a) can be applied if

1. e is the lowest edge being incident to vertex v at v's left side and
2. e is the rightmost edge being incident to vertex u at u's lower side and
3. there is no edge being incident to vertex u at u's right side.

Operations (b),(c) and (d) can be applied in symmetric cases. It can be seen easily that there appear no crossings if the bend-saving part of the algorithm obeys these rules. It is not hard to implement these local improvement steps such that it works in linear time overall. Applying our first algorithm to the astro-graph we get the drawing of Figure 7.

3.2 Two approaches for balanced 2-visibility drawings

In this subsection we give two more involved algorithms that lead to more balanced 2-visibility drawings. The algorithm of Section 3.1 produces vertices of uniform heights, but it clearly prefers one dimension against the other: The remainings of all the originally bending edges are drawn vertically, only edges being a result of the bend-saving step might run horizontally. Now we want to balance the two dimensions somehow. For that purpose we minimize the number of 0°-angles between edges (a 0°-angle arises whenever two neighboured edges are incident to a vertex at its same side). In other words: we try to use each side of the rectangle (representing the vertex) to connect edges at. Although this does not guarantee a bound for the ratio between the number of vertical edges and the number of horizontal edges, an equilibrium can be observed in practical examples.

'Non-uniform' vertices. We use a variant of the algorithms presented in Section [5] and [13], based on network flow techniques. We shortly reviewed it already in

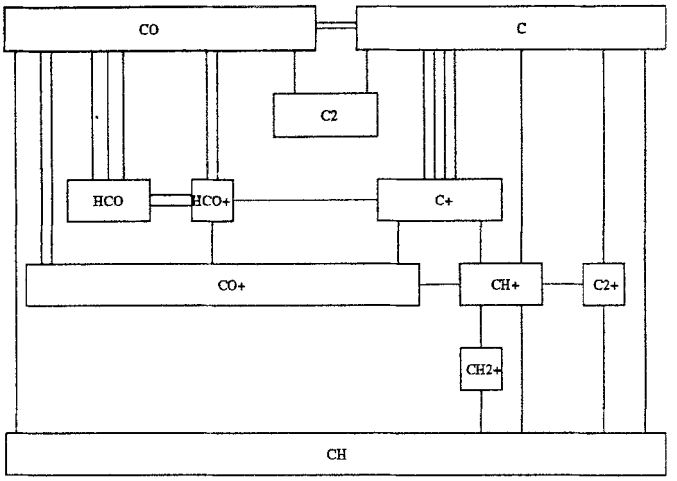


Fig. 7. Applying the local improvement algorithm to the astro-graph.

2.2. In these approaches a network is constructed having vertices and faces of the embedded planar graph as nodes. A feasible flow in this network corresponds to a drawing of the graph: A flow unit between adjacent faces characterizes a bend on an edge between these faces. A flow from a vertex v into a face f defines the angle between two edges of f having v as common vertex. In 2-visibility drawings we do not have any bends but angles of 0° are allowed arbitrarily. So we delete all arcs between two faces in the network. We add arcs from each face to each vertex being contained in the face with capacity 1 and positive cost defining the 0° angles; as a consequence a min-cost flow in this network corresponds to a no-bend drawing having a minimum number of 0° angles. Hence this yields a 2-visibility representation. Applying this algorithm to the astro graph yields the drawing of Figure 2 displayed before.

Keeping the size of the vertices 'small'. In this subsection we combine the network approach of [13] and [5], refined in Section 2.2 with the stretching idea already used in Section 3.1. We showed how to produce a bend-minimum orthogonal drawing where each edge has only one bend. Now we can easily stretch the vertices such that they cover all bends, even using only one direction (see Figure 4). Moreover, we can choose the stretching direction under some criteria like 'the used area ' or 'balance of horizontal and vertical edges' or 'sizes of the rectangles of the vertices'. Suppose we only stretch the vertices in horizontal direction. Then, since in our orthogonal drawing the vertices have a squarish shape we get a drawing where all vertices have the same 'small' height only depending on the degree of the graph. This avoids high and skinny rectangles which are possible in the second approach and enables a reasonable vertex labeling.

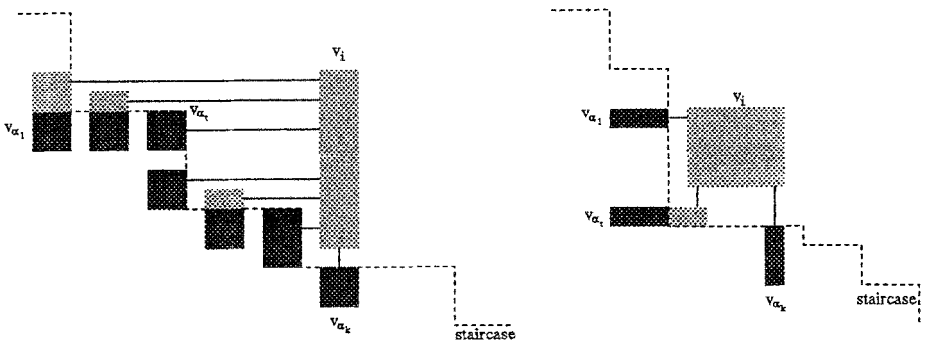
However, notice that the width of the rectangles can increase arbitrarily; if the user insists also on a small width which only depends on the size of the labeling, we propose the following technique: shrink the width of the vertices, such that only a few

adjacent vertices are not visible anymore. Inserting the edges now with some bends is an easy task and mostly leaves the size of several vertices unchanged. Obviously, this contradicts the model of 2-visibility, nevertheless it might be a practical approach for taking the size of the labeling into account.

4 Upper and lower bounds on the area

In the following, we present an alternative linear time algorithm and analyse its behaviour with respect to the used area. For counting the area, we determine the corner coordinates of the rectangles to be integers and the rectangles to have at least a size of 1×1. The edges are placed at half-integer coordinates. So, the area in Figure 9 (b) is 6×7.

Let $G = (V, E)$ be the embedded planar graph.
If G is not triangulated, add dummy edges to it to make it so.
Compute the canonical ordering of G, denoted by $v_1, \ldots, v_n$.
Place the vertices v_1 and v_2 as boxes of size 2×1 and 1×1 in an L-shape.
for $i := 3$ **to** n **do**
  Let $v_{\alpha_1}, \ldots, v_{\alpha_k}$ be the neighbours of v_i in G_{i-1} from left to right.
  Let α_t be the maximum index, with $1 \leq t \leq k$.
  Place v_i above and/or to the right of its neighbours in the drawing of G_{i-1}
    s.t. the edge (v_{α_1}, v_i) will be horizontal and (v_{α_k}, v_i) will be vertical.
  Stretch all rectangles to the right or to the top such that
    they can see v_i if they are adjacent to v_i and such that the
    stair-case invariant is maintained.



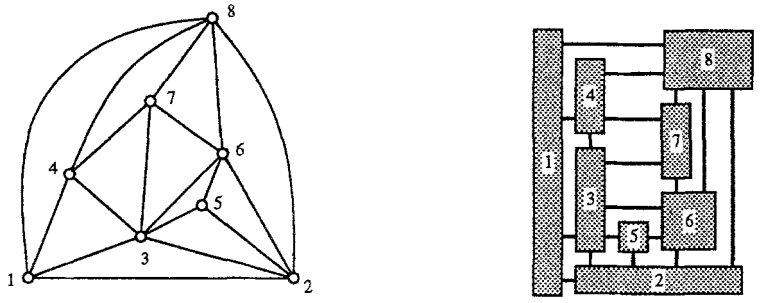
(a) The general configuration

(b) The four-connected case

Fig. 8. Two cases: Before and after insertion of vertex v_i.

From the stretching approach, described in Section 2.1, we have to come to an exact computation of every rectangle in the representation. For this computation a

simple adaption of the 'Shift'-method of Chrobak & Payne yields the desired result [2]. In Figure 9 an example is given.



(a) A canonical numbered triangulated planar graph

(b) The corresponding 2–visibility representation

Fig. 9. Example of the algorithm.

Observe that the layout will be stretched when inserting vertex v_i by the number of incoming edges to $v_i - 1$. The -1 comes from the fact that we do not need to stretch the highest indexed vertex v_{α_t}. Summing this up over all vertices ($\sum_i indegree(v_i) - 1 \leq 3n - 6 - n$) gives a bound of $2n$ on the sum of the height and the width of the layout. We state this result in the following

Theorem 3 *For every planar graph $G = (V, E)$ with $|V| = n$, there is a 2-visibility drawing for G that uses an area $x \times y$ where $x + y \leq 2 \cdot n$.*

The bound of $x+y \leq 2 \cdot n$ is tight, since it is not difficult to construct a triangulated planar graph, for which this algorithm indeed requires this area. Notice also that the required area heavily depends on the chosen canonical ordering. Again, it is not difficult to construct a planar graph, requiring an area of size $n \times n$ for one ordering, and an area of size $n \times 3$ for another canonical ordering. Using some more refinements of the canonical ordering, it is not hard to improve the theorem above by the fact $x \leq n$ and $y \leq n$, but it is certainly not trivial to improve the $2 \cdot n$ bound.

However, for 4-connected planar graphs we can easily improve this area bound. For this purpose, we use an observation made in [8]. When we insert vertex v_i with the incoming edges from $v_{\alpha_1}, \ldots, v_{\alpha_k}$ there is only one local minimum in the sequence of these vertices. Let v_{α_t} be the vertex with minimal index in the sequence. Note that in the current representation v_{α_k} lies at the point where a vertical segment of the staircase hits a horizontal one.

Here it is sufficient to stretch only this single vertex v_{α_t} by one unit such that it can see v_i, and to update the staircase-structure (see Figure 8). This means that inserting vertex v_i stretches the layout by only one unit. Moreover, similar as in the 3-connected case, we have the choice whether we stretch in horizontal or vertical direction. Hence we can balance width and height of the layout such that our result is the following:

Theorem 4 *For every 4-connected planar graph $G = (V, E)$ with $|V| = n$, there is a 2-visibility drawing for G that uses an area $x \times y$ where $x = y \leq n/2$.*

The following lower bounds shows that the upper bounds achieved by the linear time algorithms above are getting close to the lower bounds.

Theorem 5 *There is a planar graph $G = (V, E)$ with $|V| = n$, such that all 2-visibility drawings for G have an area $x \times y$ where $x + y \geq 5/3 \cdot n$, $x \geq 2/3 \cdot n$ and $y \geq 2/3 \cdot n$.*

Proof. Let G be the graph of $n/3$ nested triangles, which are internally triangulated. Note that each triangle needs at least 2 horizontal and 2 vertical lines for the realization. Since the triangles are nested, the width and the height of the layout is at least $2n/3$.

We can get an even better bound on the area if we determine the triangulation in the way as shown in the next figure. Extensive case analysis shows that for the realization of the triangulation an extra horizontal or vertical line is necessary, such that height + width is increased by 5 units when we add a new triangle.

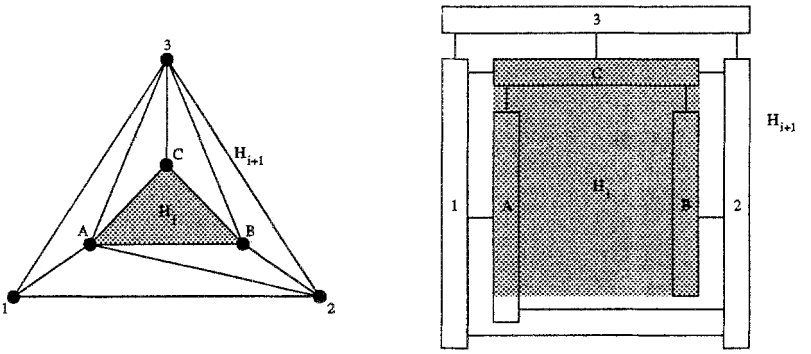


Fig. 10. The lower bound example drawn in height + width $\leq 5/3 \cdot n$

Applying the same approach for a set of nested rectangles, we get the following:

Theorem 6 *There is a 4-connected planar graph $G = (V, E)$ with $|V| = n$, such that all 2-visibility drawings for G have an area $x \times y$ where $x \geq n/2$ and $y \geq n/2$.*

5 Conclusion

In this paper we presented several practical approaches for producing 2-visibility drawings of planar graphs. The purpose of this paper is to emphasize this model and to demonstrate the quality of the resulting drawings. Moreover, we also presented theoretical upper bounds with respect to the required area. Our opinion is that that

for practical purposes 'weak' visibility drawings have no disadvantages compared to 'strong' visibility drawings, and will lead to improved practical drawings.

On the other hand, there is certainly a lack of theory. Up to now there are hardly any properties and theorems derived for 2-visibility as it happened for 1-visibility. The reason is that the 2-visibility concept is much stronger as it is used here. We only used it for planar drawings, but even some thickness-two graphs (sparse non-planar graphs) can be drawn in this model, of course yielding crossings of horizontal and vertical edges. Unfortunately, unclear is the exact characterization of the class of graphs, admitting a 2-visibility representation with crossing edges. Hence, more theoretical work is required on the area bounds, as well as a precise characterization of the balance between horizontal and vertical edges. Also theory with respect to the maximum and total edge length is a field for further research.

On the practical side, it is interesting to test the behaviour of the presented algorithms for several different subclasses of planar graphs. Especially the influence of the edge balance on the size of the rectangles for the vertices and the area yields very interesting questions.

References

1. Biedl T. and G. Kant, A better heurisitic for orthogonal graph drawings, *Proc. 2nd Ann. European Symposium on Algorithms (ESA'94)*, LNCS 855, Springer-Verlag, pp. 24-35, 1994.
2. Chrobak, M., and T.H. Payne, A Linear Time Algorithm for Drawing Planar Graphs on the Grid, *Tech. Rep. UCR-CS-90-2*, Dept. of Math. and Comp. Science, University of California at Riverside, 1990.
3. Di Battista G., P. Eades, R. Tamassia and I.G. Tollis, Algorithms for automatic graph drawing: an annotated bibliography, *Computational Geometry: Theory and Practice* 4, pp. 235–282, 1994.
4. Fraysseix, H. de, J. Pach and R. Pollack, How to draw a planar graph on a grid, *Combinatorica* 10, pp. 41–51, 1990.
5. Fößmeier, U., and M. Kaufmann, Drawing high degree graphs with low bend numbers, *Proc. 4th Symposium on Graph Drawing (GD'95)*, LNCS 1027, Springer-Verlag, pp. 254-266, 1995.
6. Hutchinson, J.P., T. Shermer and A. Vince, On representation of some thickness-two graphs, *Proc. 4th Symposium on Graph Drawing (GD'95)*, LNCS 1027, Springer-Verlag, pp. 324-332, 1996.
7. Kant, G., A more compact visibility representation, *Proc. 19th Intern. Workshop on Graph-Theoretic Concepts in Comp. Science (WG'93)*, LNCS 790, Springer-Verlag, pp. 411–424, 1994.
8. Kant, G., and X. He, Two algorithms for finding rectangular duals of planar graphs, *Proc. 19th Intern. Workshop on Graph-Theoretic Concepts in Comp. Science (WG'93)*, LNCS 790, Springer-Verlag, pp. 396–410, 1994.
9. Kirkpatrick, D.G., and S.K. Wismath, Weighted visibility graphs of bars and related flow problems, *Proc. 1st Workshop Algorithms Data Structures (WADS'89)*, LNCS 382, Springer-Verlag, pp. 325-334, 1989.
10. Mutzel, P., *The Maximum Planar Subgraph Problem*, Doctoral Dissertation, Köln 1994.
11. Papakostas A. and I. Tollis, Improved algorithms and bounds for orthogonal drawings, *Proc. DIMACS Workshop on Graph Drawing (GD'94)*, LNCS 894, Springer-Verlag, pp. 40–51, 1994.
12. Rosenstiehl, P., and R.E. Tarjan, Rectilinear planar layouts and bipolar orientations of planar graphs, *Discrete Comput. Geom.* 1, pp. 343-353, 1986.

13. Tamassia, R., On embedding a graph in the grid with the minimum number of bends, *SIAM Journal of Computing* 16, pp. 421-444, 1987.
14. Tamassia, R., G. Di Battista and C. Batini, Automatic graph drawing and readability of diagrams, *IEEE Trans. on Systems, Man and Cybernetics* 18, pp. 61–79, 1988.
15. Tamassia R. and I. Tollis, A unified approach to visibility representations of planar graphs, *Discrete and Computational Geometry* 1, pp. 321–341, 1986.
16. Tamassia, R., and I.G. Tollis, Efficient embedding of planar graphs in linear time, in: *Proc. IEEE Int. Symp. on Circuits and Systems*, Philadelphia, pp. 495–498, 1987.
17. Thomassen, C., Rectilinear drawings of graphs, *J. Graph Theory* 12, pp. 335–341, 1988.
18. Wismath, S.K., Characterizing bar line-of-sight graphs, *Proc. 1st Annual ACM Symp. on Computational Geometry*, pp. 147–152, 1985.

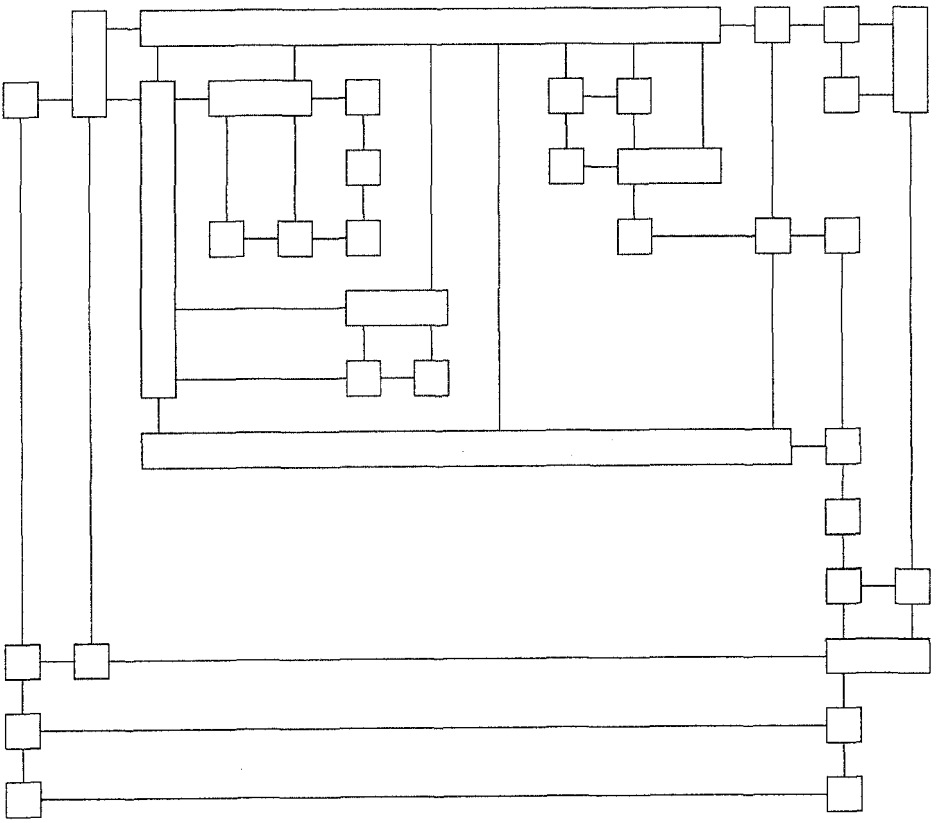


Fig. 11. Applying the first algorithm of Section 3.2 to the A-graph of the competition in GD'95 (two edges are omitted due to planarity)