

Non-Planar Orthogonal Drawings with Fixed Topology

Extended Abstract

Markus Chimani, Gunnar W. Klau, and René Weiskircher

Institute of Computer Graphics and Algorithms,
Vienna University of Technology, Austria
{mch|gunnar|weiskircher}@ads.tuwien.ac.at

Abstract. This paper discusses the calculation of bend minimal shapes for non-planar graphs with given topology. Based on the Simple-Kandinsky drawing standard – a simplification of the more complex Kandinsky standard – we show the disadvantage of using standard models for this task: We show that the minimal bend count is suboptimal, when these models are applied to non-planar graphs; it is therefore beneficial to extend these standards.

We define such an extension for Simple-Kandinsky called SKANPAG (Simple-Kandinsky for Non-Planar Graphs). It treats edge crossings in a special way by letting them share identical grid points where appropriate. Hence it allows crossings of whole bundles of edges instead of single edges only. Besides having a reduced number of bends, drawings following this standard are easier to read and consume less area than those produced by the traditional approaches.

In this paper, we show a sharp upper bound of the bend count, if the standard Simple-Kandinsky model is used to calculate shapes for non-planar graphs. Furthermore, we present an algorithm that computes provably bend-minimal drawings in the SKANPAG standard.

1 Introduction

We consider the problem of producing an orthogonal drawing of a graph in the plane. The three-phased *topology-shape-metrics* approach [1, 3, 14] breaks the drawing problem into three subproblems:

1. *Topology/Planarization:* The first phase calculates a topology of the given graph. If the graph is non-planar, this includes augmenting the graph with *dummy-nodes* which represent edge crossings. The main objective is to produce as few crossings as possible.
2. *Shape/Orthogonalization:* The second phase calculates an orthogonal shape for the given topology. Such a shape defines the bends on the edges, and the angles between adjacent edges. The main objective is to produce as few bends as possible.

3. *Metrics/Compaction*: The third phase calculates the final dimensions for the given shape, by assigning lengths to the edge segments. The main objective is to minimize the size of the resulting drawing.

In this paper we focus on the second phase of the approach, and propose an extension of the well-known Simple-Kandinsky drawing model, which is also known as Simple-Podevsnef [4]. Note that Simple-Kandinsky is a simplification of the more complex Kandinsky/Podevsnef standard (**P**lanar **o**rtogonal **d**rawing with **e**qual **v**ertex sizes and **n**on-**e**mpy **f**aces) [10]. All of the above models define orthogonal drawings with equal node size where multiple edges can be attached to a single side of a node. In contrast to Kandinsky, Simple-Kandinsky has certain restrictions on how these bundles split up. We give a brief introduction to the Simple-Kandinsky model and its corresponding algorithm, which is based on a min-cost-flow network, in Section 2.

Our extension, which is called SKANPAG (**S**imple-**K**andinsky for **N**on-**P**lanar **G**raphs), is discussed in Section 3. It allows the dummy-nodes introduced in the planarization phase to share positions on the drawing grid under certain conditions. In general, this leads to a reduction of the number of bends, as well as to a smaller area required by the drawing. Furthermore, it increases the overall readability of the resulting drawing.

We show an upper bound of the bend count for the use of classic Simple-Kandinsky on non-planar graphs in Section 4. In Section 5 we present an algorithm that generates a bend-minimal SKANPAG-compliant shape (for a given topology). To our knowledge, this algorithm is the first that can draw non-planar graphs with the minimum number of bends in the Simple-Kandinsky model. Although our method is based on an integer linear program (ILP), its running time is very low in practice, even for large and dense graphs – drawing, e. g, the K_{25} takes under 50 seconds. This is discussed in detail in Section 6. We conclude with Section 7 where we also present our ideas for further research in this field.

Since most of the proofs are quite technical and long, we only outline them in this extended abstract. Details can be found in [5, 6].

A related approach has been followed by Fößmeier and Kaufmann in [9] for the Kandinsky drawing standard. However, their method relies on the incorrect assumption that all LPs with integer coefficients would result in integer solution vectors and is beyond remedy [5].

2 The Simple-Kandinsky Model

The first polynomial approach to generate bend-minimal shapes was given by Tamassia [13], and is based on transforming the problem into a min-cost-flow network. This original method is restricted to graphs with maximum degree four.

Several attempts have been made to extend the method to the larger class of planar graphs with arbitrary node degrees. We focus on a model where nodes are drawn with uniform size. Fößmeier and Kaufmann have introduced the *Kandinsky* drawing standard [10]. By using a fine grid for the edges and a coarser grid

for the nodes, it is possible that several edges emanate from the same side of a node, forming 0° angles between them. Furthermore, the standard does not allow faces with angle sum 0° . The authors present a minimum-cost flow approach with additional constraints on the flow that can be realized, for instance, using an ILP approach as in, e. g., [8] or in the AGD library [11].

Recently, Bertolazzi, Di Battista, and Didimo have proposed a simplification of this standard – which we will refer to as the *Simple-Kandinsky standard* – by adding the following restrictions [4]:

- (S1) Each node with degree ≥ 4 has at least one edge emanating from each of its four sides. No node with degree ≤ 4 is allowed to have 0° angles.
- (S2) For each two neighboring edges that leave a node on the same side, the first bend on the rightmost edge is a right bend.

The authors present a polynomial-time algorithm based on a network flow model that computes bend-minimal drawings of planar graphs with fixed topology in this standard. Here, we present a slightly modified but equivalent flow model for this task. Although larger by a constant factor, it is favorable for our purposes due to its simplicity, consistency, and straightforward extendibility.

Let $G = (V, E)$ be a planar graph with given topology, characterized by the face set F with outer face f_o . We create an underlying min-cost-flow network with two different types of directed edges: (a) arcs from the nodes $v \in V$ to incident faces $f \in F$ with zero cost, and (b) arcs with unit cost between adjacent faces.

Each unit transported in this network corresponds to a 90° angle: A flow of x units on an arc between a node $v' \in V$ and a face $f' \in F$ implies an angle of $x \cdot 90^\circ$ on f' at v' . Each flow unit on an arc from the face f' to $f'' (\in F)$ introduces a 90° bend on the edge $e' \in E$ that is on the border of both faces.

The capacities of the arcs as well as the supplies and demands of the nodes in the network are straightforward, and similar to the original Tamassia model: Each node $v \in V$ has a supply of four units which corresponds to 360° ; each face $f \in F - \{f_o\}$ has a demand of $2 \deg(f) - 4$, while the outer face f_o has a demand of $2 \deg(f_o) + 4$, where $\deg(\cdot)$ denotes the number of edges bounding a face.

To satisfy constraints (S1) and (S2) – and therefore guarantee valid drawings – we have to apply an augmentation to the network for each high-degree node in V (see Fig. 1): We add a cyclic substructure of demand-free nodes. Each of these nodes is the target of an arc of type (b) which causes right bends and an arc of type (a). Furthermore, we insert arcs between the new nodes and the faces surrounding the high-degree node; these arcs have a lower flow bound of one unit. The construction guarantees that if two edges leave a node on the same side, forming a 0° angle, the right edge of this bundle has to have a right bend.

3 Theory of Hyper-Faces – The Skanpag Model

The key to bend-minimality for non-planar graphs is to allow certain dummy-nodes to share a grid point, see Fig. 2(a). Nodes that share a grid point are said to be *merged*. Faces that become empty by such a merge, are said to be *collapsed*.

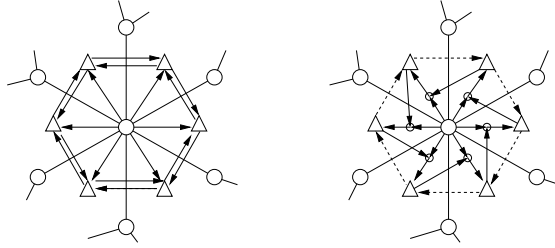
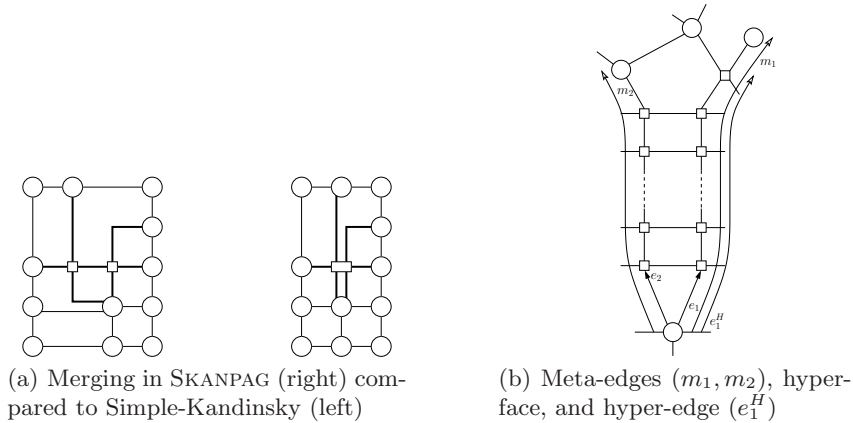


Fig. 1. Network construction (left) and correctly augmented network (right); (circles and triangles represent nodes and faces, resp., dashed arcs are unchanged by the augmentation, irrelevant arcs are not shown)

If not stated otherwise, let $G_o = (V_o, E_o)$ be a non-planar, simple, and self-loop-free graph. The graph $G = (V, E)$ denotes the planarization of G_o (based on a topology T_o). Let T be the planar topology of G that the drawing should be based on, and F the face set implied by T . We have $V = V_o \cup D$, where D is the set of the dummy-nodes introduced by the planarization.

Definition 1. If an edge $e_o \in E_o$ is split up into several sub-edges $e_i \in E$ during the planarization, we call the set $\{e_i\}$ the meta-edge of any of these e_i .

To extend Simple-Kandinsky for non-planar graphs in the most general way, we have to demand its properties only for the underlying non-planar graph G_o , not for the planarized graph G . Hence we demand a right bend on bundles for the meta-edges instead of only for their first sub-edges.



(a) Merging in SKANPAG (right) compared to Simple-Kandinsky (left)

(b) Meta-edges (m_1, m_2), hyper-face, and hyper-edge (e_1^H)

Fig. 2. Definitions. Original nodes are circles, dummy-nodes are squares

This leads to the analysis of bundles of meta-edges, and when their dummy-nodes are allowed to merge. Two meta-edges can only (partially) merge, if they

emit from a common source node $s \in V_o$ with $\deg(s) > 4$ (see property (S1)). Therefore we can deduce the two different structures of faces in F which might collapse:

- *collapsible triangle (“coltri”)*: A face that has exactly one incident node with a degree greater than four (the source node s of a bundle), and two incident dummy nodes.
- *collapsible quad (“colquad”)*: A face that has exactly four incident dummy nodes.

This leads to the concept of *hyper-faces* (see Fig. 2(b)):

Definition 2. Let e_1 and e_2 be the edges incident to a coltri f_0 and its high degree node s . A hyper-face f_0^H is a sequence of faces that starts with f_0 , contains any number of colquads, and ends with a face other than a colquad. All its faces have to be incident to a sub-edge of the meta-edges of e_1 and to a sub-edge of the meta-edge of e_2 .

Note that we only consider simple graphs; hence a hyper-face end-face can never be a coltri.

We can merge a pair of dummy-nodes that lie on the border of such a hyper-face, see Fig. 2(a), as long as there are no bends which cause a split up (“*demerge*”).

If not stated otherwise, we will give directions on the hyper-face in its *natural orientation*, where the coltri is at the bottom (as in Fig. 2(b)). We assume that m_1 (the meta-edge of e_1) is on the right side of the hyper-face. To satisfy the Simple-Kandinsky constraints, we have to assure that m_1 has at least one right bend before any left bend, and before m_2 (the meta-edge of e_2) has a right bend.

It is clear that such a right bend has to happen before we would be forced to merge a dummy-node with an original node. Hence we can specify the constraint that such a right bend has to happen on the *hyper-edge*:

Definition 3. The hyper-edge e_1^H is an ordered subset of the meta-edge m_1 and contains all the edges of the meta-edge that are on the boundary of sub-faces of the according hyper-face (see Fig. 2(b)). The analogously defined subset of m_2 is the partner edge of the hyper-edge.

We can summarize these observations: SKANPAG has to force a right bend on each hyper-edge, if its respective coltri has an opening angle of 0° . We can merge dummy-nodes if and only if their two meta-edges leave a node on the same side, and do not have any bends that would cause a split up of the bundle. Note that edges incident to a high degree node that have not been split up by the planarization step still have to satisfy property (S2) of standard Simple-Podevsnef instead.

4 Quality Guarantee of Simple-Kandinsky

Prior to SKANPAG, the only way to draw non-planar graphs was to use a planar drawing standard as a heuristics. We look at the special case of using Simple-Kandinsky, because of its strong relationship to SKANPAG. Furthermore, Simple-Kandinsky seems to be a good compromise between simplicity, execution time, and quality. Hence it is quite interesting to assess the quality difference between Simple-Kandinsky used as a heuristics and SKANPAG on non-planar graphs.

Note that for planar graphs, the minimum number of bends is equal for Simple-Kandinsky and for SKANPAG. Since there are no hyper-faces in planar graphs, there are no additional constraints, and hence no difference in the solution.

Also note that the minimal bend count in Simple-Kandinsky can never be lower than SKANPAG's: Each valid Simple-Kandinsky solution defines a valid SKANPAG solution with the same number of bends because Simple-Kandinsky's right-bend constraint is a specialization of the corresponding constraint for hyper-faces in SKANPAG.

Theorem 1. *For any given planarized graph $G = (V, E)$ and its planar topology T with h hyper-faces, Simple-Kandinsky requires at most h more bends than SKANPAG.*

Proof. We assume that a SKANPAG-algorithm has calculated a valid shape. The following observation holds true for every hyper-face h_i : If the opening angle of h_i is 90° or if there is a right bend on the first sub-edge of h_i , this shape is valid for Simple-Kandinsky, too.

If the coltri has an opening angle of 0° and there is no right bend on the first sub-edge of the hyper-edge of h_i , it is not a valid Simple-Kandinsky shape (Fig. 3(a), left). But we can achieve a related Simple-Kandinsky shape by increasing the opening angle from 0° to 90° and adding one right bend on the bundle partner of h_i (Fig. 3(a), right).

This transformation is not influenced by collapsed coltris to the left of h_i , since these have to be extended by analogous bends themselves to become Simple-Kandinsky compliant. Neither does the transformation influence its surrounding area, since its overall shape remains the same.

Hence we need at most one more bend for each hyper-face to transform a SKANPAG shape into a valid Simple-Kandinsky drawing. \square

Corollary 1. *The bound given in Theorem 1 is tight.*

Proof. Fig. 3(b) shows the graph used as a building block, containing exactly one hyper-face. We can put an arbitrary number of these blocks next to each other (joined by a simple edge). For every block, Simple-Kandinsky needs at least two bends, but SKANPAG requires only one. Hence Simple-Kandinsky requires exactly h bends more than SKANPAG. Note that each block is optimally planarized, since there exists no other planarization generating less than two dummy-nodes per block. \square

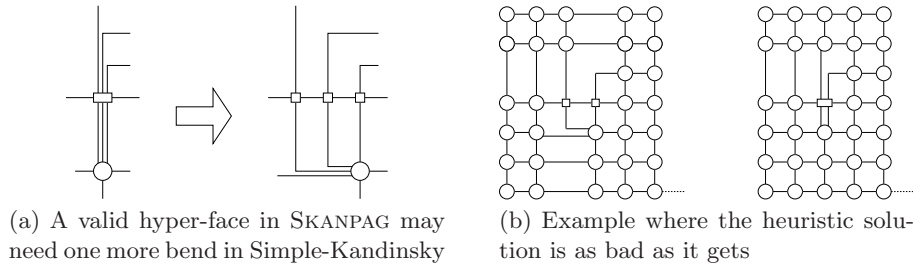


Fig. 3. Simple-Kandinsky as a heuristics

5 An Algorithm for Bend Minimal Skanpag-Drawings

Due to lack of space, we only outline the following algorithms. We implemented them as part of the AGD library [11] which is available freely for research purposes. Details, as well as the corresponding proofs, can be found in our technical paper [6] or in [5].

To solve the bend minimization problem, we use an ILP that models the underlying min-cost-flow network introduced in Section 2 and contains the right-bend-on-hyper-edge constraint.

While in Simple-Kandinsky the simple right-bend rule is enough to guarantee valid drawings, this is no longer true for SKANPAG. Hence we need an in-depth analysis of the situations that could render a correct drawing of hyper-faces impossible. Note that if all hyper-faces are drawn correctly, the complete graph will be drawn correctly (see the proof of Simple-Kandinsky’s validity in [4]).

If the coltri of a hyper-face has an opening angle of 90° , the drawing of the hyper-face is always valid (this follows from the correctness of Simple-Kandinsky). Hence we only have to analyze the case of a 0° opening angle.

By careful enumeration we can deduce two different types of situations that inhibit valid drawings of a hyper-face:

1. A flow unit that leaves a collapsed sub-face to its left side, and reenters over the same side. (Although this situation seems like a contradiction to the objective function, it occurs in bend-minimal drawings – both in SKANPAG and in Simple-Kandinsky – to generate necessary right bends)
2. A flow unit sent from a sub-face f_1 into its neighboring sub-face f_2 below it.
 - (a) The flow unit is not sent directly backwards from f_2 to f_1
 - (b) The flow unit is sent directly backwards from f_2 to f_1

We can show that we can prohibit all type 1 and type 2a errors by simple changes of the network structure, without cutting off the optimal solution. The errors of type 2b, however, are more complicated. They could, e. g., be avoided by additional constraints in the ILP, but these would introduce new 0/1-variables. Therefore we propose a different approach using a repair-function:

We solve the ILP without handling type 2b errors and thus generate an *almost valid* solution. This solution can then be repaired by a polynomial helper

function. The principle of this repair function is to analyze the remaining errors and move the corresponding flows to other positions. It ensures that all bend properties and constraints are still satisfied afterwards, but the formerly invalid flows do not generate errors anymore.

Our repair-function has an upper time bound of $O(h^2 l_{max}^2)$, where h is the number of hyper-faces and l_{max} the cardinality of the longest hyper-face. This bound can be estimated more generously as $O(|E|^2)$.

6 Computational Results

The *Rome graphs* [7] are a well established collection of 11529 graphs based on 112 graphs taken from real-world applications. They have between 10 and 100 nodes each, and 3280 graphs are planar. But even the non-planar graphs are quite sparse and have therefore very few dummy-nodes after the planarization step. Nearly 7000 graphs do not have any hyper-face at all and 1500 others contain only one. Thus we know from Theorem 1 that we cannot expect big differences between Simple-Kandinsky and SKANPAG. The implementation of our new method solves all these graphs without any need for the repair-function, nor does the LP-relaxation ever produce non-integer solutions.

Since this test suite is too sparse to show the difference between SKANPAG and Simple-Kandinsky, we *squared* each of the Rome graphs. The resulting graphs are still not extremely dense, but have more hyper-faces: only about 1200 still have none.

If not otherwise stated, the statistics compare SKANPAG to Simple-Kandinsky. As Fig 4(a) shows, we need nearly 10% less bends in the average case for big graphs, with peak values of up to 25%. The size of the drawing (Fig 4(b)) is reduced by over 20% on average. In some cases, we reduce the area consumption by 60%. As Fig 4(c) shows, the runtime performance of SKANPAG is acceptable even for large and dense graphs.

We also tested SKANPAG with complete graphs (up to K_{30}), to demonstrate the quality advantage for dense graphs. Note that the planarization of K_{30} has nearly 11000 dummy nodes. Fig 4(d) shows that we can save 15%-20% of bends for all such graphs with over 11 nodes; the area savings are even higher (up to 50%, see Fig. 4(e)). Figure 4(f) shows the runtime performance of SKANPAG and Simple-Kandinsky.

Figure 5 shows an example of a quite dense graph, drawn by Kandinsky, Simple-Kandinsky, and SKANPAG (equally scaled). More examples and details on the statistics can be found in [5].

7 Conclusion and Further Work

We have presented a new approach for drawing non-planar graphs that takes the special properties of dummy nodes into account. Our algorithm guarantees the minimum number of bends for any given topology following the Simple-Kandinsky properties, by the use of an integer linear program. It is the first

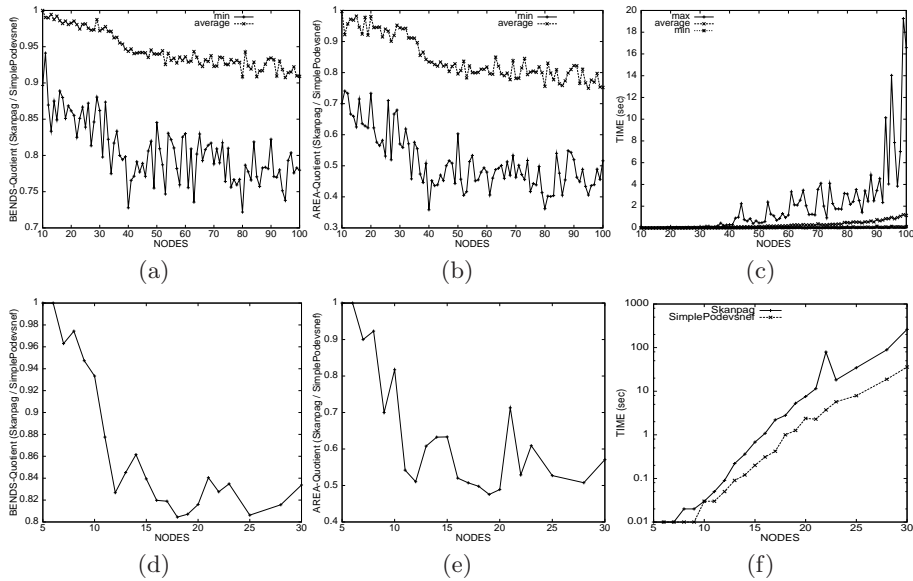


Fig. 4. Performance of SKANPAG, relative to Simple-Kandinsky. Top row: squared Rome graphs; bottom row: complete graphs. The peak at (f) is because the ILP-solver had to do several branches

approach to solve this problem and due to our polynomial time repair function, the runtime is acceptable even for large and dense graphs.

Note that our algorithm can also be used for drawing clustered graphs orthogonally. These are usually drawn by modeling the cluster boundaries as circles consisting of dummy-nodes and dummy-edges [2, 12]. By treating the dummy-nodes on the cluster boundaries just like any other dummy-node, we can achieve savings in bends and area compared to previously known algorithms.

The complexity of producing bend-minimal drawings in the SKANPAG model is still unknown. This – as well as the complexity proof for the Kandinsky model itself – is an interesting field of future study.

References

1. C. Batini, E. Nardelli, and R. Tamassia. A layout algorithm for data flow diagrams. *IEEE Trans. Softw. Eng.*, 12(4):538–546, 1986.
2. G. D. Battista, W. Didimo, and A. Marcandalli. Planarization of clustered graphs. In *Graph Drawing (Proc. GD 2001)*, volume 2265 of *LNCS*, pages 60–74, 2001.
3. G. D. Battista, P. Eades, R. Tamassia, and I. Tollis. *Graph Drawing*. Prentice Hall, 1998.
4. P. Bertolazzi, G. Di Battista, and W. Didimo. Computing orthogonal drawings with the minimum number of bends. *IEEE Transactions on Computers*, 49(8):826–840, 2000.

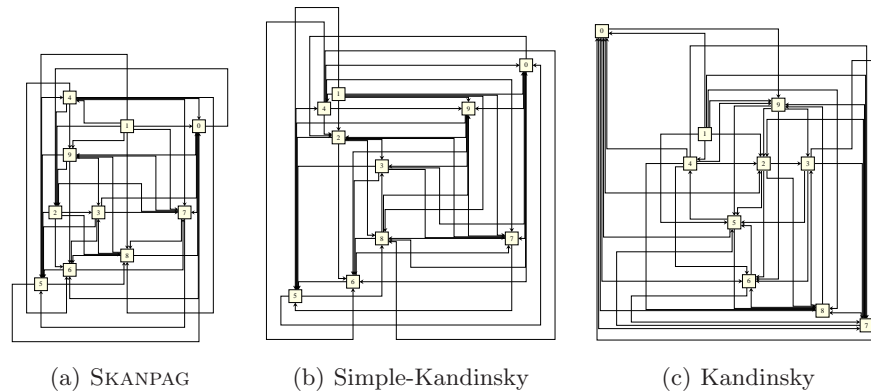


Fig. 5. Graph with 10 nodes and 42 edges, drawn using three different standards

5. M. Chimani. Bend-minimal orthogonal drawing of non-planar graphs. Master's thesis, Vienna University of Technology, Department of Computer Science, Austria, 2004.
6. M. Chimani, G. Klau, and R. Weiskircher. Non-planar orthogonal drawings with fixed topology. Technical Report TR 186 1 04 03, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2004.
7. G. Di Battista, A. Garg, and G. Liotta. An experimental comparison of three graph drawing algorithms (extended abstract). In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 306–315. ACM Press, 1995.
8. M. Eiglsperger, U. Fößmeier, and M. Kaufmann. Orthogonal graph drawing with constraints. In *10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1999)*, pages 3–11, 1999.
9. U. Fößmeier and M. Kaufmann. Algorithms and area bounds for nonplanar orthogonal drawings. In *Proc. 6th Symposium on Graph Drawing (GD'97)*, volume 1353 of *LNCS*, pages 134–145, 1997.
10. U. Fößmeier and M. Kaufmann. Drawing high degree graphs with low bend numbers. In F. J. Brandenburg, editor, *Proc. of the 3rd Int. Symp. on Graph Drawing (GD 1995)*, volume 1027 of *Lecture Notes in Computer Science*, pages 254–266, Passau, Germany, 1996. Springer.
11. M. Jünger, G. W. Klau, P. Mutzel, and R. Weiskircher. *Graph Drawing Software*, chapter AGD: A Library of Algorithms for Graph Drawing, pages 149–172. Mathematics and Visualization. Springer, 2003.
12. D. Lütke-Hüttmann. Knickminimales Zeichnen 4-planarer Clustergraphen. Master's thesis, Saarland University, Department of Computer Science, Saarbrücken, Germany, 2000.
13. R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987.
14. R. Tamassia, G. D. Battista, and C. Batini. Automatic graph drawing and readability of diagrams. *IEEE Trans. Syst. Man. Cybern.*, *SMC-18(1)*, 1988.