Applying Crossing Reduction Strategies to Layered Compound Graphs

Michael Forster

University of Passau,
94030 Passau, Germany
forster@fmi.uni-passau.de

Abstract. We present a new method for the application of 2-layer crossing reduction algorithms to layered compound graphs. It is based on an algorithm by Sander [9, 7, 8] and improves it with fewer crossings. Our method is optimal in the sense that it does not introduce unnecessary crossings by itself. If used with an optimal 2-layer crossing reduction algorithm, the crossing reduction for 2-layer compound graphs is optimal, too.

1 Introduction

Compound graphs are graphs with two sets of edges, adjacency edges and hierarchy edges. The hierarchy edges are interpreted as an inclusion relationship. They are also called "inclusion edges". There are different notions of compound graphs. Some only restrict the hierarchy edges to be acyclic, others require them to form a tree. This paper uses the latter concept.

There are many areas of application for compound graphs. The motivation for our research on the topic comes from the visualization of biochemical pathways. This is an interesting area of application for graph drawing. Parts of a reaction network take place in different regions of the cell, e. g., in the nucleus or in the cytoplasm. Moreover, there are situations where these regions are nested. This leads to clustered graphs, a special type of compound graphs, where adjacency edges are only allowed between leaves of the hierarchy tree.

The usual way to draw compound graphs is as follows: compound nodes, i. e., nodes that contain other nodes are drawn as rectangles. The contained nodes – and only these – are drawn within the rectangle. There are some extensions of layout algorithms for planar graphs and of force directed algorithms to compound graphs. See [1] for an overview.

Since we focus on directed compound graphs, we use a drawing style where the nodes are partitioned into layers. All nodes of one layer are drawn on a horizontal line. Edges are routed top down. This is similar to the layout style generated by the well-known algorithm of Sugiyama, Tagawa, and Toda for the layout of directed acyclic graphs, see Fig. 1.

M.T. Goodrich and S.G. Kobourov (Eds.): GD 2002, LNCS 2528, pp. 276–284, 2002.

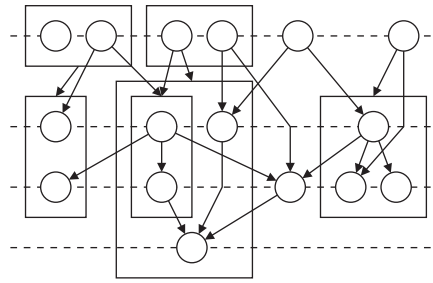
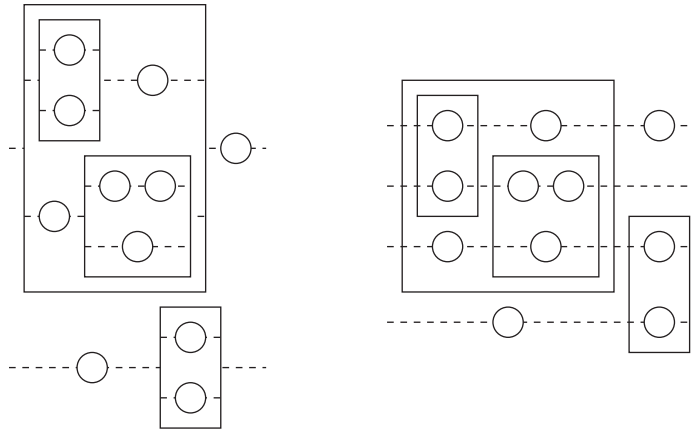


Fig. 1. A layered compound graph

2 Related Work

There are two variants of layered drawings of compound graphs: If using *local layers*, each node spans exactly one layer. The interior of compound nodes has its own interior layers that are independent of the exterior layers. The representative algorithm with local layers is an algorithm by Sugiyama and Misue [13]. With *global layers*, there is only one set of layers for all nodes. Compound nodes are allowed to span multiple layers. This is the drawing style used by Sander [9,7,8].

Drawings with local layers typically can be computed faster, while global layers give more compact drawings. Fig. 2 shows the same compound graph drawn with both variants. See [7, 8] for a more detailed comparison. For this paper, we will focus on a global layering scheme.



(a) Local layers (b) Global layers

Fig. 2. Local vs. global layers

An obvious approach for drawing directed compound graphs is to draw the interior and exterior of compound nodes independently of each other. First, the interior of the compound nodes at the lowest level of the hierarchy is drawn. Then the internal structure is hidden and the compound node is treated as a single big node.

Instead of working the hierarchy bottom up, it is possible to start at the root of the hierarchy, first drawing the outside of the compound nodes and then inserting the contents recursively. For this variant, the layering of the graph must be computed as a preprocessing step, because the size of all compound nodes must be known before the top level layering can be done.

No matter whether the hierarchy is traversed top down or bottom up, this strategy leads to unnecessary edge crossings, since parts of the layout are computed without considering the global connectivity of the graph. Fig. 3(a) shows a given compound graph, drawn without any edge crossings. Fig. 3(b) shows the graph after the first step of the algorithm. The top level of the hierarchy has been drawn, the interior of the compound node has been ignored. Up to now, we have no crossings, but the positions of the shown nodes hence are fixed. As a result, any order of the interior nodes will lead to an edge crossing, see Fig. 3(c). The other three images of Fig. 3 show a similar example for an unnecessary edge crossing, when traversing the hierarchy bottom up.

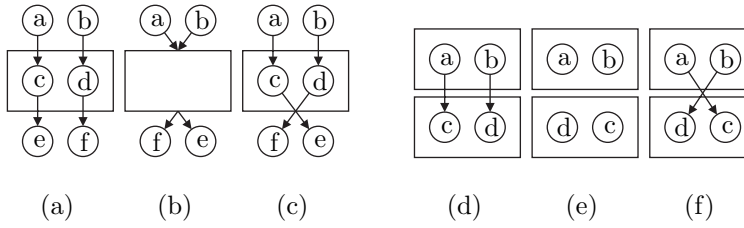


Fig. 3. Unnecessary edge crossings when considering hierarchy levels separately

It is well known that crossing reduction is NP-hard already for ordinary graphs, even if they only consist of two layers [3]. Therefore we cannot expect an efficient optimal crossing reduction algorithm for compound graphs. But in this case, the crossings do not originate from the used crossing reduction method itself, they arise from the application of the crossing reduction method to the compound graph. This means, that even with an optimal 2-layer crossing reduction strategy for ordinary graphs, these crossings are unavoidable.

Sander [9,7,8] uses a different approach. His algorithm works as the algorithm of Sugiyama, Tagawa, and Toda for the layout of directed acyclic graphs [13]. The hierarchical structure is considered only when it is necessary to prevent violations of the drawing conventions. Since Sander's method is the base of our algorithm, we recall it in section 4, in direct comparison to our method.

3 Notation

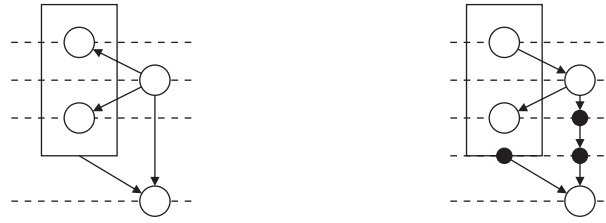
A *(directed) graph* $G = (V, E)$ consists of nodes V and edges $E \subseteq V \times V$. The *direct predecessors* and *direct successors* of a node $v \in V$ are defined as $\text{pred}_G(v) = \{ u \in V \mid (u, v) \in E \}$ and $\text{succ}_G(v) = \{ u \in V \mid (v, u) \in E \}$. The *successors* $\text{succ}^*_G(v)$ and *predecessors* $\text{pred}^*_G(v)$ of v are the respective reflexive transitive closures.

A *compound graph* $G = (V, E, H)$ consists of a graph (V, E) with *adjacency edges* E and an additional relation $H \subseteq V \times V$ such that the graph $T = (V, H)$ is a directed rooted tree with its edges oriented from the root towards the leaves. T is called the *hierarchy tree* of G, its edges are called *hierarchy edges*. The nodes of a compound graph G are partitioned into *base nodes* $B = \{ v \in V \mid \text{succ}_T(v) = \emptyset \}$ and *compound nodes* $C = V \backslash B$. $G|_B$ is called the *base graph* of G.

A *layered compound graph* $G = (V, E, H, L)$ is a compound graph (V, E, H) where each base node is assigned to a positive integer $L \colon B \to \mathbb{N}$. B is partitioned into disjoint layers $B_1, \ldots, B_n$, where $B_i = \{ v \in B \mid L(v) = i \}$. The minimum (maximum) layer $L_{\min}(v)$ ($L_{\max}(v)$) of a node $v \in V$ is defined as the minimum (maximum) layer number of the leaves reachable in the hierarchy tree. For a base node $b \in B$ this results in $L_{\min}(b) = L_{\max}(b) = L(b)$.

The *layer hierarchy tree* $T_i = (V_i, H_i)$ for the i-th layer of an layered compound graph $G = (V, E, H, L)$ is that part of the hierarchy tree that is relevant for the i-th layer, its nodes are $V_i = \{ v \in V \mid L_{\min}(v) \leq i \leq L_{\max}(v) \}$.

A *(layered) clustered graph* is a (layered) compound graph where edges are only allowed between the base nodes. A layered compound (or clustered) graph is said to be *proper* if every edge $(u, v) \in E$ is directed downwards and spans exactly one layer, i. e., $L_{\min}(v) = L_{\max}(u) + 1$. This implies that there are no adjacency edges between a node and a predecessor or successor of it in the hierarchy tree. Any layered compound graph can be converted to a proper layered clustered graph by reversing edges and inserting dummy nodes, see Fig. 4 for an example.



(a) A layered compound graph (b) Inserted dummy nodes

Fig. 4. Converting a layered compound graph to a proper layered clustered graph

4 Crossing Reduction

Reducing the crossings in the drawing of a compound graph is a crucial step in the layout process. We assume that the given compound graph has already been layered and has been converted to a proper layered clustered graph by reversing some edges and inserting dummy nodes. This can be done as described in [7, 8].

The crossing reduction method for clustered graphs proposed by Sander is based on the crossing reduction step in the conventional Sugiyama algorithm for drawing directed acyclic graphs. It starts with a random order of the topmost layer. It then iteratively considers each layer from top to bottom and reorders the nodes on that layer such that the crossings between incoming edges are reduced. When the algorithm has reordered the last layer, it traverses the graph from bottom to top, reordering the layers so that crossings between outgoing edges are reduced. These two traversals are repeated until some termination criterion is met. Since the problem to minimize the crossings between two layers is NP-hard, heuristics are used, e. g. the well-known barycenter heuristic, and there are others with varying quality and running times [2, 4, 5].

The main idea of our algorithm is a scheme, how any standard 2-layer crossing reduction method can be applied to proper layered clustered graphs. The main part of the crossing reduction step stays the same, but only base nodes are considered. The base graph is traversed top down and bottom up in the same way, considering the hierarchy only when doing the 2-layer crossing reduction.

There are two new restrictions which must be satisfied when reducing crossings in clustered graphs as opposed to unclustered graphs.

R1: The nodes on a layer that belong to the same compound node must be placed next to each other with no other nodes between them.
R2: The relative position of two compound nodes must be the same on all layers, i. e., compound nodes must not "cross" each other.

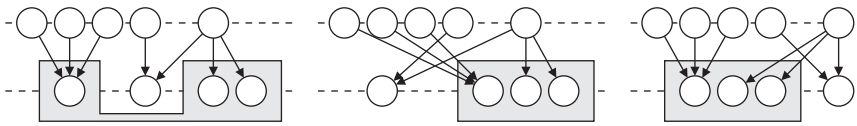
If any of these restrictions is violated, it is not possible to draw compound nodes as rectangles that contain exactly the respective subtree of the hierarchy, see Fig. 5 for examples.

It is the strict enforcement of these extra restrictions that differentiates our method from that of Sander. In Sander's method the hierarchy is ignored first,

(a) Violation of R1 (b) Violation of R2

Fig. 5. Violations of the order restrictions

which leads to violations of the restrictions. These violations are resolved afterwards by sorting the compound nodes by the barycenter of the contained base nodes. Only the intermediate base node order is considered and the adjacency edges are ignored. It is easy to construct simple examples, where this strategy gives many unnecessary crossings, see Fig. 6. Our strategy is to consider the restrictions right from the start.



(a) before reordering (b) after reordering (c) optimal result

Fig. 6. Unnecessary crossings with Sander's method

4.1 Respecting the Hierarchy on a Single Layer (R1)

Our method works by modifying the 2-layer crossing reduction step used as a subroutine in the global crossing reduction. We assume a layered clustered graph $G = (V, E, H, L)$, with two layers B_1 and B_2 and compound nodes C, $V = B_1 \cup B_2 \cup C$, its hierarchy tree $T = (V, H)$, and two layer hierarchy trees T_1, T_2. The order of B_1 is fixed, B_2 must be reordered. We consider the layer hierarchy tree of the bottom layer T_2 and ignore R2 for now. We observe the following lemma:

Lemma 1. *An order of the base nodes complies with R1 if and only if there exists a child order of the layer hierarchy tree so that the order of the base nodes can be obtained by a preorder (postorder) traversal.*

Because of this observation we will concentrate on finding child orders for the layer hierarchy tree and from this obtain the order of the base nodes. The following lemma characterizes the relationship between edge crossings and the child order of some node in the layer hierarchy tree:

Lemma 2. *Let $e_1 = (s_1, t_1)$, $e_2 = (s_2, t_2) \in E$ be two adjacency edges in G, and let t be the lowest common ancestor of t_1 and t_2 in the layer hierarchy tree with two children t'_1 and t'_2 that are predecessors of t_1 and t_2, respectively. The edges e_1 and e_2 cross if and only if t'_1 and t'_2 have a different relative order than s_1 and s_2.*

Thus each possible edge crossing can be associated with a unique compound node in the layer hierarchy tree. Also each edge crossing is independent of the child orders of all other hierarchy tree nodes in T_2. The number of crossings in some order of the base nodes is the sum of the crossings associated to each layer hierarchy node. We get the following theorem:

Theorem 1. *An order of the base nodes has a minimal number of crossings if and only if the corresponding layer hierarchy tree has a minimal number of crossings at each node.*

By Theorem 1, it is justified to compute the child order of all compound nodes independently without losing quality. To minimize the number of crossings associated to a compound node $h \in V_2$ we construct a new graph G'_h called the crossing reduction graph of h. G'_h consists of two layers. The upper layer B_1 stays the same as in G, the lower layer B'_2 consists of the children of h in T_2. The relevant adjacency edges of G are then transferred to G'_h in the following manner:

- Edges (s,t) ending in a node t that is not a successor of h in T_2 are ignored.
- For each remaining edge (s,t), an edge (s,c) with weight $w(s,c) = 1$ is created, where c is the unique child of h which is a predecessor of t. If the edge already exists, its weight is increased by 1.

We get the following weight function

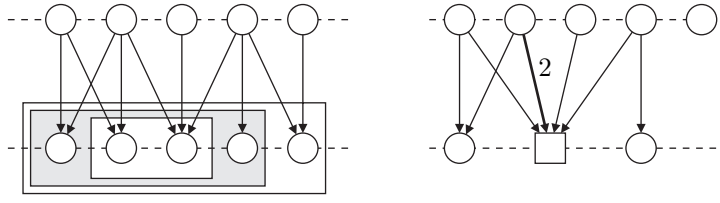
$$w\colon \begin{cases} B_1 \times B'_2 & \to \mathbb{N}_0, \\ (s,c) & \mapsto \left|\{\, t \in B_2 \cap \mathrm{succ}^*_{T_2}(c) \mid (s,t) \in E \,\}\right| \end{cases}$$

and a corresponding weighted graph G'_h:

$$\begin{aligned} G'_h &= (V'_h, E'_h, w) \\ V'_h &= B_1 \cup B'_2, \quad B'_2 = \mathrm{succ}_{T_2}(h) \\ E'_h &= \{\, (s,c) \in B_1 \times B'_2 \mid w(s,c) > 0 \,\}\ . \end{aligned}$$

See Fig. 7 for an illustration. Please note that all crossing reduction graphs can be precomputed before starting the traversal of the graph, since they do not change during the crossing reduction.

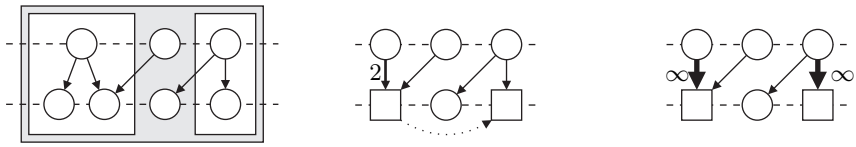
A conventional algorithm for weighted 2-layer crossing reduction is then applied to the crossing reduction graph and the given order of the upper layer. The resulting order of the lower layer is used as the order for the children of h. In the same way a child order for all other internal nodes of the layer hierarchy tree is computed, and we get an order for the base nodes by traversing the tree.



(a) A clustered graph

(b) The crossing reduction graph of the grey node

Fig. 7. Creating the crossing reduction graph



(a) Compound nodes across two layers (b) Constraint method (c) Heavy edge method

Fig. 8. Respecting the hierarchy across layers (R2)

4.2 Respecting the Hierarchy Across Layers (R2)

To respect R2 right from the start, we present two alternative strategies: the constraint method and the heavy edge method.

The constraint method is the more promising alternative. It guarantees the compliance with R2, but it depends on a 2-layer crossing reduction algorithm that supports constraints, i. e., predefined relative orders of some node pairs. Some of the conventional crossing reduction methods have been extended to support constraints with varying success. There is also a crossing reduction method by Schreiber [10], that has been designed specifically for the support of constraints.

To comply with R2, the constraint method prohibits the relative position of two compound nodes spanning adjacent layers being different. This is done by inserting a constraint into the crossing reduction graph of one specific node. In Fig. 8(b) the dotted arrow shows how constraints are inserted. Please note that it is sufficient to add constraints between compounds nodes having the same parent in the layer hierarchy tree.

An alternative to the constraint method is the heavy edge method. It does not need a crossing minimization algorithm that supports constraints, but models compound nodes as edges with a very high weight instead. These are labeled with ∞ in Fig. 8(c). Unfortunately, this approach does not guarantee compliance with R2 when used with a non-optimal crossing reduction algorithm. Consequently it may be necessary to reorder the nodes in the same way as Sander.

Additionally, these edges penalize crossings between adjacency edges and compound nodes. Avoiding these crossings make drawing better understandable, but it also leads to more crossings between adjacency edges.

5 Discussion

We presented a new method for crossing reduction in layered compound graphs. Our method improves Sander's method with regard to the number of crossings. Depending on structure of the hierarchy we may also gain running time improvements. An implementation of the algorithm is in progress, using Schreiber's 2-layer crossing minimization with constraints support [10]. We shall have experimental results soon.

References

1. Ralf Brockenauer and Sabine Cornelsen. Drawing clusters and hierarchies. In Kaufmann and Wagner [6], chapter 8, pages 193–227.
2. Giuseppe di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing*. Prentice Hall, 1999.
3. Peter Eades and Nicholas C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11:379–403, 1994.
4. Michael Jünger and Petra Mutzel. Exact and heuristic algorithms for 2-layer straightline crossing minimization. In *Proc. Workshop on Graph Drawing '95*, pages 337–348. Springer, 1996.
5. Michael Jünger and Petra Mutzel. 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. *Journal of Graph Algorithms and Application*, 1(1):1–25, 1997.
6. Michael Kaufmann and Dorothea Wagner, editors. *Drawing Graphs*, volume 2025 of *Lecture Notes in Computer Science*. Springer, 2001.
7. Georg Sander. Layout of compound directed graphs. Technical Report A/03/96, Universität Saarbrücken, 1996.
8. Georg Sander. *Visualisierungstechniken für den Compilerbau*. PhD thesis, Universität Saarbrücken, 1996.
9. Georg Sander. Graph layout for applications in compiler construction. *Theoretical Computer Science*, 217:175–214, 1999.
10. Falk Schreiber. *Visualisierung biochemischer Reaktionsnetze*. PhD thesis, Universität Passau, 2001.
11. Falk Schreiber. High quality visualization of biochemical pathways in biopath. *In Silico Biology*, 2(0006), 2002. `http://www.bioinfo.de/isb/2002/02/0006/`.
12. Kozo Sugiyama and Kazuo Misue. Visualization of structural information: Automatic drawing of compound digraphs. *IEEE Trans. Systems, Man and Cybernetics*, 21(4):876–892, 1991.
13. Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Systems, Man and Cybernetics*, SMC-11(2):109–125, 1981.