Sketch-Driven Orthogonal Graph Drawing*

Ulrik Brandes[1], Markus Eiglsperger[2], Michael Kaufmann[2], and Dorothea Wagner[1]

[1] Department of Computer & Information Science, University of Konstanz.
{Ulrik.Brandes|Dorothea.Wagner}@uni-konstanz.de
[2] Wilhelm Schickard Institute for Computer Science, University of Tübingen.
{eiglsper|mk}@informatik.uni-tuebingen.de

Abstract. We present an orthogonal graph drawing algorithm that uses a sketchy drawing of the graph as input. While the algorithm produces an orthogonal drawing with few bends in the Kandinsky model it also preserves the general appearance of the sketch. Potential applications for this kind of drawing algorithm include the generation of schematic maps from geographic networks and interactive orthogonal graph drawing.

1 Introduction

Orthogonal drawings of graphs are common in many areas, but especially so in technical and engineering applications where their apparent tidiness seems to be appreciated. Typical examples include circuit schematics, entity-relationship diagrams and system plans. A commonality in these applications is that, quite often, a sketch of the drawing is already available, either because a user has created the graph in an editor, or because the vertices have some natural, e.g. geographic, position. In such cases, standard black-box algorithms for graph layout do not leave sufficient control over the appearance of the drawing with the engineer.

We extend a successful approach for orthogonal graph layout to start from a sketch of the graph and produce a tidy, orthogonal drawing with few bends that preserves the overall appearance of the sketch. For the purpose of this paper, a "sketch" of a graph is a drawing that exhibits the main features to be conveyed, but is unsatisfactory from an aesthetic point of view.

In practice, a sketch may for instance be a manual layout produced by the user of a database schema editor or a physically realized network, and it may but does not have to be orthogonal. Our approach can thus be viewed as a more general case of dynamic graph drawing, since we do not require that the sketch is drawn by a layout algorithm or even in the same representation as the target drawing. An alternative view is that an existing drawing is to be improved subject to user-supplied constraints or hints for the algorithm.

There is a fair amount of research on dynamic graph layout (see [4] for an overview) and on utilizing user interaction for force-directed [22, 14], quasi-visibility [21] and layered layout [9], and while there are even attempts to learn

* Partially supported by DFG under grants Br 2158/1-1 and Ka 812/8-1.

parameters of layout objectives from example drawings [18, 19], we know but one instance of a method for explicit clean-up of a given sketch. SCHEMAP is a system generating schematic maps for communication networks that was demonstrated at last year's Graph Drawing Software Exhibition [17]. This system gradually orthogonalizes a given network layout ("ground plan") into a schematic map while preserving the embedding (including crossings). Since it is based on a force-directed algorithm there is no guarantee that the result is indeed orthogonal, and running times are apparently far from interactive.

We propose an efficient, dedicatedly orthogonal layout algorithm that extends the Kandinsky approach of [13] according to the Bayesian paradigm for dynamic layout of [2]. We therefore review briefly these two approaches in Sect. 2, before we describe our new algorithm in Sect. 3 and provide examples in Sect. 4. We conclude in Sect. 5.

2 Preliminaries

Our algorithm for sketch-driven orthogonal graph layout rests on two pillars: an orthogonal layout algorithm for general planar graphs, and a framework for extending (static) layout algorithms to dynamic graphs. We review briefly these foundations and refer to [8, 15] for background reading.

2.1 Orthogonal Graph Layout and Kandinsky Models

We assume that the reader is familiar with the concepts of planarity and network flow. An *embedded planar graph* $G(V, E, F)$ is a planar graph with a specific circular order of edges around vertices and a specific external face, admitting a planar drawing that respects the given embedding. Unless otherwise specified, the planar graphs we consider are always embedded.

A *planar orthogonal box drawing* of a planar graph is a planar drawing that maps each vertex to a box and each edge to a sequence of horizontal and vertical segments. We assume for the rest of this work *grid drawings*, i.e. drawings in which the center of the boxes and the bends along the edges have integer coordinates. An *orthogonal shape* Q is a mapping from the set of faces F of a graph G to clockwise ordered lists of tuples (e_i, a_i, b_i), $1 \leq i \leq |Q(f)|$, where e_i is an edge, $a_i \in \{1, \ldots, 4\}$ represents the angle formed with the following edge inside the appropriate face in multiples of $90°$, and b_i is the list of bends of the edge. A *quasi-orthogonal shape* is an orthogonal shape with $a_i = 0$ allowed, where a $0°$ angle means that the succeeding edge is adjacent to the same side of the vertex as the preceding edge. We denote with $Q(f, i)$ the i-th tuple of $Q(f)$, with $a(Q, f, i)$ the value of the angle field of $Q(f, i)$, and with $b(Q, f, i)$ the value of the bend field of $Q(f, i)$. A quasi-orthogonal shape Q is called *valid*, if there is a planar orthogonal box drawing with quasi-orthogonal shape Q. Note that quasi-orthogonal shape are not related to quasi-orthogonal drawings as described in [16].

Different drawing conventions have been proposed for planar orthogonal box drawings. We will concentrate on Kandinsky models. All Kandinsky models impose the following two constraints on the drawing which we call Kandinsky properties. The *bend-or-end property* is defined as follows: Let $G=(V,E)$ be an embedded graph and consider a planar orthogonal box drawing of G. Let e_1 and e_2 be two edges adjacent to the same side of a vertex v, e_1 following e_2 in the embedding, and let f be the face to which e_1 and e_2 are adjacent. Then either e_1 must have a last bend with a 270° angle in f or e_2 must have a first bend with 270° angle in f. The *non-empty face property* forbids some degenerated cases for triangles in the graph. See [13] for a detailed description of the Kandinsky properties. An important consequence is stated in the following lemma.

Lemma 1 ([13]). *Every* $0°$ *angle of a Kandinsky drawing has a unique corresponding* $270°$ *bend.*

Bends which correspond to a 0° angle are called *vertex-bends*, all other bends are called *face-bends*.

A Kandinsky drawing of a graph can be determined in the topology-shape-metrics framework [8, Ch. 2.3] by planarizing the graph (topology step), computing angles and bends (shape step), and finally determining the length of edge segments and size of vertex boxes (metrics step, also called *compaction*). Given an embedded planar graph, the shape of a Kandinsky drawing with the minimum number of bends is obtained from a minimum cost flow in a network that extends the well-known approach of [23] to account for 0° angles [13]. See Sect. 3 for details.

2.2 Dynamic Graph Layout and the Bayesian Paradigm

In dynamic graph drawing, the input is a sequence of graphs which represent the states of a single graph that is changing over time. Dynamic graphs can be visualized, e.g., in an animation or in a sequence of drawings, but it is important to keep changes between consecutive frames to a minimum in order not to destroy a viewer's mental map of the graph [10]. Methods for dynamic orthogonal layout are proposed in [20, 5, 3, 7].

The core modeling task of dynamic layout, i.e. the combination of criteria for good (static) layout with the requirement of small change, is therefore very similar to that of sketch-driven layout. For layout algorithms based on the optimization of an objective function, the *Bayesian framework* [2, 1] suggests to incorporate a *difference metric* [6] as a penalty in the objective function. Optimization of the combined objective function thus naturally results in a trade-off between static layout criteria and stability.

Since orthogonal drawing algorithms in the topology-shape-metrics framework heavily depend on the angles and bends computed in the shape step, it seems natural to use the change in angles and bends as a difference metric for orthogonal shapes [3, 1].

3 The Algorithm

Throughout this section, let Σ be an *admissible* drawing (a *sketch*) of a graph $G_\Sigma = (V_\Sigma, E_\Sigma)$, where a sketch is admissible if there is no overlap between edges and non-incident vertices, and no more than two edges cross in single point. Any sketch can be transformed into an admissible one. Our objective is to determine an orthogonal box drawing of G_Σ with the following properties:

- the topology is preserved,
- the drawing is in the Kandinsky model,
- angles in the drawing deviate little from angles in the sketch (stability), and
- the drawing contains few bends (readability).

Our algorithm follows the topology-shape-metrics approach and proceeds as follows. First, a planarization $G = (V, E, F)$ of G_Σ is determined. by replacing each crossing in the sketch by a dummy node, so that $V = V_\Sigma \cup C$ where C is the set of dummy nodes. Since we do not change the embedding of G in the following steps, this ensures that the topology of G_Σ is preserved. Next, a quasi-orthogonal shape of G is determined from Σ by rounding angles in the sketch to the nearest multiple of $90°$ and classifying each edge bend as either a 90^0 or a $270°$ bend. Note that the resulting quasi-orthogonal shape S need not be valid. A valid quasi-orthogonal shape Q in the Kandinsky model is then determined so as to satisfy a trade-off between stability and bend-number. Finally, a compaction algorithm is applied on Q to compute a Kandinsky drawing of G, from which the final drawing is obtained by replacing dummy nodes with edge crossings.

We concentrate on the problem of determining a quasi-orthogonal shape Q, since all other steps in the algorithm can be carried out using standard algorithms. The problem is stated in terms of an optimization problem for which we present a min-cost flow formulation based on [13].

3.1 Problem Statement

The problem of finding a quasi-orthogonal shape Q is a bi-criteria optimization problem where the two objectives are readability (number of bends) and stability (change in shape).

The readability of a shape Q is independent of the given sketch and defined as the total number of bends, namely

$$B(Q) = \frac{1}{2} \sum_{f \in F} \sum_{(e,a,b) \in Q(f)} |b| \ .$$

The stability of an orthogonal shape Q is expressed in terms of the difference between angles in Q and corresponding angles in the shape S of the sketch

$$\Delta_A(Q,S) = \sum_{f \in F} \sum_{1 \leq i \leq |f|} |a(S,f,i) - a(Q,f,i)|$$

and the difference in edge bends

$$\Delta_B(Q,S) = \sum_{f\in F}\sum_{1\le i\le |f|} \Delta(b(S,f,i), b(Q,f,i))$$

where $\Delta(s_1, s_2)$ denotes a restricted edit-distance between two strings, in which only insert and delete operations are permitted. This is similar to the shape difference metric used in [6]. From these components we form a weighted compromise between the degree of change with respect to the given sketch and the number of bends in the quasi-orthogonal shape. Our objective function thus reads

$$D(Q|S) = \underbrace{\alpha \cdot \Delta_A(Q,S) + \beta \cdot \Delta_B(Q,S)}_{\text{stability}} + \underbrace{\gamma \cdot (B(Q) - B(S))}_{\text{readability}}$$

where parameters α, β, and γ control the relative importance of angle or bend changes and bend number. We are now ready to state our problem formally.

Problem 1. Given a quasi-orthogonal shape S of a planar graph G, find a valid quasi-orthogonal shape Q of G in the Kandinsky model such that $D(Q|S)$ is minimum.

3.2 Bend-Minimum Kandinsky Shapes

Bend-minimum Kandinsky shapes for planar embedded graphs can be obtained by solving a path-based min-cost flow problem [13]. Since it is essential for our algorithm, we briefly review some important properties of the flow network, but assume that the reader is familiar with Tamassia's min-cost flow formulation for creating bend-minimum drawings of embedded planar graphs with maximum degree four [23]. This formulation contains a node for each vertex (*vertex-node*), and for each face (*face-node*) of the input graph.

The angle defined inside a face f between to consecutive edges a, b sharing a vertex v is determined by the flow on the network-edge that connects the vertex-node of v with the face-node of f. A flow of 0 defines an angle of 90°, a flow of 1 an angle of 180° and so forth. This is illustrated in Fig. 1. Since the sum

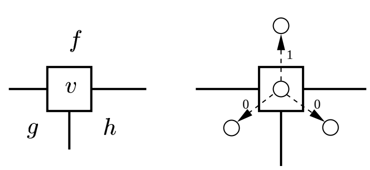


Fig. 1. A vertex v incident to faces f, g, and h, and the corresponding part of the min-cost flow network where arc labels indicate flow values according to the depicted shape

of angles around a vertex must be 360°, each vertex-node is connected to the source of the network with an edge of capacity $degree(v) - 4$.

Since zero flow represents an angle of 90°, an angle of 0° corresponds to negative flow which is not feasible in the network. In Kandinsky networks, this problem is solved as shown in Fig. 2 by introducing additional vertices and arcs which allow flow coming from a face-node to enter a vertex-node. Before a unit of flow can enter a vertex node, it crosses an edge and thus creates a bend. It follows that each 0° angle has the associated bend required by Lemma 1. For each vertex-node v we define the $supply(v)$ as $degree(v) - 4$. A vertex-node v with positive supply is connected to the network source with capacity $supply(v)$, and a vertex-node v with negative supply is connected to the network sink with capacity $-supply(v)$.

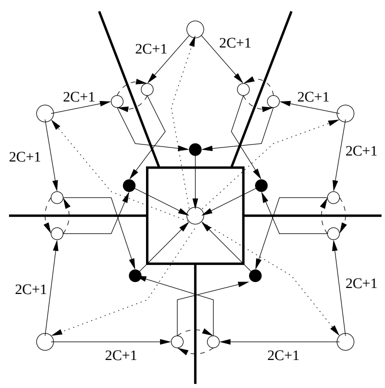
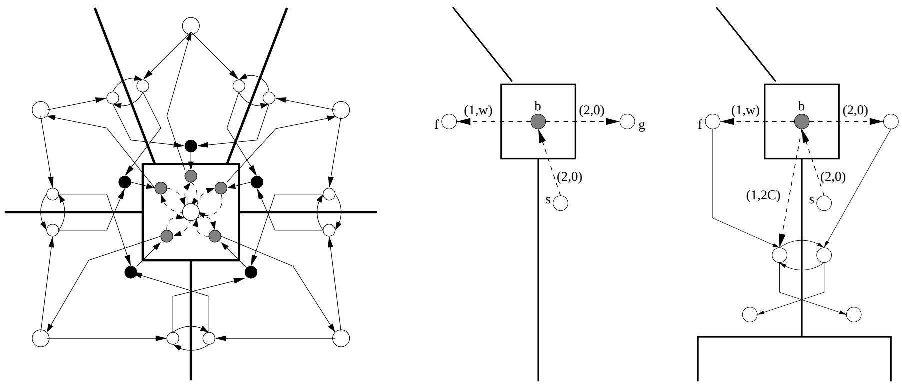


Fig. 2. Kandinsky flow network for a node adjacent to five edges. Thick lines depict the original node and edges, thin lines depict network nodes and edges. All capacities are one except at pointed arcs which have capacity three. If not stated explicitly, arcs have zero cost except for dashed circular arcs which have cost $-C$

3.3 Sketch-Driven Kandinsky Shapes

In this section we describe modifications to the Kandinsky network that yield a network for Problem 1.

The first modification concerns angles around a vertex v. We define for each angle defined by two consecutive edges e and e' on a vertex v an *angle-node* $a^v_{e,e'}$. We create two arcs, one from $a^v_{e,e'}$ to the vertex-node of v and one in the opposite direction. Both arcs have unconstrained capacity and cost α. We denote with $U_A(v)$ the set of angle-nodes of a vertex v. Angle-nodes are the only nodes connected to vertex nodes. All arcs which were connected to vertex-nodes in the original formulation are now connected to the corresponding angle-node. For each angle, and therefore for each angle-node w, there is a *target angle*



(a) around a vertex

(b) at regular and vertex-bends

Fig. 3. Modified Kandinsky network. Newly introduced arcs are dashed and labeled with capacity and cost in (b), where $w = \beta - \gamma$. Newly introduced nodes are marked gray. Node s is the global source

$a(w)$ defined by the shape S of the sketch. If $a(w) > 0$, then w is connected to the source with an arc of capacity $a(w)$ and cost zero. If $a(w) < 0$, then w is connected to the sink with an arc of capacity $a(w)$ and cost zero. Additionally we remove $a(w)$ from the supply of the vertex-node. Therefore the supply of a vertex node is now defined as $\mathit{supply}'(v) = \mathit{degree}(v) - 4 - \sum_{w \in U_A(v)} a(w)$. Our construction is illustrated in Fig. 3(a).

The second modification concerns the modeling of bends. For each bend in S we create the device shown left in Fig. 3(b). A demand of 1 is added to both faces incident to the edge containing the bend. The effect of this device is, that either Q contains the bend at zero cost or the bend is removed at cost $\beta - \gamma$. Additionally we assign arcs between faces the cost $\beta + \gamma$. Note that the first and the last bend on an edge are potential vertex-bends. For this type of bends we create an extra arc as shown right in Fig. 3(b). This arc ensures that vertex-bends can be confirmed at zero cost.

Lemma 2. *An optimal solution of the above path-based min-cost flow problem yields an optimal solution for Problem 1.*

Sketch of Proof. Observe that the original Kandinsky network is modified inside of boxes that represent vertices of the input graph. The total amount of supply in a box remains unchanged since $\mathit{supply}'(v) + \sum_{U_A(v)} a(w) = \mathit{degree}(v) - 4 = \mathit{supply}(v)$. The same holds for bend devices. Therefore the set of quasi-orthogonal shapes that correspond to a feasible flow in the network is the same as in the original formulation, namely the set of all valid quasi-orthogonal shapes. It remains to show that a min-cost flow in the network yields an optimal solution for $D(Q|S)$.

Let x be a feasible flow in the network, and Q the valid quasi-orthogonal shape corresponding to x. Consider the arc $a(v, w)$ between an angle-node w and its vertex-node v in the flow network and $a(w, v)$ the reverse edge. We assume that either $x(a(v, w)) = 0$ or $x(a(w, x)) = 0$. The angle-node w represents the angle between an edge e and its following edge in the embedding. Let (e, a_S, b_S) be a tuple in S and (e, a_Q, b_Q) be the corresponding tuple in Q describing this angle. Then $a_Q = a_S - x(a(v, w)) + x(a(w, v))$ and therefore $\alpha \cdot |a_S - a_Q| = \alpha \cdot (a(w, v)) + \alpha(a(v, w))$.

At bend devices there are two possibilities: either a bend is conserved at cost zero, or it is deleted at cost $\beta - \gamma$. A newly created bend has cost $\beta + \gamma$ in x. Let b_Q have minimum edit distance to b_S with k_D deletions and k_I insertions. Then $k_D(\beta - \gamma) + k_I(\beta + \gamma) = \beta(k_D + k_I) + \gamma(k_I - k_D) = \Delta(b_Q, b_S) + \gamma \cdot (|b_Q| - |b_S|)$. $\square$

Lemma 2 and [13] hence yield an efficient algorithm for an optimal trade-off between bend-reduction and sketch-preservation.

Theorem 1. *Problem 1 can be solved in time* $O(n^2 \log n)$.

4 Examples

While our approach works very well for graphs which are sufficiently connected, graphs with many tree like structures in the outer face are more difficult, because small angle changes result in major differences in appearance. We use an augmentation method in this case to improve the quality of the algorithm. The idea of the augmentation is to put a *frame* around the graph and connect vertices on the outer face to this frame by straight lines, see Fig. 4. We then execute the algorithm and finally remove the frame together with the augmentation edges. The introduced edges give the drawing a greater stability. We connect only vertices which are on the conex hull of the sketch drawing with the frame to avoid introducing new crossings.

An experimental version of our approach has been implemented in Java using yFiles [24]. Figure 5 shows the result of our algorithm applied to the example used in the paper that initiated this work [17]. In contrast to the force-directed approach used there, our network-flow approach has negligible running time and is therefore truly interactive.

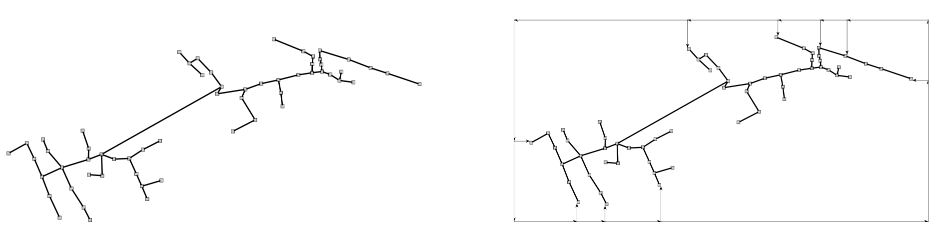


Fig. 4. Ground plan from [17] and its framed sketch.

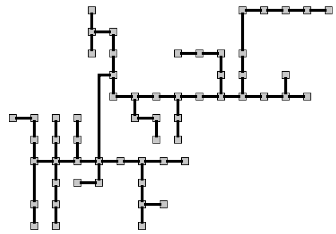


Fig. 5. Schematic map produced by our algorithm for ground plan from [17].

A second application that we envision is the beautification of ER diagram sketches of which the designer wants certain features to be preserved. An example is given in Fig. 6. Note that some degree one vertices such as the one labeled "Query" change their relative position in order to avoid vertex-bends, which would have been necessary if the angles had been fixed. This a good example for the trade-off optimization.

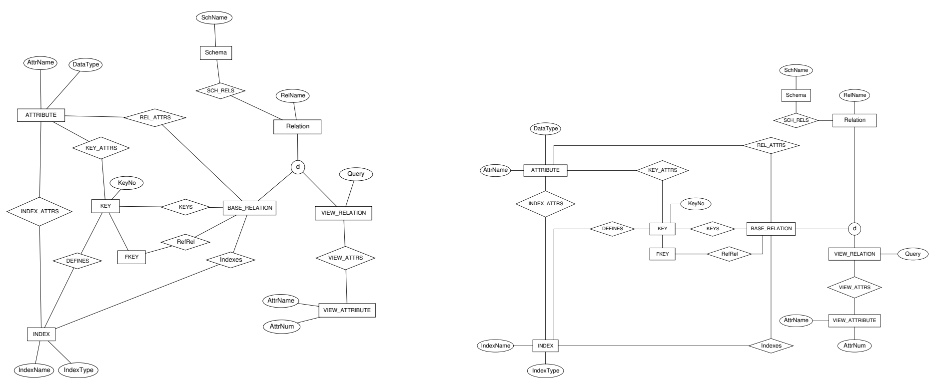


Fig. 6. ER diagram from [12, p. 577]) and an orthogonalized version

5 Discussion

Our approach can be extended to a fully dynamic orthogonal graph drawing algorithm by considering the most recent drawing as a sketch. Note that the structure of the graph may since have changed, i.e. nodes and vertices may have been added or removed. The deletion of graph elements poses no problem, since we can simply remove them from the sketch as well. Additions to the graph require more care, since, e.g., newly created edges split an angle in the sketch drawing into two, and we can not infer the target values for those two angles in the network. We do have, however, a target value for the sum of these angles.

Therefore merging the two angle-nodes into one and using the sum of the angles as the target value for the merged node solves this problem. See Fig. 7 for an illustration.

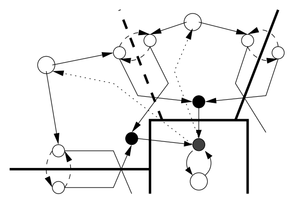


Fig. 7. Edge insertion for dynamic layout

It is worth noting that the globally defined parameters α, β, and γ can be adjusted locally for every angle and bend. This may be useful in dynamic graph drawing, e.g. to give older parts of a drawing increased stability.

An issue we have not addressed in this work is the compaction of sketch drawings. In our implementation we use the compaction algorithm described in [11] together with visibility postprocessing. However, sometimes it may be helpful to preserve some distances in the sketch drawing, especially in the dynamic scenario. Our future research will investigate this problem.

References

1. U. Brandes. *Layout of Graph Visualizations.* PhD thesis, University of Konstanz, 1999. `http://www.ub.uni-konstanz/kops/volltexte/1999/255/`.
2. U. Brandes and D. Wagner. A Bayesian paradigm for dynamic graph layout. Proc. Graph Drawing '97. Springer LNCS 1353:236–247, 1997.
3. U. Brandes and D. Wagner. Dynamic grid embedding with few bends and changes. Proc. Algorithms and Computation '98. Springer LNCS 1533:89–98, 1998.
4. J. Branke. Dynamic graph drawing. In *Drawing Graphs: Methods and Models*, Springer LNCS Tutorial 2025:228–246, 2001.
5. S. Bridgeman, J. Fanto, A. Garg, R. Tamassia, and L. Vismara. INTERACTIVE-GIOTTO: An algorithm for interactive orthogonal graph drawing. Proc. Graph Drawing '97. Springer LNCS 1353:303–308, 1997.
6. S. Bridgeman and R. Tamassia. Difference metrics for interactive orthogonal graph drawing algorithms. *Journal of Graph Algorithms and Applications*, 4(3):47–74, 2000.
7. M. Closson, S. Gartshore, J. Johansen, and S. Wismath. Fully dynamic 3-dimensional orthogonal graph drawing. *Journal of Graph Algorithms and Applications*, 5(2):1–35, 2001.
8. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall, 1999.
9. H. do Nascimento and P. Eades. User hints for directed graph drawing. Proc. Graph Drawing '01. Springer LNCS 2265:124–138, 2002.

10. P. Eades, W. Lai, K. Misue, and K. Sugiyama. Preserving the mental map of a diagram. *Proc. Compugraphics '91*, pp. 24–33, 1991.
11. M. Eiglsperger, and M. Kaufmann. Fast Compaction for Orthogonal Drawings with Vertices of Prescribed Size Proc. Graph Drawing '01. Springer LNCS 2265:124–138, 2002.
12. R. Elmasri and S. Navathe. *Fundamentals of Database Systems.* Addison-Wesley, 3rd ed., 2000.
13. U. Fößmeier and M. Kaufmann. Drawing high degree graphs with low bend numbers. Proc. Graph Drawing '95. Springer LNCS 1027:254–266, 1996.
14. J. Ignatowicz. Drawing force-directed graphs using Optigraph. Proc. Graph Drawing '95. Springer LNCS 1027:333–336, 1996.
15. M. Kaufmann and D. Wagner, editors. *Drawing Graphs: Methods and Models.* Springer LNCS Tutorial 2025, 2001.
16. G. W. Klau and P. Mutzel. Quasi-orthogonal drawing of planar graphs. TR 98-1-013, Max-Planck-Institut für Informatik, Saarbrücken, 1998.
17. U. Lauther and A. Stübinger. Generating schematic cable plans using springembedder methods. Proc. Graph Drawing '01. Springer LNCS 2265:465–466, 2002.
18. T. Masui. Graphic object layout with interactive genetic algorithms. *Proc. IEEE Visual Languages '92*, pp. 74–87, 1992.
19. X. Mendonça and P. Eades. Learning aesthetics for visualization. *Anais do XX Seminário Integrado de Software e Hardware*, pp. 76–88, 1993.
20. A. Papakostas and I. G. Tollis. Issues in interactive orthogonal graph drawing. Proc. Graph Drawing '95. Springer LNCS 1027:419–430, 1996.
21. G. Paris. Cooperation between interactive actions and automatic drawing in a schematic editor. Proc. Graph Drawing '98. Springer LNCS 1547:394–402, 1998.
22. K. Ryall, J. Marks, and S. Shieber. An interactive system for drawing graphs. Proc. Graph Drawing '96. Springer LNCS 1190:387–394, 1997.
23. R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.
24. R. Wiese, M. Eiglsperger, and M. Kaufmann. yfiles: Visualization and automatic layout of graphs. Proc. Graph Drawing '01. Springer LNCS 2265:453–454, 2002.