Determining Sugiyama Topology with Model Order

Sören Domrös
Department of Computer Science, Kiel University, Germany

Reinhard von Hanxleden
Department of Computer Science, Kiel University, Germany

Abstract

Traditional implementations of the Sugiyama algorithm optimize aesthetic criteria such as the number of backward edges, edge length, or edge crossings. If we, however, utilize the model order, as provided e. g. by a textual graph input file, we can determine the topology of a Sugiyama layout in a one-pass algorithm while controlling the secondary notation and with it the intention expressed by the underlying model, which typically cannot be captured by layout algorithms.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases Automatic Layout, Model Order, Layered Layout

Digital Object Identifier 10.4230/LIPIcs.GD.2024.48

Category Poster Abstract

1 Introduction

Jünger and Mutzel [4] pointed out that in "many cases, such as the drawing of flowcharts, the input data can be expected to determine the choice" of edge reversal when explaining the cycle breaking step of the Sugiyama [6] algorithm.

We here propose to go even further and to use the "input data", also referred to as *model order* [1, 2], to determine the whole topology of a layered layout.

2 Efficient Solutions to Layered Layout Topology

The *Sugiyama* or *layered* algorithm is divided into five phases: the *topological phases* cycle breaking, layer assignment, and crossing minimization and the *geometrical phases* node placement and edge routing. Domrös et al. [1] presented an approach to partially solve the topological phases in linear time, but requiring intermediate processors that sorted the nodes before each step. They argued that a manually written input model, e. g. a program in a domain-specific language, carries intention and secondary notation that can be transferred into the layout, which can typically not be captured by static layout algorithms [5]. We here showcase under which assumptions this is possible and show obviously not optimal (ONO) layouts that might be created by this approach. Directly controlling the layout by the input model increases control over the layout without introducing additional layout constraints. It also is naturally embedded in the typical editing workflow for programming.

We assume a directed ordered graph $G = (V, E)$ with an ordered set of nodes V and $|V| = n$ and ordered set of edges E with $|E| = e$ that corresponds to an input model with nodes ordered breadth-first based on their desired position and edges ordered based on their desired destination, as visualized in Figure 1.

The *cycle breaking step* can be trivially solved in $\mathcal{O}(e)$ by iterating over all edges and reversing all edges for which the source node has a higher model order than the target node [1], as seen in Figure 1a at the edges between `n5` and `n2`. Note that this solution might not find the minimal feedback arc set, but assuming that the model order of the input model is intended this would be acceptable.

(a) ONO backward edges and edge crossings.

(b) ONO edge length.

Figure 1 Two example drawings that determine topology from model order. Figure 1a is not optimal regarding backward edges and edges crossings. Figure 1b is not optimal regarding edge length. However, both layouts reflect model order.

The *layer assignment step* assigns the first node to the first layer and continues iterating over all nodes and assigns them to the current layer until a node occurs that is connected to an already placed node, which creates a new layer, as seen in Figure 1b. By marking each target node of all already handled nodes as a node that needs to be in a new layer and assuming the breadth-first order of nodes, we can limit the runtime of the layer assignment to $\mathcal{O}(n + e)$ without considering dummy nodes. Additionally, we need to mark potential positions for long edge dummy nodes that need to be introduced if an edge spans multiple layers to create a proper layered graph. By considering the sources of a dummy node and the nodes in the previous layer, the dummy nodes can be directly assigned to their correct position in a layer without an additional sorting step. This improves the layer assignment post-processing proposed by Domrös et al. [1], which requires a sorting step. Again, this layer assignment may not minimize the edge length but instead expresses the model order.

Crossing minimization can be skipped entirely since the nodes are already in the intended order and the edge order corresponds to the order of the nodes in a sensible model. The order of outgoing edges on a node and the position of the dummy nodes introduced for long edges already determine the order of nodes, dummy nodes, and edges and hence the whole topology. Since this, however, limits the position of dummy nodes and can only statically assign dummy nodes to a specific position relative on real nodes, sorting, as proposed by Domrös et al. [1], or greedily switching nodes and edge anchor points as proposed by Eades and Kelly [3] might still be beneficial, which would increase the runtime complexity.

3 Obviously not Optimal Input Models

When comparing dummy nodes to real nodes that are part of the input model when deciding for potential routes during layer assignment, one has to statically decide whether a dummy node is below or above a so-called *dangling source node* that has no connection to a previous layer. E. g., in Figure 1a dummy node are placed above real nodes and as a result `n4` is below the edges from `n5` to `n2`. This means, we cannot control this by reordering the input model.

Another concern are edges that require an additional edge dummy in the same layer to route the edge around its node, which we also refer to as *feedback edges*. Here, similar to the dummy nodes compared to dangling source nodes, we cannot statically decide whether the feedback edge dummy should be above or below a node without creating unnecessary edge crossings that cannot be prevented by controlling the input model.

Finally, the approach presented here, while very fast, is probably only sensible if the model order of the input model carries indeed intention, as it does not aim to optimize standard aesthetic criteria such as crossing minimization. If only the order of nodes is intentional, or only the order of edges, one might prefer to fall back to approaches proposed by Domrös et al. [1, 2] that enforce node or edge order selectively, use them only as a tie-breaker, or utilize different partial ordering groups to only consider model order that corresponds to intention.

References

1 Sören Domrös, Max Riepe, and Reinhard von Hanxleden. Model order in Sugiyama layouts. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2023) - Volume 3: IVAPP*, pages 77–88. INSTICC, SciTePress, 2023. doi:10.5220/0011656700003417.

2 Sören Domrös and Reinhard von Hanxleden. Diagram control and model order for sugiyama layouts. In *Diagrammatic Representation and Inference, 14th International Conference, DIAGRAMS '24*. To be published, 2024.

3 Peter Eades and David Kelly. Heuristics for reducing crossings in 2-layered networks. *Ars Combinatoria*, 21:89–98, 1986.

4 Michael Jünger and Petra Mutzel, editors. *Graph Drawing Software*. Springer, 2004. doi:10.1007/978-3-642-18638-7.

5 Marian Petre. Why looking isn't always seeing: Readership skills and graphical programming. *Communications of the ACM*, 38(6):33–44, June 1995. doi:10.1145/203241.203251.

6 Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics*, 11(2):109–125, February 1981. doi:10.1109/TSMC.1981.4308636.