The Eclipse Layout Kernel

Maximilian Kasperowski
Department of Computer Science, Kiel University, Germany

Sören Domrös
Department of Computer Science, Kiel University, Germany

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

Abstract

The Eclipse Layout Kernel (ELK) is an open-source framework[1] written in Java, which is transpiled to the JavaScript library elkjs[2]. ELK provides extensible and modular algorithms, visibility for diagramming research, and has an active community. The ELK project is both a validation platform for graph drawing algorithm researchers, and a freely available library put in production use to provide automatic layout for academic and commercial applications.

The report by Domrös et al. [3] presents an overview of the available algorithms, the development history, related publications, as well as lessons learned from developing the open-source framework. ELK welcomes new users as well as new contributors.

2012 ACM Subject Classification Software and its engineering → Software libraries and repositories

Keywords and phrases Automatic Layout, Layered Layout, Layout Library

Digital Object Identifier 10.4230/LIPIcs.GD.2024.56

Category Software Abstract

1 ELK's Infrastructure and Applications

ELK provides infrastructure to develop algorithms for arbitrary types of graphs including digraphs, hypergraphs, and compound graphs that may be layouted bottom-up or top-down [6] and allows using different algorithms in different subgraphs. ELK's phase and processor infrastructure supports pre- and post-processors that can be modularly inserted into any algorithm. This allows solving domain-specific layout questions as well as comparing layout strategies without reimplementing whole algorithms.

ELK contains highly configurable[3] algorithms for layered layouts, rectangle packing, tree drawing, force and stress layouts, radial layouts, and packing of disconnected components. The flagship algorithm is ELK Layered based on the Sugiyama algorithm [11]. It can be configured with over 140 layout options to control details such as spacing for edges, nodes, disconnected components and ports, as well as what strategies for different phases of the layout algorithm. These include model order, node and port constraints, node size constraints, compaction, edge wrapping, label placement, self-loop arrangement, and layout direction.

Notable layout strategies that have been implemented for ELK Layered based on published research include Forster constraint resolving during barycenter crossing minimization [4], Brandes and Köpf node placement [1], edge bundling for dataflow diagrams [9] and size-aware and port-aware horizontal node coordinate assignment by Rüegg et al. [8], and port constraints in dataflow diagrams by Spönemann et al. [10]. Moreover, Jabrayilov et al. and Rüegg et al. have worked on the graph layering problem for general directed graphs [5, 7].

[1] `https://eclipse.dev/elk/`
[2] `https://www.npmjs.com/package/elkjs`
[3] `https://eclipse.dev/elk/reference/options.html`

ELK and elkjs are integrated into popular diagramming tools such as mermaid, GLSP, reactflow, Lingua Franca, Sirius, Sprotty, and KIELER, which provide real world layout problems to solve and makes graph drawing research results visible and usable. Particularly elkjs has become very popular[4] in recent years, as it lets users embed automatic graph visualizations within web applications. elkjs has received about 1700 "stars" on GitHub and has currently over 800 000 weekly downloads.

Much of ELK's value stems from community interaction. This interaction provides us with valuable insights on how real applications utilize graph drawing. GitHub or gitter[5] can be used to ask layout questions or report problems. The elklive tool[6] allows sharing graph configurations in the online editor via a simple link. One key insight of the community interaction is that users do not want complicated solutions but rather control over the layouts.

2 Future Work

Future work focuses on laying the groundwork for visualizing large hierarchical models and exploring how model order [2] can be utilized by layout algorithms.

Top-down layout is an approach to draw nested graphs that serves as an alternative to the bottom-up approach [6]. By drawing nested graphs starting at the root instead of at the leaves we obtain better high-level overviews, especially for large nested graphs.

Model order aims to bring secondary notation and with it intention from a textual model into a diagram. This can determine the whole topology of a layered graph by model order.

References

1 U. Brandes and B. Köpf. Fast and simple horizontal coordinate assignment. In *Proc. GD '01*, 2002. `doi:10.1007/3-540-45848-4`.

2 S. Domrös, M. Riepe, and R. von Hanxleden. Model order in Sugiyama layouts. In *Proc. VISIGRAPP 2023 - Volume 3: IVAPP*, 2023. `doi:10.5220/0011656700003417`.

3 S. Domrös, R. von Hanxleden, M. Spönemann, U. Rüegg, and C. D. Schulze. The Eclipse Layout Kernel, 2023. `doi:10.48550/arXiv.2311.00533`.

4 M. Forster. A fast and simple heuristic for constrained two-level crossing reduction. In *Proc. GD '04*, 2005. `doi:10.1007/978-3-540-31843-9_22`.

5 A. Jabrayilov, S. Mallach, P. Mutzel, U. Rüegg, and R. von Hanxleden. Compact layered drawings of general directed graphs. In *Proc. GD '16*, 2016. `doi:10.1007/978-3-319-50106-2`.

6 M. Kasperowski and R. von Hanxleden. Top-down drawings of compound graphs, December 2023. `doi:10.48550/arXiv.2312.07319`.

7 U. Rüegg, T. Ehlers, M. Spönemann, and R. von Hanxleden. A generalization of the directed graph layering problem. In *Proc. GD '16*, 2016. `doi:10.1007/978-3-319-50106-2_16`.

8 U. Rüegg, C. D. Schulze, J. J. Carstens, and R. von Hanxleden. Size- and port-aware horizontal node coordinate assignment. In *Proc. GD '15*, 2015. `doi:10.1007/978-3-319-27261-0_12`.

9 U. Rüegg, C. D. Schulze, C. Sprung, N. Wechselberg, and R. von Hanxleden. Edge bundling for dataflow diagrams. Poster at GD '16, 2016.

10 M. Spönemann, H. Fuhrmann, R. von Hanxleden, and P. Mutzel. Port constraints in hierarchical layout of data flow diagrams. In *Proc. GD '09*, 2010. `doi:10.1007/978-3-642-11805-0`.

11 K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst. Man. Cybern.*, 1981. `doi:10.1109/TSMC.1981.4308636`.

[4] `https://npmtrends.com/elkjs`
[5] `https://app.gitter.im/#/room/#eclipse_elk:gitter.im`
[6] `https://rtsys.informatik.uni-kiel.de/elklive/index.html`