Semi-dynamic Orthogonal Drawings of Planar Graphs
(Extended Abstract)

Walter Bachl

sd&m AG, Thomas Dehler-Str. 27, 81837 München
Walter.Bachl@sdm.de

Abstract. We introduce a new approach to orthogonal drawings of planar graphs. We define invariants that are respected by every drawing of the graph. The invariants are the embedding together with relative positions of adjacent vertices. Insertions imply only minor changes of the invariants. This preserves the users mental map. Our technique is applicable to two-connected planar graphs with vertices of arbitrary size and degree. New vertices and edges can be added to the graph in $O(\log n)$ time. The algorithm produces drawings with at most $m+f$ bends, where m and f are the number of edges and faces of the graph.

1 Introduction

Orthogonal drawings of graphs are used in many applications, e.g. entity-relationship diagrams or VLSI design. They occur in interactive programs like CASE tools, where graphs are built step by step in an interactive session. Clearly the user shall be supported in this task. Such a system should not only create nice drawings. The given layout shall be changed such that the users mental map is preserved ([7,10]). A lot of work has been done on incremental orthogonal layout of non planar graphs ([11,3,9,1]). But we know only one activity in the area of incremental orthogonal layout of planar graphs [2] which is based on [8]. Brandes et al. optimize a combination of static and interactive requirements, this means layout quality and the preservation of the mental map.

We introduce a different approach with invariants for the representation of the users mental map. We restrict ourselves to two-connected planar graphs and allow changes of the graph by insertions of vertices and edges. These insertions must preserve planarity, the embedding, and the mental map. Our model is based on the concept of the Kandinsky algorithm [8]. There are unit size vertices with sufficiently many pins on each side to connect arbitrary many edges to the node. In the following, we first define the restrictions R and show how update-operations change R. We examine the number of bends which can be achieved with this restrictions. This leads to the 4-sector model. Then we show how to implement the update-operations $O(\log n)$ and how to generate drawings in $O(n)$ time.

M.T. Goodrich and S.G. Kobourov (Eds.): GD 2002, LNCS 2528, pp. 354–361, 2002.

2 Restrictions

We consider two-connected planar graphs together with an embedding. Additionally we define restrictions as invariants for insertions.

Definition 1. *Given a planar graph $G(V, E)$. A* restriction system *R for G consists of*

1. *An embedding of the graph.*
2. *Two attributes $e.\preceq_x$ and $e.\preceq_y$ for each edge $e = \{v_1, v_2\} \in E$.*

A drawing of G is said to respect R, if it is drawn with the embedding and for each edge $e = \{v_1, v_2\}$, if $e.\preceq_x$, then v_1 has to be left of v_2 and if $e.\preceq_y$, then v_1 has to be below v_2. Both relations are proper. Hence, $e.\preceq_x$ and $e.\preceq_y$ have an intuitive meaning.

Let $G(V, E, R)$ denote a graph G with vertices V, edges E and the restriction system R. For the semi-dynamic changes we consider the following operations:

- $initialize()$: Create a complete graph of three vertices as initial graph.
- $insertVertex(edge\ e)$: Replace the edge $e = \{v_1, v_2\}$ by a new vertex v and the two edges $\{v_1, v\}$ and $\{v, v_2\}$.
- $insertEdge(vertex\ v_1, v_2, face\ f)$: Insert the edge $\{v_1, v_2\}$ in the common face f of v_1 and v_2.

This set of operations is sufficient to produce any two-connected planar graph ([5]). The users mental map is preserved by the following requirements.

Definition 2. *Let $G(V, E, R)$ be a planar embedded graph with a restriction system, which is transformed to $G'(V', E', R')$ by an update operation from above.*
$insertEdge(v_1, v_2, f)$ is called restriction preserving update operation *if:*

1. *$\forall e_i \in E$: $e_i.\preceq_x$ and $e_i.\preceq_y$ are not changed.*
2. *The embeddings of $G(V, E, R)$ and $G'(V', E' \setminus \{e\}, R')$ are identical.*

$insertNode(e)$ is called restriction preserving update operation *if:*

1. *$\forall e_i \in E \setminus \{e\}$: $e_i.\preceq_x$ and $e_i.\preceq_y$ are not changed.*
2. *The embeddings of $G(V, E, R)$ and $G'(V', E', R')$ are identical, if the vertex v and the edges e_1 and e_2 are replaced in G' by a new edge $\{v_1, v_2\}$.*
3. *e_1 and e_2 inherit the attributes $\preceq_x$ and $\preceq_y$ from e, i.e. $\{v_1, v\}.\preceq_x = \{v, v_2\}.\preceq_x = \{v_1, v_2\}.\preceq_x \wedge \{v_1, v\}.\preceq_y = \{v, v_2\}.\preceq_y = \{v_1, v_2\}.\preceq_y$.*

3 Bounds on the Number of Bends

We introduce the 4-sector model, which will serve as an invariant for our algorithm. From now on we use figures as graphical representations for restriction systems. If $\{v_1, v_2\}.\preceq_x$, then v_1 is drawn to the left of v_2. Similary for $\preceq_y$.

Definition 3. *A graph with the restriction system $G(V, E, R)$ is in the* 4-sector model *if the vertices around every face f are arranged such that it is possible to divide them into four groups, where in*

- *sector 1, the cyclic successor of every vertex is to its upper right,*
- *sector 2, the cyclic successor of every vertex is to its lower right,*
- *sector 3, the cyclic successor of every vertex is to its lower left,*
- *sector 4, the cyclic successor of every vertex is to its upper left.*

Note, that every sector can be empty or consist of a single vertex only. Figure 1 shows two graphical representations of faces in the 4-sector model. The ordering of the vertices in the 4-sector model is unique. It is a necessary condition for the insertion of a new edge with at most one bend. Is it also sufficient?

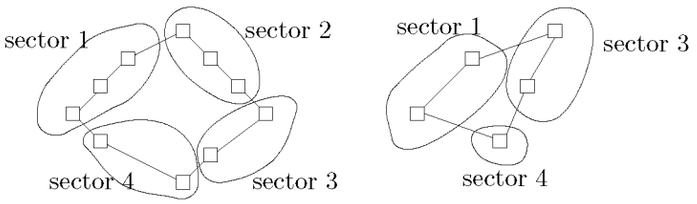

Fig. 1. Two drawings of faces in the sector model.

Lemma 1. *There are graphs with a restriction systems R in the 4-sector model, where a restriction preserving update operation insertEdge inserts an edge with (at least) two bends in any orthogonal drawing respecting R.*

Proof. See Figure 2. Suppose, that by the restriction system R the vertices are in relative positions as shown. Then in any drawing of the graph an edge between v_1 and v_4 in f has to be drawn with two bends.

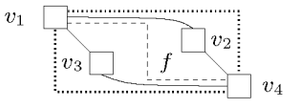

Fig. 2. Orthogonal edge between v_1 and v_4 needs two bends.

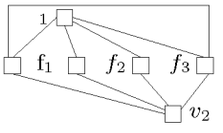

Fig. 3. Notations for the proof of Theorem 1.

This is no restriction of the 4-sector model, because two bends are unavoidable.

Theorem 1. *There are graphs, where a sequence of restriction preserving update operations leads to a graph with a restriction system such that in any orthogonal planar drawing that respects R, at least one edge has at least two bends. This is independent of the initial restriction system.*

Proof. Consider the vertices v_1 and v_2 in Figure 3, which shows an embedding. For any restriction system, a connecting edge $\{v_1, v_2\}$ through at least one of the common faces must have at least two bends.

In the remaining faces we connect the two other vertices by an edge. Thus, one common face of v_1 and v_2 remains, where v_1 and v_2 can be connected only by an edge with at least two bends.

4 Interactive Algorithm

We show how to implement the update operations in $O(\log n)$ and then construct a drawing of a graph with a restriction system in the 4-sector model. $initialize()$ creates a triangle with a restriction system in $O(1)$.

4.1 Insert-Operations

Theorem 2. *Given a planar graph with a restriction system R in the 4-sector model. Then $insertVertex(\text{edge } e)$ can be made restriction preserving such that the resulting graph and the restriction system are in the 4-sector model.*

Proof. The proof is similar to a derivation step in [4]. The edge e is replaced and the new edges inherit $\preceq_x$ and $\preceq_y$ from e. See Figure 4 for an illustration.

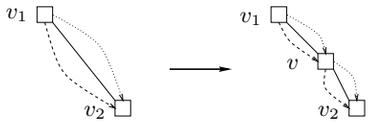

Fig. 4. Insertion of a new vertex where the partial-order in x-direction is represented by a dashed arrow and the partial-order in y-direction by a dotted arrow.

Theorem 3. *Given a planar graph with a restriction system in the 4-sector model. Then $insertEdge(v_1, v_2, f)$ can be made restriction preserving such that the resulting graph and the restriction system are in the 4-sector model.*

Proof. The operation $insertEdge(v_1, v_2, f)$ divides f into two new faces f_1 and f_2. The embedding is updated by the insertion of e between the two edges adjacent to v_1 and the two edges adjacent to v_2 in face f. Hence, the embedding is preserved as required by definition 2.

Next consider the restrictions. The attributes $\preceq_x$ and $\preceq_y$ have to be computed for the new edge. We discuss the main cases. All other cases are similar.

If v_1 and v_2 are in the same sector, then v_1 and v_2 are ordered in both directions and $\preceq_x$ and $\preceq_y$ of the new edge are taken from this ordering.

If v_2 is in the sector following the sector of v_1, then the vertices are ordered only in one direction and unordered in the other. The given direction is taken by the edge, the other is chosen arbitrarily.

Finally, if v_1 and v_2 are in diagonal sectors of the face (e.g. v_1 in sector 1 and v_2 in sector 3), then v_1 and v_2 are unordered in both directions. Because of the geometric properties of the 4-sector model we have one forbidden ordering of the vertices (v_1 can not be right and below of v_2). The algorithm chooses one of the three remaining orderings (e.g. v_1 above and left of v_2).

In all cases, this leads to two new faces which inherit a part of the sectors of face f and the new edge serves as part of one sector in both faces. The graph and the restriction system respect the 4-sector model. No dependencies are changed.

4.2 Data Structures and Time Complexity

Our data structure is an extension of a DCEL ([12]) that is commonly used for planar embeddings. The data structures consist of the following attributes:

- Edge e:
  1. The embedding is stored in a DCEL. Thus every edge has pointers to its cyclic successor and predecessor at both end vertices.
  2. $\preceq_x$ and $\preceq_y$ are stored in the attributes of each edge.
- Face f:
  1. $vertex[1..4]$: $vertex[i]$ is the first vertex of sector i of f.
  2. $edge[1..4]$: $edge[i]$ connects $vertex[i]$ to its cyclic predecessor in f.
  3. $tree[1..4]$: $tree[i]$ is a balanced tree for all vertices in sector i. It supports the operations "insert()", "navigate()" (traversal from a leaf to the root) and "split()" in logarithmic time. At the root we store the number of the face and the number of the sector.
- Vertex v: Every vertex v holds pointers to the edges which end at v in a face f, if v is not the first vertex of a sector in f. The pointers to the edges have a connection to the corresponding face. Since every vertex can be in the middle of a sector in only two faces, two pointers are sufficient.
  1. $edge[1..2]$: $edge[i]$ is a pointer to the edge which connects v to its cyclic predecessor in face f_i.
  2. $leaf[1..2]$: $leaf[i]$ is a pointer to a leaf in the tree, which is assigned to the sector in face i which contains vertex v.

Theorem 4. *The operations $insertNode(e)$ and $insertEdge(v_1, v_2, f)$ can be implemented in $O(\log n)$ time using a data structure of size $O(n)$.*

Proof. For $insertVertex(e)$ the embedding is updated by replacing e by the new edges. Edge e is in two faces. From its end vertices we can access the trees and

insert the new vertex in $O(\log n)$ *time. For* $insertEdge(v_1, v_2, f)$*, we have to find the edges at* v_1 *and* v_2 *between which the new edge has to be inserted. If* v_i *marks the beginning of a sector in* f*, we get the edges from the attributes* $edge[1..4]$ *of* f*. Otherwise we traverse the trees* $leaf[1..2]$ *to the root and compare the face number with* f*. One of them must be* f *and we take the edge from the attribute* $edge[1..2]$*.*

f *is split into two new faces* f_1 *and* f_2*, and we compute the attributes of* f_1 *and* f_2 *from the attributes of* f*. The attributes* $edge[1..4]$ *and* $vertex[1..4]$ *can be computed in constant time from the same attributes of* f *and the vertices* v_1 *and* v_2*.* $tree[1..4]$ *is either a part of a split tree or a full tree of a face* f *or is created with only one vertex. Additionally, the number of the face at the root of every tree has to be updated. By using the trees this takes* $O(\log n)$ *time.*

$\preceq_x$ *and* $\preceq_y$*: For* $insertVertex(e)$ *the replaced edge is given. Thus, we can copy* $\preceq_x$ *and* $\preceq_y$ *in* $O(1)$ *to the new edges. For* $insertEdge(v_1, v_2, f)$*, we have to know the sectors of* v_1 *and* v_2 *in* f *to find valid dependencies for the new edge. If a vertex is the first vertex of a sector, then we can get this information from the attributes of the face. Otherwise, we search in the two trees which are used to find the edges that are adjacent to the vertex in face* f*. Since we store the number of the sector at the root we have access to the number of the sector of the vertex.*

The space requirement for all balanced binary trees is in $O(n)$*, because every vertex is represented in at most two trees. All other attributes have constant size.*

4.3 Drawing Algorithm

We generate an orthogonal representation, and then can use any algorithm which computes an orthogonal drawing from an orthogonal representation (e.g. [8]). The algorithm has the following properties:

1. The drawing respects the restriction system.
2. At most $|F|$ edges have two bends. The remaining edges have one bend.
3. The algorithm takes $O(n)$ time.
4. The routing of an edge is fixed as long as the number of its bends is unchanged. In [6] the routing of an edge is claimed to be part of the mental map.
5. The drawings are in the Kandinsky-Model [8].

Since the embedding is given, only the routing of the edges has to be computed. Every edge starts vertically at the right side of its left adjacent vertex followed by a horizontal segment. If the edge has one bend, it ends at the right vertex. In case of an edge with two bends we have an additional vertical segment. Figure 5 shows all possible edge routings. Thus it remains to compute which edges have two bends and which have only one bend.

Lemma 2. *Given a planar graph and its restriction system in the 4-sector model. An edge e has two bends if there exists a face where*

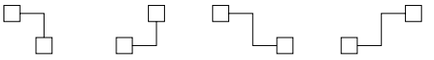

Fig. 5. Possible routings of an edge.

1. *sector 1 and 3 are empty and sector 4 consists only of edge e or*
2. *sector 2 and 4 are empty and sector 1 consists only of edge e or*
3. *the cyclic successor of e at the right end vertex v of e has two bends and both edges leave v to the upper left or*
4. *the cyclic predecessor of e at the right end vertex v of e has two bends and both edges leave v to the lower left.*

All other edges have one bend. The orthogonal representation which is constructed by this rules leads to a planar drawing of the graph.

Proof. Clearly, edges in different sectors do not intersect, since the algorithm of [8] partitions a face into disjoint rectangles. Also edges in the same sector cannot intersect and it remains to show, that edges do not intersect at incident vertices.

Consider the right side of a vertex v. Let $e_1 = \{v, v_1\}$ *be the first edge in counterclockwise order, which ends at the right side of v. Let* $e_2 = \{v, v_2\}$ *be its successor in the counterclockwise order around vertex v and* e_2 *ends at the right side of v.* e_1 *and* e_2 *cannot intersect, if edge* e_1 *leaves v to the lower right and* e_2 *to the upper right. If both* e_1 *and* e_2 *leave v to the lower right, then the vertical segment of* e_1 *has to be left of the vertical segment of* e_2*. This holds as long as* v_1 *and* v_2 *are unordered in x-direction or* v_1 *is to the left of* v_2*. If there is a path in the x-dependencies from vertex* v_2 *to vertex* v_1*, then* v_1 *has to be right of* v_2*. We show, that edge* e_1 *must have two bends and therefore the vertical segment of* e_1 *can be to the left of the vertical segment of* e_2*.*

Consider face f, which is bounded by the edges e_1 *and* e_2 *(Figure 6). Let* e_3 *be the cyclic successor of* e_1 *at vertex* v_1 *and* e_4 *be the last edge in the path in the x-dependencies from* v_2 *to* v_1*. Since* e_4 *is on a path in the x-dependencies, it must be attached to* v_1 *on the left side. The path cannot be to the left of edge* e_1 *and therefore* e_4 *ends above* e_1 *at vertex* v_1*. Since* e_3 *must be between* e_1 *and* e_4*, it also leaves* v_1 *to the upper left and therefore it is in face f in sector 2. Since* e_1 *is in sector 4 and* e_2 *and* e_3 *are in sector 2, all remaining edges are in sector 2 and edge* e_1 *has two bends (rule one).*

All other cases at other sides of the vertices are similar.

Theorem 5. *Given a planar embedded graph with a restriction system in the 4-sector model. Then at most f edges have two bends, where f is the number of faces of the graph.*

5 Further Extensions

Our approach is easy to extend to additional requirements. For example, for graphs with vertices of arbitrary sizes, the drawing algorithm takes the width of

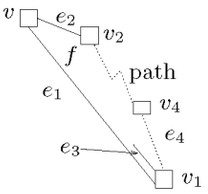

Fig. 6. Notations for the proof of Lemma 2.

the vertices into account, when it evaluates the $\preceq_x$ dependencies and similarily for the height and $\preceq_y$ dependencies.

Acknowledgment

I would like to thank Prof. F.J. Brandenburg for his helpful suggestions. This research was done while the author was with the University of Passau.

References

1. S.S. Bridgeman, J. Fanto, A. Garg, R. Tamassia, and L. Vismara. InteractiveGiotto: An algorithm for interactive orthogonal graph drawing. In *Graph Drawing (GD 97)*, LNCS 1353, pages 303–308. Springer, 1997.
2. U. Brandes, M. Güdemann, and D. Wagner. Fully dynamic orthogonal graph layout for interactive systems. Technical report, Universität Konstanz, 2000.
3. T.C. Biedl and M. Kaufmann. Area-efficient static and incremental graph drawings. In *Proc. European Symposium on Algorithms (ESA '97)*, Lecture Notes in Computer Science 1284, pages 37–52. Springer, 1997.
4. F. J. Brandenburg. Designing graph drawings by layout graph grammars. In *Graph Drawing (GD 94)*, LNCS 894, pages 416–427. Springer, 1995.
5. G. Di Battista and R. Tamassia. On-line planarity testing. *SIAM Journal on Computing*, 25(5):956–997, 1996.
6. S.S. Bridgeman and R. Tamassia. Difference metrics for interactive orthogonal graph drawing algorithms. In *Graph Drawing (GD 98)*, LNCS 1547, pages 57–71. Springer, 1998.
7. P. Eades, W. Lai, K. Misue, and K. Sugiyama. Preserving the mental map of a diagram. In *Proceedings of Compugraphics '91*, pages 24–33, 1991.
8. U. Fößmeier and M. Kaufmann. Drawing high degree graphs with low bend numbers. In *Graph Drawing (GD 95)*, LNCS 1353, pages 254–266. Springer, 1996.
9. U. Fößmeier. Interactive orthogonal graph drawing: Algorithms and bounds. In *Graph Drawing (GD 97)*, LNCS 1353, pages 111–123. Springer, 1997.
10. K. Misue, P.Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *Journal of visual languages and computing*, 6:183–210, 1995.
11. A. Papakostas and I.G. Tollis. Issues in interactive orthogonal graph drawing. In *Graph Drawing (GD 95)*, LNCS 1027, pages 419–430. Springer, 1996.
12. F.P. Preparata and M.I. Shamos Computational Geometry: An Introduction Springer. 1985