Partitioned Drawings*

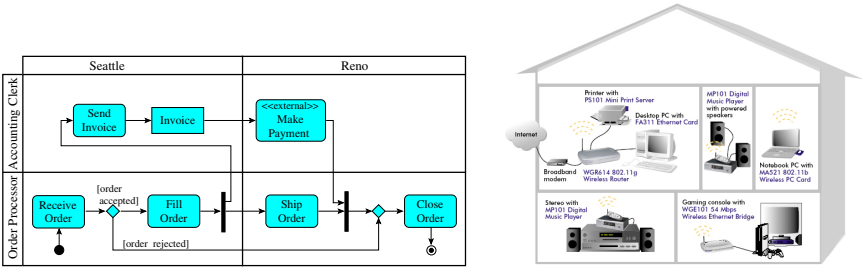
Martin Siebenhaller

Universität Tübingen, WSI für Informatik, Sand 13,
72076 Tübingen, Germany
siebenha@informatik.uni-tuebingen.de

Abstract. In this paper we consider the problem of creating partitioned drawings of graphs. In a partitioned drawing each vertex is placed inside a given partition cell of a rectangular partition of the drawing area. This problem has several applications in practice, e.g. for UML activity diagrams or wiring schematics. We first formalize the problem and analyze its complexity. Then we give a heuristic approach which is based on the topology-shape-metrics approach and produces partitioned drawings in time $O((|V|+c)^2 \log(|V|+c))$, where c denotes the number of crossings.

1 Introduction

In the area of graph drawing there are several approaches for drawing clustered graphs [2]. In a cluster drawing all vertices of a cluster are placed inside the same closed region (usually a rectangle). Regions could be nested, their positions are not given as input. In this paper, we consider the problem of placing each vertex inside a predetermined partition cell of a rectangular partitioned drawing area. Partition cells have fixed positions and do not overlap.



(a) (b)

Fig. 1. Partitioned drawings: (a) shows an UML activity diagram taken from the UML 2.0 specification and (b) a wiring schematic taken from www.netgear.de

Simple partitions use only vertical or horizontal swim-lanes (stripes) to subdivide the drawing area. They are especially useful when we have to emphasize

* This work has been supported by DFG-grant Ka812/8-2.

M. Kaufmann and D. Wagner (Eds.): GD 2006, LNCS 4372, pp. 252–257, 2007.

a logical flow or time flow in a drawing. In UML activity diagrams, partitions are often used to divide a diagram into logical areas, e.g. organizational units in a business model. Activity diagrams also offer more complex, grid like partitions which are a combination of horizontal and vertical swim-lanes (see Figure 1(a)). Irregular partitions (see Figure 1(b)) are often used to indicate geometric information or positions.

The paper is organized as follows: In Section 2 we give a formal definition of the partitioned drawing problem followed by some theoretical results. A heuristic approach is presented in Section 3. We conclude the paper with a short discussion.

2 Definitions

In the following, we assume that the reader is familiar with the concept of planarity, the topology-shape-metrics approach for orthogonal graph drawings and Sugiyama's approach for layered graph drawings, see e.g. [2].

Let A_R denote the (rectangular) drawing area. A *(rectangular) partition* P_R of A_R is a partition of A_R into a set $R = r_1, .., r_k$ of non-overlapping rectangles (=partition cells). Figure 2(a) gives an example. The corresponding *partition grid graph* P_G is constructed by placing a vertex on each point where a horizontal segment touches or intersects a vertical segment. P_G's underlying structure is a regular grid graph, which enables us to assign grid coordinates to the vertices as shown in Figure 2(b). The largest vertical (horizontal) coordinate is denoted by $Y_{\max}$ ($X_{\max}$). For each partition cell $r \in R$ let r^t, r^b, r^l and r^r represent the grid coordinate of the top, bottom, left and right border respectively (e.g. for partition cell r_4 in Figure 2 is $r_4^t = 2$, $r_4^b = 3$, $r_4^l = 3$ and $r_4^r = 4$). In our approach those coordinates do not indicate distances. Only the topology and shape of P_G has to be preserved, the size of the partition cells is not fixed.

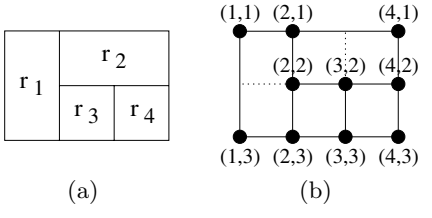


Fig. 2. A rectangular partition (a) and the corresponding partition grid graph (b)

Let $G = (V, E)$ denote an undirected graph, P_R a rectangular partition and $p: V \to R$ a function that maps each vertex to a partition cell.

Definition 1. *A drawing of G is called* partitioned drawing*, if each vertex $v \in V$ is drawn inside $p(v)$. G is called* p-planar *if it has a partitioned drawing without edge crossings.*

Theorem 1. *A graph G is p-planar if and only if it is planar. A (polygonal) p-planar embedding of a p-planar graph can be constructed in time $O(|V|^2)$.*

Proof. A p-planar graph is planar by definition. Let us assume that each vertex $v \in V$ is assigned to an arbitrarily distinct location inside $p(v)$. Pach and Wenger [8] showed that every planar graph admits a planar embedding which maps each vertex to an arbitrarily prescribed distinct location and each edge to a polygonal curve with $O(|V|)$ bends. This embedding can be found in $O(|V|^2)$ time [8] and implies that each planar graph is p-planar. □

An example for such an embedding is given in Figure 3(b). This result has several consequences: Since testing a graph for planarity can be done in linear time [7], the same is true for testing p-planarity. The problem of finding a planarization of a non-planar graph with a minimum number of crossings is NP-hard [6]. Thus the same holds for finding a p-planarization for those graphs.

For an orthogonal partitioned drawing of a graph $G = (V, E)$ we can state the following: if we do not prescribe an embedding, G can always be drawn with less or equal than one bend per edge (omitting self-loops) and thus $\leq |E|$ bends at all. Note, that this bound is tight. A drawing with one bend per edge can simply be realized by placing each vertex $v \in V$ inside $p(v)$ such that there is no pair of vertices having the same x-coordinate (y-coordinate). Then the edges can be routed as in Figure 3(c). This strategy produces only few bends but it does not observe the number of crossings and thus often produce unsatisfying results. For a fixed embedding, the orthogonalization would often produce a lot of bends and strange edge routes.

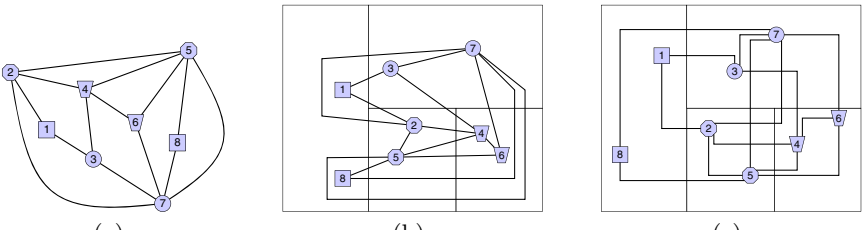


Fig. 3. (a) shows a planar graph and (b) the corresponding p-planar embedding using the partition of Figure 2(a). (c) shows a partitioned drawing with one bend per edge.

In practice we need to find a compromise between minimizing the number of crossings and minimizing the number of bends. Furthermore, we have to incorporate other requirements like vertices of prescribed size or edge labeling.

3 Algorithmic Approach

In this section we present a topology-shape-metrics approach for the automatic layout of partitioned diagrams. More precisely, we sketch the modifications necessary to include partitions into its planarization and orthogonalization phase.

3.1 Planarization

Our planarization strategy is based on the following three steps:

1. **Creating an initial partitioned drawing:**
   We use Sugiyama's approach to create an initial partitioned drawing. It is highly adaptable and especially suited for our purpose even if the input graph is undirected. The modifications for its different phases are quite simple:
   For the layering phase we insert $Y_{\max}$ dummy vertices d_j^y into G which represent the vertical grid coordinates of the partition grid graph P_G. These vertices are connected by directed edges (d_j^y, d_{j+1}^y), $1 \leq j \leq Y_{\max} - 1$. For each vertex $v \in V$ with $p(v) = r$ we insert two directed edges $(v, d_{r^b}^y)$ and $(d_{r^t}^y, v)$. Thus, each feasible layering has the property that a vertex v is placed between the top and bottom border of partition cell $p(v)$.
   The crossing minimization is modified such that the vertex order inside a layer is consistent with the fixed order of the corresponding partition cells.
   For the horizontal coordinate assignment we insert $X_{\max}$ dummy vertices d_j^x into the compaction graph which represent the horizontal grid coordinates of P_G. These vertices are connected by directed edges (d_j^x, d_{j+1}^x), $1 \leq j \leq X_{\max} - 1$. For each vertex v with $p(v) = r$ we insert two directed edges $(v, d_{r^r}^x)$ and $(d_{r^l}^x, v)$ into the compaction graph to guarantee that v is placed between the left and right border of partition cell $p(v)$.
2. **Construction of a p-planar embedding:**
   We "materialize" the partition P_R by inserting the corresponding partition grid graph P_G into the drawing constructed in the above step. Each vertex $v \in P_G$ is placed on its related coordinate (d_i^x, d_j^y). To create a p-planar embedding, we detect crossings with a sweep-line algorithm and replace them by dummy vertices. This can be done in time $O(|S| \log |S| + c)$ where c denotes the number of crossings and S the set of segments. In our approach the number of segments is $|S| = O(|V| + |E|)$. Now, we determine the cyclic order of the edges around each vertex.
3. **Rerouting of edges:**
   The layered drawing produced by Sugiyama's approach is too restrictive because undirected edges can be routed non-monotonically. Thus, we perform a rerouting step to further reduce the number of crossings. The rerouting is based on shortest-path computations in the dual graph [2]. Since the size of the planarized graph is $O(|V| + c)$, the runtime of the rerouting is $O((|V| + c)|E|)$.

3.2 Orthogonalization

The orthogonalization phase has to preserve the shape of the partition grid graph P_G. The phase is divided into two steps:

1. First, we fix the angles between consecutive edges around each vertex of P_G. The angles have to comply with the partition structure.
2. In the second step, we calculate the shape of the edges. The fixed angles assigned in the first step are not allowed to change. Furthermore, the edge segments of P_G are not allowed to bend. Hence, we assign high bend costs to those segments. The shape calculation of the edges is done with the network flow approach described in [5] applying the modifications of [1,3] that guarantee the conservation of prescribed angles and bends. We use a heuristic network solver that produces satisfying results in practice and has running time $O((|V|+c)^2 \log(|V|+c))$. There is always a valid drawing in which the fixed angles and shapes are observed [3].

4 Discussion and Conclusion

First we discuss the running time of our new approach. We assume that the number of partition cells is $O(|V|)$. With the efficient implementation of Sugiyama's algorithm described in [4], we can create the initial partitioned drawing in time $O((|V|+|E|)\log|E|)$. The calculation of the planar embedding has time $O((|V|+|E|)\log(|V|+|E|)+c)$ and the rerouting $O((|V|+c)|E|)$ (c denotes the number of crossings). Since the planarized graph has $|V|+c$ vertices the number of edges is $|E| = O(|V|+c)$. The orthogonalization step takes time $O((|V|+c)^2 \log(|V|+c))$. For the compaction we use the fast constructive algorithm described in [3] with a flow-based post-processing step and quadratic runtime. We sum up with the following theorem:

Theorem 2. *Given an undirected graph $G = (V, E)$, a rectangular partition P_R of the drawing area and a mapping p of the vertices to partition cells. Our approach creates a partitioned drawing of G in time $O((|V|+c)^2 \log(|V|+c))$ where c denotes the number of crossings in the planarized graph.*

We implemented our approach in Java using the yFiles library [10]. Two example layouts are given in Figure 4. More examples can be found at `http://www-pr.informatik.uni-tuebingen.de/partitioned-drawings/`. For diagrams with about 100 vertices and 150 edges we typically need a runtime of less than three seconds on a 3 GHz Pentium IV System with 1 GB RAM.

In this work we introduced the problem of finding a partitioned drawing of a graph $G = (V, E)$. We presented a heuristic approach that produces pleasing results in time $O((|V|+c)^2 \log(|V|+c))$. The approach can easily be extended to include edge labels or edges that should be drawn upward [9].

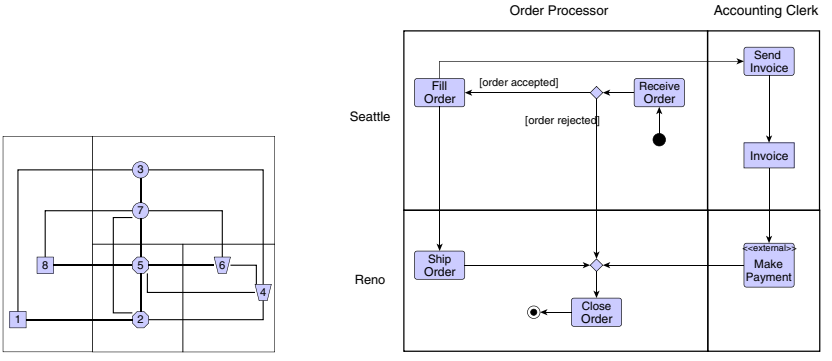


Fig. 4. Examples drawn with our new approach: (a) shows our layout for the example of Figure 3 and (b) the UML activity diagram of Figure 1(a)

References

1. U. Brandes, M. Eiglsperger, M. Kaufmann, and D. Wagner. Sketch-driven orthogonal graph drawing. In *Proceedings of the 10th International Symposium on Graph Drawing (GD'02)*, volume 2528 of *LNCS*, pages 1–12. Springer, 2002.
2. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.
3. M. Eiglsperger. *Automatic Layout of UML Class Diagrams: A Topology-Shape-Metrics Approach*. PhD thesis, Universität Tübingen, 2003.
4. M. Eiglsperger, M. Siebenhaller, and M. Kaufmann. An efficient implementation of Sugiyama's algorithm for layered graph drawing. In *Proceedings of the 12th Symposium on Graph Drawing (GD'04)*, volume 3383 of *LNCS*, pages 155–166. Springer, 2005.
5. U. Fößmeier and M. Kaufmann. Drawing high degree graphs with low bend numbers. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*, volume 1027 of *LNCS*, pages 254–266. Springer, 1996.
6. M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4:312–316, 1983.
7. J. E. Hopcroft and R. E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974.
8. J. Pach and R. Wenger. Embedding planar graphs at fixed vertex locations. In *Proceedings of the 6th Symposium on Graph Drawing (GD'98)*, volume 1547 of *LNCS*, pages 263–274. Springer, 1998.
9. M. Siebenhaller and M. Kaufmann. Mixed upward planarization - fast and robust. In *Proceedings of the 13th Symposium on Graph Drawing (GD'05)*, volume 3843 of *LNCS*, pages 522–523. Springer, 2006.
10. yWorks. yFiles - a java graph layout and visualization library (WWW document). `http://www.yworks.com` (accessed September 2006).