Graph Visualization

U. Doğrusöz
and
G. Sander

Tom Sawyer Software, 804 Hearst Avenue, Berkeley, CA 94710, USA
info@tomsawyer.com

Graph drawing, or layout, is the positioning of nodes (objects) and the routing of edges (relations) in a graph in order to produce an aesthetically pleasing, comprehensible drawing of the underlying data. Our objective has been to create practical graph visualization technology. Graph drawing algorithms have been studied and improved to meet industry requirements for generality, efficiency, breadth, and extendibility. In addition, new algorithms for problems such as incremental and constrained layout and complexity management have been designed to facilitate more flexible visualization applications.

Categories and Subject Descriptors: D.2.13 [**Software Engineering**]: Reusable Software; G.2.2 [**Graph Theory**]: Miscellaneous—*Graph Algorithms*; I.3.m [**Computer Graphics**]: Miscellaneous

General Terms: Component Technology, Graph Visualization

Additional Key Words and Phrases: Graph layout, Class libraries, Graphical user interface

1. INTRODUCTION

As graphical user interfaces (GUI) have improved and more state-of-the-art software tools have incorporated visual components, graph visualization, specifically automatic graph layout, has become crucial especially in areas such as network management, database design, software engineering, and web design and management.

Graph layout comes in different flavors depending on the application type and the data to be visualized, ranging from trees (e.g., directory structure) to directed graphs (e.g., PERT chart) to general graphs (e.g., network map), from straight-line drawings to orthogonal drawings (e.g., database schema). For instance, the layout style and requirements of a flowchart are very different from those of a database

Research supported in part by NIST, Advanced Technology Program grant number 70NANB5H1162.

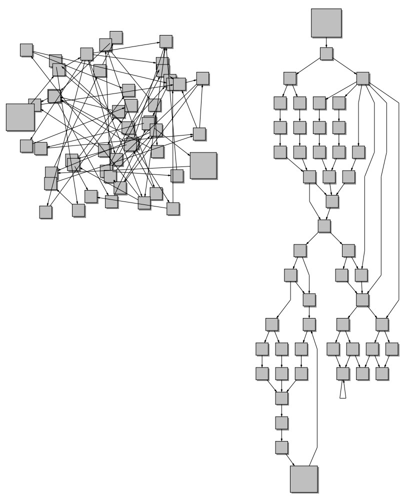

Fig. 1. The same graph before and after layout.

schema.

Even though graph drawing aesthetics are subjective and may need to be tailored to suit personal preferences, several criteria such as avoiding crossings and bends in edges, displaying any symmetry present in the structure of the graph, and minimizing the total drawing area are commonly accepted. Unfortunately, for general graphs the optimization problems associated with most of these criteria are among the intractable (NP-hard) problems of graph theory [Di Battista et al. 1994].

2. THEORY, PRACTICE, AND CHALLENGES

Over the past decade or so, there has been a significant amount of research in graph drawing theory. An annual symposium on graph drawing has been held for researchers, practitioners, and users working on different aspects of graph drawing for the past four years [Tamassia et al. 1997], and many graph layout and editing systems have been developed in universities and research institutions [Di Battista et al. 1994].

As is the case with most other areas, there has been a significant gap between the theory and practice of graph drawing. Not only are most theoretical results for *restricted* classes of graphs (e.g., the graph is assumed to be planar or not to have any nodes of degree higher than four) and do not apply to the majority of real-life graphs, but many of the most challenging problems resulting from practical applications of graph drawing theory have not been sufficiently studied.

Tom Sawyer Software has been working towards bridging this gap. The *Graph Layout Toolkit* (GLT) [tss 1997b] and *Graph Editor Toolkit* (GET) [tss 1997a] of Tom Sawyer Software are graph layout, display, and editing libraries that facilitate easy integration and customization of GUI programs for development of industrial graph visualization tools.

The graph drawing algorithms included in GLT and GET have been developed

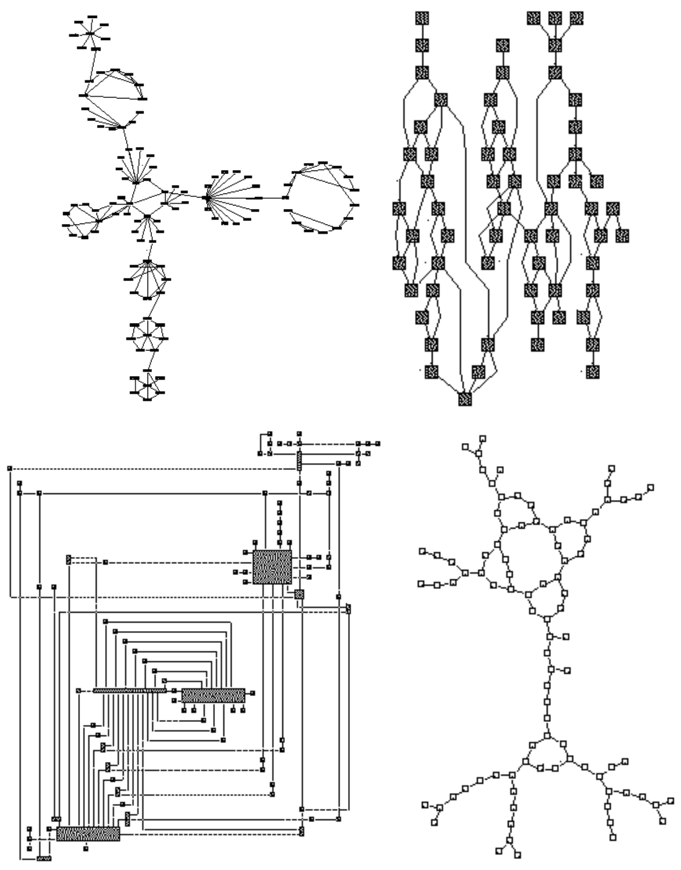

Fig. 2. Examples of graphs drawn in circular, hierarchical, orthogonal, and symmetric styles.

with the following main objectives:

—*Generality:* Solutions should not place any restrictions on the type of graphs.

—*Efficiency:* Solutions should be both space and time efficient since majority of applications use graph visualization in an interactive manner.

—*Breadth:* Requirements of various industries should be met.

—*Portability and Extendibility:* Solutions should be as portable and customizable as possible.

GLT provides a graph model and a drawing framework and offers four different layout styles: *circular*, *hierarchical*, *orthogonal*, and *symmetric*. Each of these styles addresses the needs of different software applications. GLT is independent of any display or graphics software, thereby providing users with considerable design flexibility. GET extends GLT to support rapid and efficient GUI application development. With this extendible software, diagram display and editing functionality such as zooming, scrolling, bitmap drawing, and mouse handling are provided.

The following sections summarize what we believe to be crucial for successful industrial graph visualization tools. Note that most of these issues are inter-related and the solution to one might depend on the others. Furthermore, solutions to many of these issues depend heavily on the underlying layout style and technique used to solve them. There has been limited research by the graph drawing community on most of these problems. Though we have made significant advances in these areas, there is still substantial room for improvement.

2.1 Incremental graph layout:

It is crucial to preserve a "mental picture" of the drawing of a graph over successive layouts. It can be distracting to make a slight modification, perform a layout, and have the resulting drawing appear very different from the previous drawing.

We have developed incremental layout algorithms for our Symmetric and Hierarchical libraries, and are working on supporting the same for other libraries.

2.2 Complexity management and compound graphs:

Often times a single graph is not sufficient to represent information due to the graph's overwhelming size or semantic limitations. Such information may be organized to span several graphs with relations among them. The two most common types of such relations are *inter-graph edges*, which connect nodes in different graphs, and *navigation*, where a node or an edge in one graph represents another graph. When such relations are part of the information to be drawn, layout algorithms need to respect inter-graph edges and provide techniques for routing them as well as facilitating techniques that allow nesting of graphs resulting from navigation relations.

Our *navigation manager* facilitates the creation of navigation links between nodes (or edges) and graphs, and allows nesting of graphs through an *expand* and *collapse* mechanism. In addition, *hiding* and *folding* operations let users selectively reduce the amount of information to be visualized. Furthermore, we have been working on automatic graph partitioning techniques for very large graphs with Dr. Arunabha Sen's group at Arizona State University.

2.3 Constraints:

Even though most of the information to be drawn is "logical," many applications enforce certain placement requirements on nodes. These requirements range from fixing one or both the coordinates of locations of some objects to "clustering" a specified group of objects.

Limited support for constraints, such as restricting a node to a specific layer in a layered hierarchical drawing, is available with GLT. We are also working on a more generalized constraint framework that can be used in all of the layout libraries.

2.4 Complex shaped nodes and attachment points:

Certain applications require support for a sophisticated clipping model where an edge connects to a node at a certain location, or port, along one of its sides. For instance, an edge might be associated with only one of several rows in a table node. In this instance, the edge must be clipped to the node at the port adjacent to the appropriate row in order to convey the information accurately. In other cases, the shape of a node is rather irregular (e.g., an *OR* gate in a logical diagram) and requires complex clipping techniques.

In GLT, nodes can have in and out (or left, right, top, and bottom) ports to which edges can attach. We are working on a more general attachment point mechanism as well as a better formalization of logical ports (e.g., a pin on an electrical component whose position is not fixed) versus physical ports (attachment points).

2.5 Labeling:

Labels are visual descriptions, such as text or icons, that are assigned to nodes and edges. Just as layout of nodes and edges in a graph is a time-consuming and monotonous task, so is the positioning of labels. Therefore, algorithms for automatic positioning of labels. The label positioning objectives include the elimination of ambiguity (which node or edge a label is associated with) and flexible placement constraints (i.e., whether a label is associated with the center or a specific end of a long edge).

With Dr. Ioannis G. Tollis' group at the University of Texas at Dallas, we have developed automatic label positioning algorithms that support multiple labels per edge. The research for a more general automatic placement algorithm for labels including node labels is in progress.

3. CONCLUSION

The techniques we have developed and implemented [tss 1997b] for many challenging problems of graph drawing including incremental and constrained layout and complexity management techniques have been well-received by software developers using our graph visualization tools. We continue to research and improve our graph visualization algorithms, and believe that there continues to be room for improvement and advances in the area.

REFERENCES

1992-1997b. Graph Layout Toolkit User's Guide and Reference Manual. Tom Sawyer Software, Berkeley, CA, USA.

1997a. Graph Editor Toolkit User's Guide and Reference Manual. Tom Sawyer Software, Berkeley, CA, USA.

Di Battista, G., Eades, P., Tamassia, R., and Tollis, I. G. 1994. Algorithms for drawing graphs: an annotated bibliography. *Comput. Geom. Theory Appl. 4*, 235–282.

Tamassia, R., Tollis, I. G., Brandenburg, F. J., North, S., and Di Battista, G. Eds. 1994-1997. *Symposium on Graph Drawing*, Volume 894, 1097, 1190, and 1353 of *Lecture Notes in Computer Science* (1994-1997). Springer-Verlag.