yFiles – From Data to Meaningful Visualizations

Evmorfia Argyriou
yWorks GmbH, Tübingen, Germany

Benjamin Niedermann
yWorks GmbH, Tübingen, Germany

Abstract

Graph visualizations help with complex data analysis but often require expert knowledge to apply and configure advanced algorithms. yFiles, a diagramming SDK, bridges this gap by enabling developers to create interactive visualizations easily. This work demonstrates how yFiles helps transform raw data into accessible graph visualizations.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases diagramming SDK, layout algorithms, interactive graph visualization

Digital Object Identifier 10.4230/LIPIcs.GD.2024.55

Category Software Abstract

1 yFiles – From Data to Graph Visualizations

Graph visualizations help users gain insights from complex data. Depending on the structure of the data, different methods can be applied to place the graph nodes and route the edges. For instance, hierarchical data is well represented with layered drawings, whereas social network data often benefits from force-directed layouts. Research provides powerful algorithms for drawing graphs [1, 4], but their application generally requires expert knowledge. Meanwhile, many users outside the field of graph drawing seek simple, effective tools to visualize their data without the need to understand the technical details. This work shows how yFiles[1], a diagramming SDK, bridges the gap between sophisticated algorithms and the practical needs of users. We demonstrate how this library can transform raw data into interactive graph visualizations accessible to anyone without needing to "reinvent the wheel".

Challenge 1: From Data to Graph Structures. In typical use cases, users of the diagramming SDK aim to answer questions based on structured data. The first step involves understanding the data structure and identifying specific characteristics. yFiles offers a range of graph analysis algorithms, including k-means or k-core clustering, and centrality algorithms, to help users understand the graph's structure. This step is crucial for selecting the most appropriate layout algorithm; see Fig. 1 for examples.

Challenge 2: From Graph Structures to Layouts. Once the data structure is understood, users select a specific layout that best fits the graph structure, for instance, a layered [5] or force-based layout [2]. However, most users are not experts in graph drawing and prefer to configure layouts using visual parameters rather than technical ones, such as force strengths. Visual parameters might include node distances, node sizes, edge thickness/bends, labels, or more sophisticated constraints such as group nodes, node ordering, layer assignment, or port constraints. yFiles maps these visual settings onto appropriate algorithms and their parameters, making advanced layout techniques accessible without requiring users to understand the underlying details. For example, the organic layout consists of several force-based algorithms that are applied depending on the graph structure [2, 3].

[1] `https://yfiles.com`

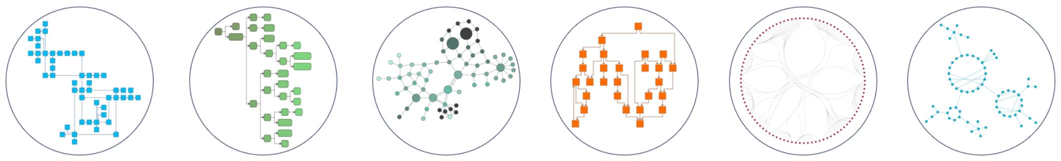

Figure 1 Examples of layouts offered by yFiles. From left to right: orthogonal, tree, organic (force-based), hierarchic (layered drawing), single cycle, circular layout.

Adapting algorithms from research to complex use cases, regardless of graph size or constraints, is a significant challenge. Among others, yFiles addresses this by allowing the combination of different layout algorithms. For instance, a graph with group nodes might use a force-based method for top-level nodes and a layered layout for the groups' contents. yFiles provides a powerful pipeline mechanism for assembling various layouts using both built-in and user-defined algorithms. The SDK also supports customizing layout algorithms with a wide range of configuration options. Additionally, it offers features such as automatic substructure detection and enhanced visualization of elements such as cycles, stars, and parallel structures. Moreover, yFiles strongly emphasizes generic and common features, such as labeling, group nodes, edge grouping, restricted port locations, and table-like structures, all of which are supported out of the box.

Challenge 3: From Layouts to Network Visualizations. Beyond arranging graph elements, their visual representation – including colors, shapes, and edge thickness – is crucial as these visual aspects encode additional information. yFiles offers a variety of predefined styles for graph elements and also allows users to create custom styles. The SDK integrates seamlessly with various rendering engines, such as SVG or WebGL, depending on the platform.

All layout algorithms can be applied to graphs with animations, helping users understand transitions between different visualizations. This is facilitated by a generic animation framework and morphing algorithms.

Challenge 4: Making the Graph Interactive and Dynamic. Interactivity allows users to explore and edit the graph structure and the underlying data. Common interactions include navigation (zooming and panning), basic modifications, and more complex actions such as collapsing/expanding group nodes and filtering elements. These require layout algorithms that can adapt to changes, which yFiles supports through features such as fixed node subsets and incremental element placement.

Conclusion

yFiles bridges the gap between formal algorithms and the practical needs of users. By providing advanced layout algorithms and integration with existing frameworks, it enables users to create not only visually appealing but also meaningful visualizations of their data.

References

1 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall PTR, USA, 1st edition, 1998.

2 Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991. doi:10.1002/SPE.4380211102.

3 Tomihisa Kamada and Satoru Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989. doi:10.1016/0020-0190(89)90102-6.

4 Michael Kaufmann and Dorothea Wagner, editors. *Drawing graphs: methods and models*. Springer-Verlag, Berlin, Heidelberg, 2001.

5 Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11:109–125, 1981. URL: https://api.semanticscholar.org/CorpusID:8367756, doi:10.1109/TSMC.1981.4308636.