Drawing High Degree Graphs with Low Bend Numbers

Ulrich Fößmeier Michael Kaufmann

Universität Tübingen, Wilhelm-Schickard-Institut, Sand 13, 72076 Tübingen, Germany,
email: foessmei / mk@informatik.uni-tuebingen.de

Abstract. We consider the problem of drawing plane graphs with an arbitrarily high vertex degree orthogonally into the plane such that the number of bends on the edges should be minimized. It has been known how to achieve the bend minimum without any restriction of the size of the vertices. Naturally, the vertices should be represented by uniformly small squares. In addition we might require that each face should be represented by a non-empty region. This would allow a labeling of the faces. We present an efficient algorithm which provably achieves the bend minimum following these constraints. Omitting the latter requirement we conjecture that the problem becomes NP-hard. For that case we give advices for good approximations. We demonstrate the effectiveness of our approaches giving some interesting examples.

1 Introduction

Embedding planar graphs into a grid while optimizing quality parameters like area, edge lengths or bend number is not only a challenging combinatorial problem, but viewed as a graph drawing problem, it has many direct practical applications.

A drawing of a graph in the plane roughly consists of a representation of the vertices by geometric objects (circles, squares, rectangles, lines, ...) and an assignment of the vertices to geometric positions in the plane. The edges are represented by curves between the objects representing two vertices; in orthogonal representations they form a continous sequence of horizontal and vertical line segments embedded in the plane. Intersections of a horizontal and a vertical line segment belonging to the same curve are called *bends*. The goal in graph drawing is the production of *nice* drawings: The representation of the graph should be simple, the incidence structure should be easily readable, planar graphs should be drawn planar, the vertices should be distributed nicely in the drawing. There are many other requirements, which might be even contradicting.

Orthogonal drawings of planar graphs have been extensively studied in the last decade. There are plenty of applications like VLSI-design [10], entity relationship and data flow diagrams in Software Engineering (e.g. [5, 12, 2, 14, 1]) or visualization of interactions inside of molecular structures in Astrophysics [3]. In some of them multiple edges are necessary, so we have to handle multigraphs. Many results and algorithms have been worked out [4], one of the most important is Tamassia's algorithm from [16]: How to draw a 4-planar graph orthogonally with the minimum number of bends preserving a given planar representation. (A 4-planar graph has a vertex degree of at most 4. A planar representation is given by a fixed cyclic order of the incident edges of each vertex.) We call a planar graph together with a planar representation a *plane* graph.

It makes sense to preserve the planar representation, since very often a (non orthogonal) drawing is given which should have a certain familiarity with the orthogonal layout. Moreover an efficient algorithm for the problem without this restriction cannot be expected since finding a bend minimum solution over all planar representations is NP-hard [8].

In most of the applications it is necessary to visualize planar graphs with a vertex degree of more than four; this aspect cannot naturally be captured in a usual orthogonal drawing, so we have to look for extensions of Tamassia's approach.

Approaches like to proceed to k-gonal grids if the maximum degree is k or to split each vertex of large degree into cluster of many subvertices of degree 3 or 4 turn out to be useless or to be at most good heuristics which may terribly fail in some cases. Our target is a provably good quality of the drawing under the commonly used standards.

Our criterion of optimum is the number of bends. Note that there is always a representation without any bends if we demand no other qualities of the drawing: A *visibility representation* e.g. [17, 13] satisfies all requirements established so far. But in such drawings the size of the vertices may grow independently of the vertex degree: Fig. 1a) shows an example where the size of a vertex grows arbitrarily whereas the degree of the vertex v is a small constant (in the example: 5).

Drawings like Fig. 1a) do not fulfill our wishes in many applications. The size of the vertices should be determined by the degree and not by the structure of the graph. More concrete our model is as follows: We use a grid with uniform distance λ between the grid lines. The size of the vertices should be smaller than λ, and their centers should be placed on the intersections of the grid lines; this ensures that no vertex is intersected by any grid line except of those defining its position and consequently two vertices can never intersect. Let max be the maximal number of edges being incident to a single side of any vertex. We require every vertex to be a square with side length $(2 \cdot max - 1)$. The motivation for the factor 2 is given in section 4. We choose λ to be twice the side length of the vertices. This leaves enough space to route the edges.

With these requirements we ensure a clear and well-arranged look of the drawing. Note that for each vertex v, no two different adjacent vertices can be connected by straight lines emanating from the same side of v.

Segments of edges being incident to the same side of some vertex may run very close together (i.e. the distance between two of them is much smaller than λ) and so it is possible to draw a face in such a way that every point inside of the face has a tiny distance to some bounding edge of the face (see face C in Fig. 1b). We call such faces *empty faces*.

We consider two models: In the first model we allow empty faces and call the corresponding drawings 'Planar Orthogonal Drawings with Equal Vertex Size' (*podevs*). An example can be seen in Fig. 1b. Trying to meet the general podevs-requirements, we noticed that the possible representation of the faces by empty regions (face C in Fig. 1b) does not fit in our efficient approach. We conjecture that in general the bend-minimization problem for podevs is NP-hard. Therefore in our second model we additionally require that at least the non-trivial faces (faces with at least three adjacent vertices) should have a non-empty region. An optimal drawing with this constraint is given in Fig. 1c). We call such drawings 'Planar Orthogonal Drawings with Equal Vertex Size and Non-Empty Faces' (*podevsnef*). Note that podevsnef-drawings always allow a (reasonable) labeling of the faces whereas this is impossible in the case of faces which are represented by an empty region.

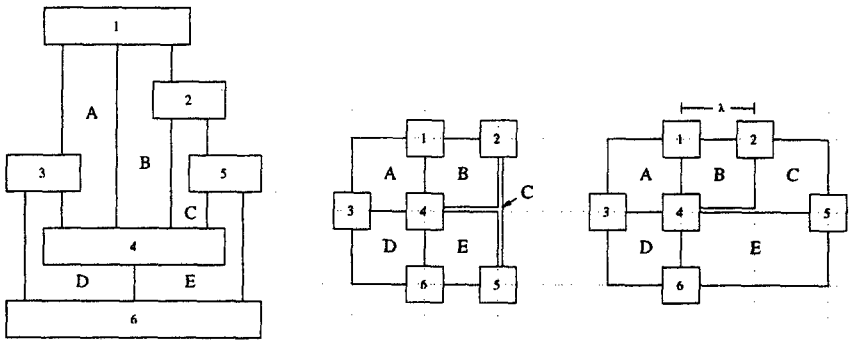


Fig. 1.

a) A visibility representation

b) An optimal solution (podevs)

c) An optimal solution allowing labeling of the faces (podevsnef)

The rest of this paper is organized as follows: In section 2 we perform the first modifications on Tamassia's algorithm and receive some kind of two-dimensional visibility representation. In section 3 the main algorithm (for drawings like in Fig. 1c)) is given. We incorporate the requirements of small uniform square sizes for each vertex and of non-zero area for the faces. In section 4 we shortly describe the postprocessing transformation from the topological to the geometric layout, which is called compaction. Some remarks on further improvements (heuristics for empty faces, compact visibility representations using the approach in section 2) conclude the paper. Many proofs and details are omitted and can be found in [7].

2 Nearly Orthogonal Representations

The basis of our data structure is the *orthogonal representation* defined in [16]: Given a graph $G = (V, E)$ and a planar representation for it; an orthogonal representation for G and its planar representation is a set of lists $H(f)$, one for every face f of G, whose elements are triples $((u, v), s, a)$, where

— (u, v) is an edge of G,
— s is a binary string, and
— a is an integer in the set {90, 180, 270, 360}.

If f is an internal face the edges in $H(f)$ appear in clockwise order and counterclockwise otherwise. The string $s[(u, v)]$ describes the bends of the edge (u, v): The kth bit of $s[(u, v)]$ represents the kth bend that appears at the right side of (u, v), as it is encountered when going along (u, v). The binary symbols **0** and **1** represent angles of 90 and 270 degrees, respectively. $a[(u, v)]$ specifies the angle formed in face f at the vertex v by the edge (u, v) and the following edge in $H(f)$.

For our purposes we have to modify this concept and use *nearly orthogonal representations*. We assign 0°-values to angles between two parallel edges being incident to the same vertex at its same, because in this case the representation remains consistent such that the sum of the interior angles of a polygon with k edges is equal to $(k - 2) \cdot 180°$. So in a nearly orthogonal representation $a[(u, v)]$ is a number in the set {0, 90, 180, 270, 360}.

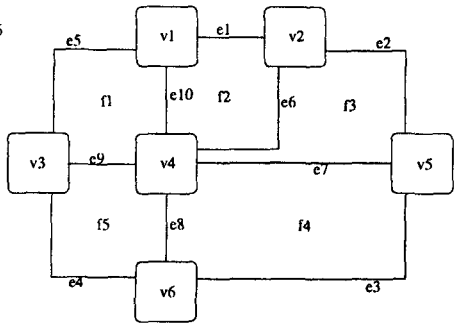


Fig. 2. A Nearly Orthogonal Representation

Fig. 2 shows a drawing for the graph of Fig. 1 with edge identifiers. The nearly orthogonal representation of the example in Fig. 2 is:
$f_1 = ((e_9, \varepsilon, 90), (e_5, 0, 90), (e_{10}, \varepsilon, 90))$;
$f_2 = ((e_{10}, \varepsilon, 90), (e_1, \varepsilon, 90), (e_6, 0, 90))$;
$f_3 = ((e_6, 1, 90), (e_2, 0, 90), (e_7, \varepsilon, 0))$;
$f_4 = ((e_7, \varepsilon, 90), (e_3, 0, 90), (e_8, \varepsilon, 90))$;
$f_5 = ((e_4, 0, 90), (e_9, \varepsilon, 90), (e_8, \varepsilon, 90))$;
$f_6 = ((e_1, \varepsilon, 180), (e_5, 1, 180), (e_4, 1, 180), (e_3, 1, 180), (e_2, 1, 180))$.
Note the angle of 0° in the face f_3.

In [16] Tamassia describes a 1:1-correspondence between an orthogonal representation H of a graph G and a flow in some network N_H defined 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 of H: 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

corresponds to an orthogonal representation H with exactly B bends. Thus the cost minimum solution of the flow problem corresponds to the bend minimum drawing.

In a nearly orthogonal representation H_0 we have to handle angles of degree 0. 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.

Fig. 3 shows the network and the flow for the drawing of Fig. 2; only arcs with nonzero flow are drawn. Note the arcs from face f_3 to vertex v_4 and from vertex v_4 to t.

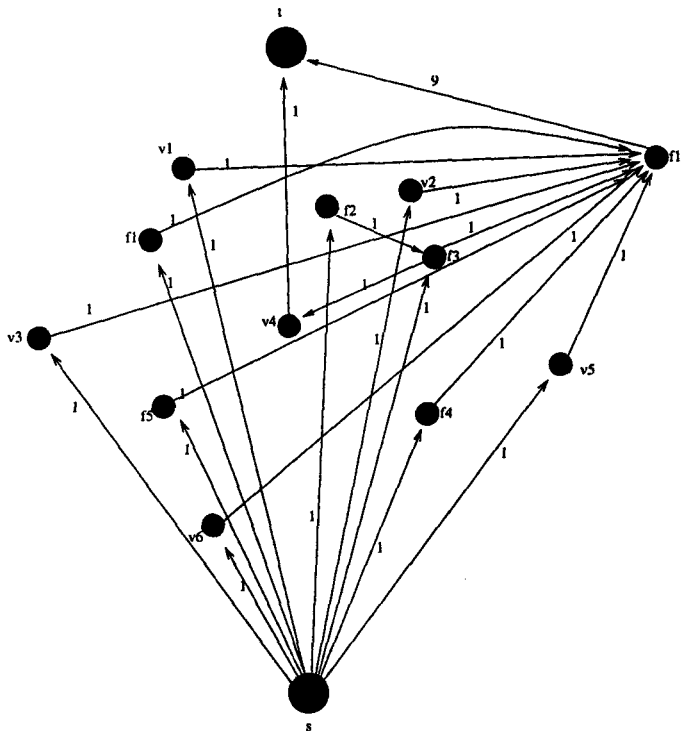


Fig. 3. Network and Flow for the drawing of Fig. 2

The flow in Fig. 3 is not optimal: Since arcs of type d) as well as arcs of type g) have cost zero, it is always possible to establish a zero cost flow in the network. This flow leads to a drawing resembling a visibility representation drawing without any bends, but with very large vertices (cf. Fig. 1a)).

We discuss the quality of such drawings and possible extensions in section 5. In the next section we modify the network in order to get drawings with vertices that are not much larger than necessary.

3 A Bend-Minimizing Algorithm

We shortly recall the requirements for the model given in Section 1. We want to generate drawings like the one in Fig. 1b or 1c respectively, that means: All vertices have square shape and the same size and the centers of the squares lie on grid points of a grid where the unit distance is twice the length of a square side.

We already mentioned that at most one edge being incident to a certain side of some vertex can be a straight line. The only exception are a group of consecutive multiple edges between two vertices v and w, i.e. multiple edges arising in consecutive order in the adjacency lists of v and w, thus defining faces of degree two. We call this (these) edge(s) without bend the *middle edge(s)*.
Moreover, all edges on the same side of v at the left of the middle edge (counterclockwise) are required to bend to the left and the edges at the right of the middle edge (clockwise) bend to the right. We call these bends *vertex bends*, because they have nothing to do with the topological structure of the graph, but they are necessary only because of the vertex degree. For illustration see Fig. 4a).

For the next considerations we assume that there is only one middle edge in each direction. We consider two models:
a) The general *podevs* and
b) the *podevsnef*, where we demand every face to have a non-zero area.
At first we discuss the difference between the two models; with *k-face* we denote a face f with deg(f) = k.

Lemma 1 *[7] Every k-face with $k \geq 4$ can be represented by a non-empty region.*

So the significant difference between our models is to allow or to forbid triangles with zero area. Our algorithm for a podevsnef is based on the following

Lemma 2 *[7] Every 0°-angle of a podevsnef has a unique corresponding 270°-bend.*

Using these observations we describe an algorithm to compute a podevsnef: A flow on an arc of type g) from $f \in U_F$ to $v \in U_V$ is allowed if and only if there is a flow on an arc of type e) from some $g \in U_F$ to f where g is one of the two faces on the other sides of the edges e_1 and e_2 which define the face f at the vertex v (this expresses the correspondence between the 0°-angle and the 270°-bend). We model this situation by replacing the arcs of type g) by arcs of type h) going directly from face g to node v: For every edge e being incident with vertex v and neighboring faces f and g. there are arcs of type h) from f to v and from g to v. Arc (g, v) stands for the combination of arc (g, f) of type e) and arc (f, v) of type g). Thus type h) arcs have capacity 1 and cost 1. Note that there are two arcs of type h) from a face f to a node v (v belonging to f): They correspond to the two type e) arcs belonging to the two edges of f incident to v.

After having solved the Min-Cost-Flow-problem we re-insert the arcs (f, v) and (g, f) instead of (g, v) (with the corresponding flow values) and compute the nearly orthogonal representation as described in section 2.

Not every feasible flow in the network described so far has a corresponding drawing: Let e_i and e_j be two consecutive edges in the adjacency list of some vertex v and f_i, f_j and f_k the resulting faces such that f_j lies between e_i and e_j (cf. Fig. 4b)).

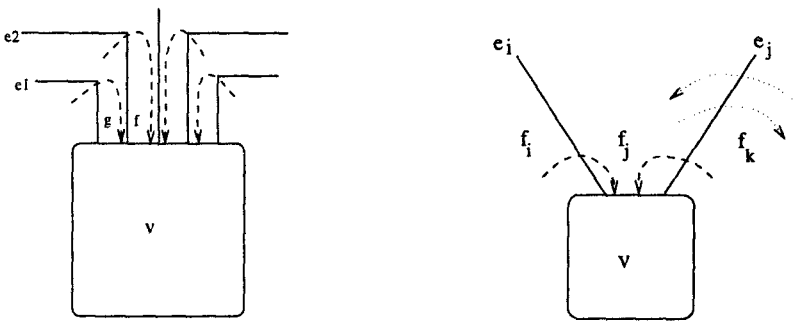


Fig. 4. a) Vertex Bends b) Forbidden Combinations

We have to avoid two cases:

(a) Since there are no negative angles, the flow from a face into a node must be restricted to 1: it is not allowed that the flows on the arcs of type h) from f_i to v over e_i and from f_k to v over edge e_j are simultaneously equal to 1 (dashed lines in Fig. 4b)).

(b) Further, since all edges at the left of the middle edge(s) must have their vertex bend to the left and all edges at the right of the middle edge(s) must have their vertex bend to the right, it is forbidden to have a flow of value 1 from f_i to v over edge e_i and simultaneously a flow of value 1 from f_j to v over the same edge e_i (see dotted lines in Fig. 4b)).

Although it is easy to avoid any of the forbidden cases by usual means of flow problems, we cannot guarantee both conditions simultaneously. So we realize one of the conditions (we choose condition (a)) using the capacity restrictions of the flow problem and the other one by punishing it with extraordenarily high cost; thus we establish the arcs of type h) not directly, but use the following construction: Let v be a node in U_V and $f_{i_1}, \ldots, f_{i_k}$ an ordered list of the faces around the vertex v in the graph (e.g. in clockwise order); let $e_{i_1}, \ldots, e_{i_k}$ be the edges that separate these faces such that e_{i_j} separates face $f_{i_{j-1 \bmod k}}$ and face f_{i_j}. See Fig. 5 for illustration (for $k = 3$).

Then we add for every edge e_{i_j} being incident to v two nodes $H^l_{e_{i_j}}$ and $H^r_{e_{i_j}}$, where $H^l_{e_{i_j}}$ ($H^r_{e_{i_j}}$) corresponds to the arc of type h) crossing edge e_{i_j} in clockwise (counterclockwise) order around v; further $H_{f_{i_j}}$ are new nodes in the network for every face f_{i_j}.

New arcs are:

- Arcs with capacity 1 and cost $2c+1$ (c having a suitable value) from f_{i_j} to $H^r_{e_{i_j}}$ and to $H^l_{e_{i_{j+1 \bmod k}}}$, i.e. to the edges of the graph corresponding to the arcs of

type h) starting in face f_{i_j} and crossing a bounding edge of this face.

- Arcs with capacity 1 and cost 0 from the nodes $H_{f_{i_j}}$ to node v; the arcs of these two types replace the arcs of type h).

- Arcs with capacity 1 and cost 0 from node $H^l_{e_{i_j}}$ to node $H_{f_{i_j}}$ and from node $H^r_{e_{i_{j+1 \bmod k}}}$ to node $H_{f_{i_j}}$, i.e. from two auxiliary nodes for two neighboured edges to the auxiliary node for the face between them. These arcs guarantee condition (a).

- Arcs with capacity 1 and cost $-c$ from node $H^l_{e_{i_j}}$ to node $H^r_{e_{i_j}}$ and vice versa. Every pair of such arcs defines a cycle of cost $-2c$ and thus a cost minimum path from a node f_{i_j} to a node v has cost $2c + 1 - 2c = 1$ corresponding to one necessary bend as in the case at the arcs of type h). Every time a second flow unit passes one of the nodes $H^l_{e_{i_j}}$ or $H^r_{e_{i_j}}$, the arcs with negative cost are already satisfied and thus the path from f_{i_j} to v has cost $2c + 1$.

Fig. 5 shows the construction of this part of the network. All capacities are 1 and all costs not indicated are 0.

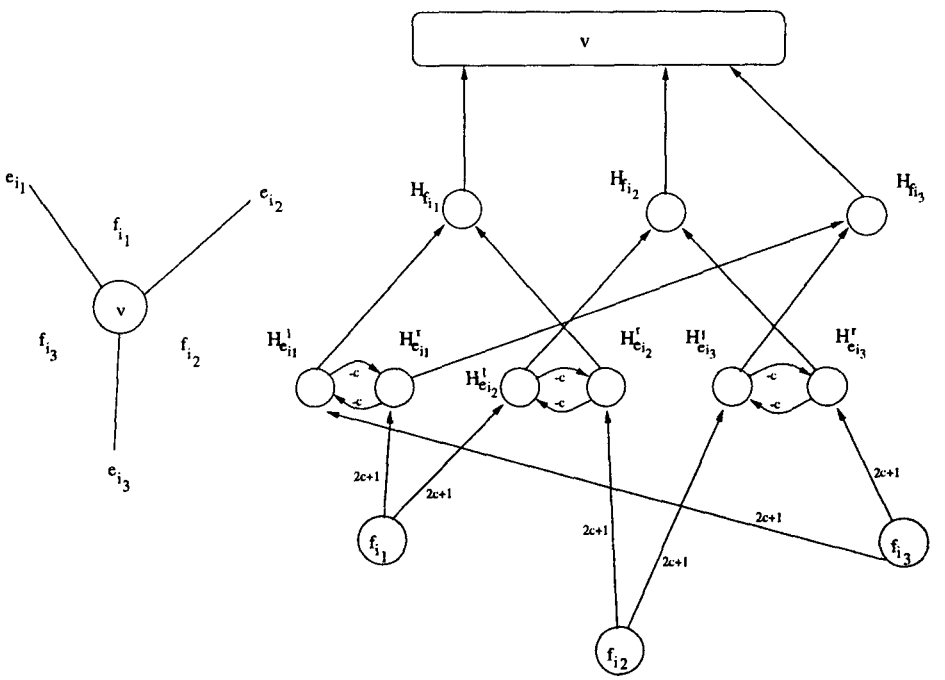


Fig. 5. The construction for the arcs of type h)

Now we can formulate the algorithm that computes the nearly orthogonal representation:

Algorithm

(1) Establish the network as described above;
(2) Solve the Min-Cost-Flow problem;
(3) Re-insert the arcs of type h) instead of the auxiliary construction and replace them by arcs of type e) and g);
(4) Compute the bends and angles of the drawing using the value of the flow on arcs of type d), e) and g).

Note that we have to solve a Min-Cost-Flow problem in a network with negative cycles. But that does not cause any troubles here, since all these cycles are known and have length 2; so we can use a standard augmentation algorithm where the cost-minimum paths are determined by (a slightly modified version of) Dijkstra's algorithm.
Unfortunately we do not have a 1:1-correspondence between a feasible max-flow in the network and a feasible drawing; but if we choose c large enough, we can state the following

Lemma 3 *For every feasible max-flow in the network with cost $b < c$ there is a corresponding drawing with exactly b bends.*

Proof: Taking into account the considerations about vertex bends (see Fig. 4a)) the lemma can be proved by the same arguments as those applied in [16]. The only thing that remains to be shown is that a feasible max-flow of cost smaller than c always exists. It suffices to show that there is a drawing with $b < c$ bends, since in this case it is easy to construct a flow with cost b. In [7] an algorithm is described which computes a drawing with at most $2m \leq 6n$ bends, so with $c = 6n$ the Lemma is correct. $\Diamond$

Lemma 4 *The algorithm above computes a nearly orthogonal representation in time $O(n^2 \log n)$.*

Proof: For every vertex and for every face of the graph there is a node in the network; further we have auxiliary nodes: Two for every edge and one for every face; thus planarity of the graph implies linearity of the number of nodes in the network. The number of arcs of type a), b), c) and f) is proportional to the number of vertices of the graph, the number of arcs of type d), e), g) and h) is proportional to the number of edges. So the network has a linear number of nodes and arcs and the Min-Cost-Flow problem can be solved in time $O(n^2 \log n)$. The rest of the algorithm runs in linear time. $\Diamond$

4 Compaction

To get a drawing from the nearly orthogonal representation, we have to assign lengths to the edge segments in a consistent way; let s be the size of the vertices, i.e. the length of the square side. We want to place the centers of the squares on a grid with unit distance $\lambda = 2s$; then the distance between two vertices is at least as large as

the vertex itself. So we have to guarantee that the edge lengths are concurring with this constraint, in particular that the length of every straight edge is a multiple of s. Tamassia [16] describes a linear time algorithm that solves the corresponding problem for a vertex degree of at most 4, and we want to use a similar technique. We replace every vertex v by a square of $8s$ small vertices (see Fig. 6), where the small vertex in the middle of each side (marked with an m in Fig. 6) will be incident to the (or one of the) middle edge(s) in this direction; there are enough small vertices for the rest of the edges being incident to v.

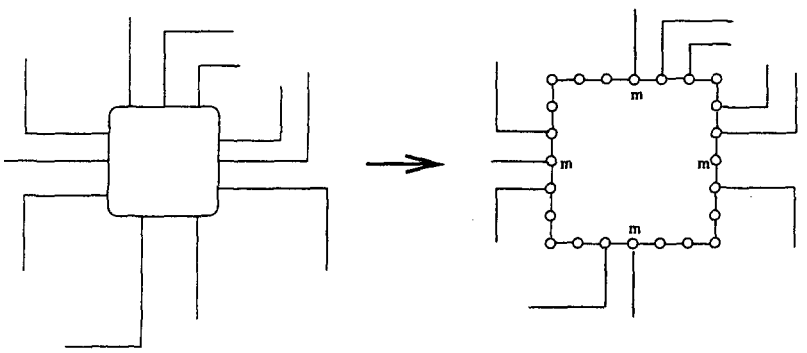


Fig. 6. Replacing a Large Vertex by Small Vertices.

Note that a square of $4s$ small vertices would not be enough: Let e.g. s be equal to 3 and $e = (u, v)$ be a vertical middle edge such that e is the rightmost edge at the bottom side of u and the leftmost edge at the top side of v. Then it would be impossible to draw e without any bend, if the centers of the two squares representing u and v should have the same x-coordinate.
So the size of the vertices is $2s \times 2s$ instead of a possible size of $s \times s$; but in the latter case we would have drawings with more bends, because we could not guarantee that we can draw middle edges without bends.
For the resulting graph (after the construction described here) we run a variant of the compaction algorithm of [16] to compute coordinates for the vertices and thus the final drawing.

Summarizing the results of the sections 3 and 4 we formulate

Theorem 1 *Our algorithm together with the compaction algorithm computes a podevsnef with the minimum number of bends in $O(n^2 \log n)$ time.*

5 Concluding Remarks and Discussion

This work has been motivated by the two figures at the end of the paper taken from the doctoral dissertation of Petra Mutzel [11]. She has convinced us to work on this specific model: On the left side the original picture from the paper on Astrophysics,

on the right side the hand-made layout after the maximum planarization step. The third picture shows a bend-minimum podevsnef produced by an implementation of our Algorithm using GraphEd [9] (the non-planar edge between the vertices HCO^+ and CH was deleted by hand since the algorithm can only handle planar graphs). Another interesting example drawn as podevsnef and shown at the end is the planar graph from the competition in last year's GD-conference.

Our algorithm works well for podevs restricted to have non-empty faces. Trying to omit this restriction we saw that the case of empty triangles is the only problematic case. We are currently working on an NP-hardness proof for the general problem. Running several examples we get the impression that a preference of those (computably hard) configurations lead to clear and understandable drawings, since the edges are bundled and clearly separated. Possible heuristics to achieve such empty faces are to eventually flip corners, or more drastically to change the network flow problem if an empty triangle can be achieved by a local transformation.

A promising approach which is attractive by other criteria is the simple one described in section 2. Here we achieve some visibility drawings where the edges are either horizontal or vertical lines. The standard efficient algorithms produce visibility drawings as shown in figure 1a). By only a slight modification of the corresponding network we can balance the number of horizontal and vertical edges and get a much better drawing instead (see Fig. 7).

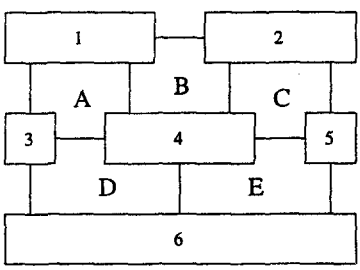


Fig. 7. A more attractive 'visibility representation'

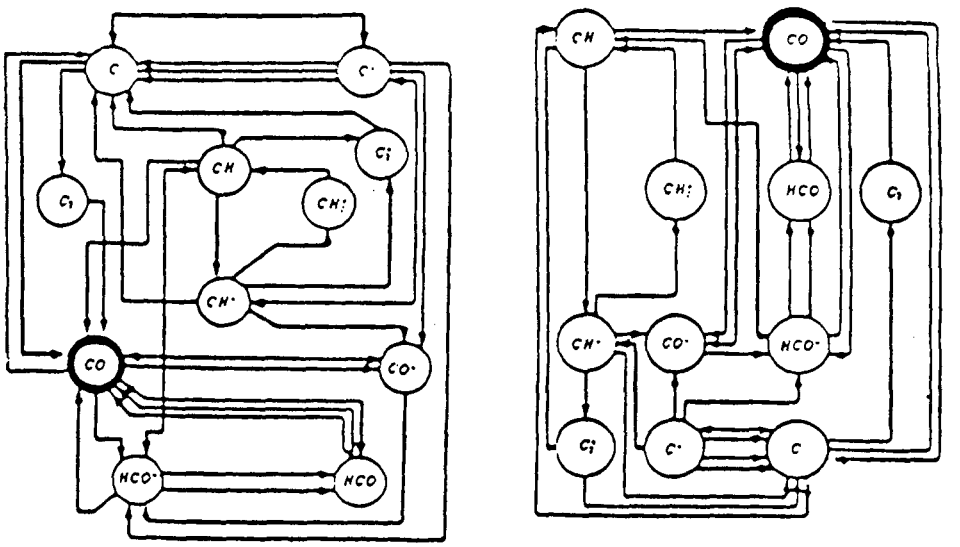
Aspects for further research are to improve the used area by local transformations maintaining the minimum bend number and to give bounds for the used area. Another is to allow different vertex sizes not depending on the graph structure, but depending on the size of some text labels to be written inside.

Acknowledgement: We wish to thank Harald Lauer, Petra Mutzel and Roberto Tamassia for helpful discussions and comments on earlier versions of this paper.

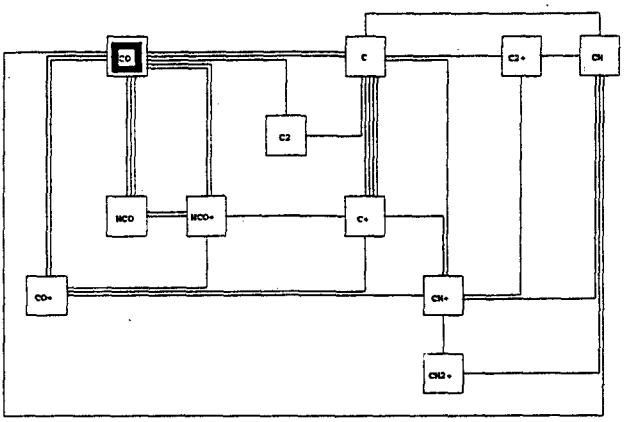
References

1. Batini, C., E. Nardelli, and R. Tamassia, *A Layout Algorithm for Data-Flow Diagrams*, IEEE Trans. on Software Engineering, Vol. SE-12 (4), pp. 538-546, 1986.
2. Batini, C., M. Talamo, and R. Tamassia, *Computer Aided Layout Of Entity-Relationship Diagrams*, The Journal of Systems and Software, Vol. 4, pp. 163-173, 1984.

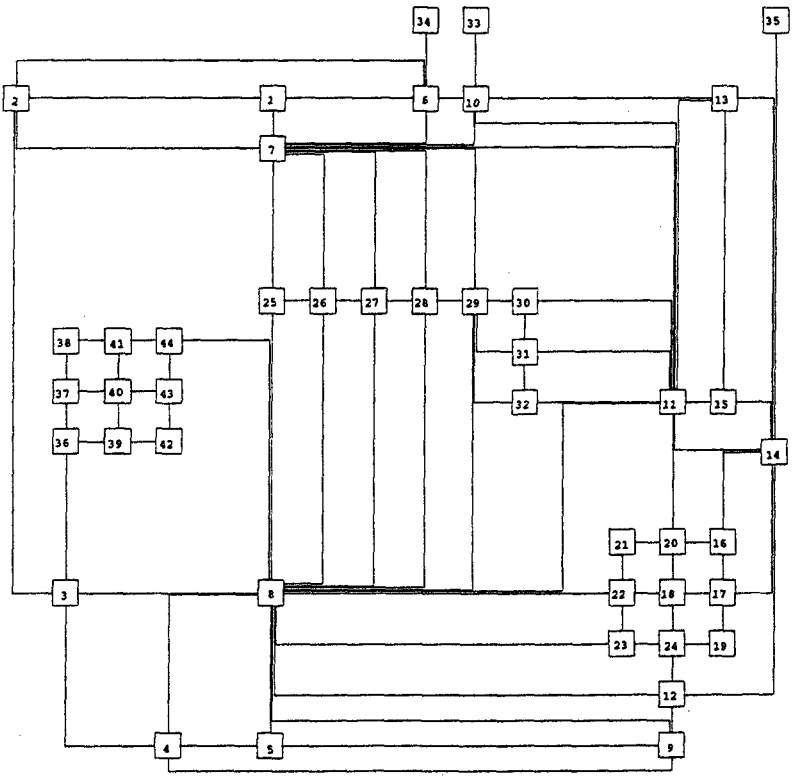
3. H.K.B. Beck, H.-P. Galil, R. Henkel, and E. Sedlmayr: *Chemistry in circumstellar shells, I. Chromospheric radiation fields and dust formation in optimcally thin shells of M-giants*, Astron. Astrophys. 265 (1992) 626–642.
4. Di Battista G., P. Eades, R. Tamassia and I.G. Tollis, *Algorithms for Automatic Graph Drawing: An Annotated Bibliography*, Tech.Rep., Dept.of Comp.Sc., Brown Univ., 1993.
5. Di Battista, E. Pietrosanti, R. Tamassia and I.G. Tollis, *Automatic Layout of PERT Diagrams with XPERT*, Proc. IEEE Workshop on Visual Lang. (VL'89), 171-176, 1989.
6. Di Battista, G., L. Vismara, *Angles of Planar Triangular Graphs*, Proc. of the 25th ACM Symposium on the Theory of Computing, San Diego, California, 1993.
7. Fößmeier, U., and M. Kaufmann, *Drawing High Degree Graphs with Low Bend Numbers*, Technical Report WSI-95-21, Univ. Tübingen 1995.
8. Garg, A. and R. Tamassia, *On the Computational Complexity of Upward and Rectilinear Planarity Testing*, Proc. of GD '94, Princeton, 1994.
9. Himsolt, M., *Konzeption und Implementierung von Graphneditoren*, Doctoral Dissertation, Passau 1993.
10. Lengauer, Th., *Combinatorial Algorithms for Integrated Circuit Layout*, Teubner/Wiley & Sons, Stuttgart/Chichester, 1990.
11. Mutzel, P., *The Maximum Planar Subgraph Problem*, Doctoral Dissertation, Köln 1994.
12. Reiner, D., et al., *A Database Designer's Workbench* in Entity-Relationship Approach, ed. S. Spaccapietra, pp. 347-360, North-Holland, 1987.
13. Rosenstiehl, P., and R.E. Tarjan, Rectilinear planar layouts and bipolar orientations of planar graphs, *Discrete and Comp. Geometry* 1 (1986), pp. 343–353.
14. Protsko, L.B., P.G. Sorenson, J.P. Tremblay, and D.A. Schaefer, *Towards the Automatic Generation of Software Diagrams*, IEEE Trans. on Software Engineering, Vol. SE-17 (1), pp. 10-21, 1991.
15. Storer, J.A., *The node cost measure for embedding graphs in the planar grid*, Proc. 12th ACM Symposium on the Theory of Computing, 1980, pp. 201-210.
16. Tamassia, R., *On Embedding a Graph in the Grid with the Minimum Number of Bends*, SIAM Journal of Computing, vol. 16, no. 3, 421 - 444, 1987.
17. Tamassia, R., and I.G. Tollis, *A unified approach to visibility representations of planar graphs*, Discr. and Comp. Geometry 1 (1986), pp. 321–341.



The layouts for the astrophysics-example from [11]



The podevsnef for the astrophysics-example



The podevsnef for the planar competition graph from GD'94