Network Visualization Survey

Ales Komarek, Jakub Pavlik, and Vladimir Sobeslav(✉)

Faculty of Informatics and Management, University of Hradec Kralove,
Rokitanskeho 62, Hradec Kralove, Czech Republic
{ales.komarek,jakub.pavlik.7,vladimir.sobeslav}@uhk.cz
http://uhk.cz/

Abstract. This paper compares modern approaches to draw complex graph data that create create compelling visualizations. Graphs are used to represent more and more complex systems that are used across various scientific domains. Some graph visualizations are used to model network topologies or service architectures in information service fields, model genomes in biomedicine or complex molecules in chemistry. Visualizations of graph data play important role in interpreting the meaning of graphed data. The adage *"A picture is worth a thousand words"* generally refers to the notion that a complex idea can be conveyed with just a single still image and that is actually the main reason for creating visualisations of any kind. Graph drawings from all domains follow the same rules covered by the graph theory. This work outlines different layout strategies used for drawing graph data. Tested graph drawing layouts are compared by standard quality measures to determine their suitability for various areas of usage.

Keywords: Graph · Graph theory · Data flow · Visualization

1 Introduction

Growing number of systems are using various visualization techniques to display data it holds. Business intelligence systems use graphs to simplify data and graph drawings are interface for the users to better comprehend the data. [2,5,14] This paper's goal is to present and compare traditional and modern visualisation techniques and determine what kind of graph data it can represent. Main areas of focus for graph drawings are following knowledge domains.

1. Network topologies with flows
2. Service oriented architectures

The section 2 of the paper summarizes graph theory therms and principles relevant to the graph drawing. [1] This chapter also lists quality measures that are used for graph layout quality evaluations.

Chapter 3 introduces various graph drawing layouts that can be used for actual graph visualisations. Covered visualisation techniques are traditional force-directed algorithms, adjacency matrices or arc diagrams, [7,12] as well

M. Núñez et al. (Eds.): ICCCI 2015, Part II, LNCS 9330, pp. 275–284, 2015.
DOI: 10.1007/978-3-319-24306-1_27

as new approaches to graph layouts like hive plots, chord diagrams, Sankey diagrams and Pivot graphs. [8,9,13] Adjacency matrix is special as it displays graph edges as the graph's matrix coordinates instead of lines as the other graph drawings.

Following chapter shows differences of these visualisation techniques by given parameters. All of the visualisations were tested with samples of graphs data. The D3 [14] graphical library was used to create real graph drawings from the same dataset. D3 is being developed and improved by Mike Bostock and uses document driven approach to visualize any kind of information. Some graph drawings require additional data which had to be provided to allow test of all drawings. The last chapter 5 summarises the usability of selected visualization techniques.

2 Graph Drawing Theory

This section describes important therms from graph theory used for graph visualizations. Graph theory is an area of mathematical research with a large specialized vocabulary. Some authors use the same word with different meanings. Some authors use different words to mean the same thing. [1,7] A drawing of a graph or network diagram is a pictorial representation of the vertices and edges of a given graph. This drawing should not be confused with the graph itself, very different layouts can correspond to the same graph. In the abstract, all that matters is which pairs of vertices are connected by edges. In the concrete, however, the arrangement of these vertices and edges within a drawing affects its understandability, usability and aesthetics [1,3]. Some graphs even change over time by adding and deleting edges (dynamic graph drawing). Their goal is to preserve the user's mental map over time. This paper focus are solely static graph drawings

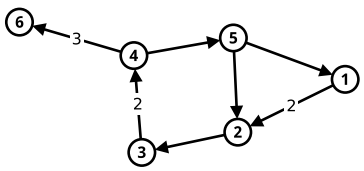

Fig. 1. Simple Graph Example

Figure 1 shows example graph with several vertices and weighted edges which is used further in this chapter to demonstrate some graph theory principles important for creating visualizations.

2.1 Adjacency and Degree

Degree or valency of a vertex is usually a measure of immediate adjacency. An edge connects two vertices. these two vertices are said to be incident to

that edge, or, equivalently, that edge incident to those two vertices. All degree-related concepts have to do with adjacency or incidence. A vertex of degree 0 is an isolated vertex. A vertex of degree 1 is a leaf.

In the simple graph example, vertices 1 and 3 have a degree of 2, vertices 2, 4 and 5 have a degree of 3, and vertex 6 has a degree of 1. If number of edges is finite, then the total sum of vertex degrees is equal to twice the number of edges.

2.2 Weighted Graphs

A weighted graph associates a certain weight with every edge in the graph. Weights are usually real numbers. Weights may be restricted to rational numbers or integers. The weight of a path or the weight of a tree in a weighted graph is the sum of the weights of the selected edges. Sometimes a non-edge (a vertex pair with no connecting edge) is indicated by labeling it with a special weight representing infinity. Sometimes the word cost is used instead of weight. When stated without any qualification, a graph is always assumed to be unweighted.

In some writing on graph theory the term network is a synonym for a weighted graph. A network may be directed or undirected, it may contain special vertices (nodes), such as source or sink. The classical network problems include:

1. minimum cost spanning tree
2. shortest paths
3. maximal flow (and the max-flow min-cut theorem)

2.3 Direction

Directed edge is an ordered pair of vertices that can be represented graphically as an arrow drawn between these vertices. In such an ordered pair the first vertex is called the initial vertex or tail; the second one is called the terminal vertex or head (because it appears at the arrow head). An undirected edge disregards any sense of direction and treats vertices on both sides interchangeably. A loop in a digraph, however, keeps a sense of direction and treats both head and tail identically.

A set of arcs are multiple, or parallel, if they share the same head and the same tail. A pair of arcs are anti-parallel if one's head/tail is the other's tail/head. A digraph, or directed graph, or oriented graph, is analogous to an undirected graph except that it contains only arcs. A mixed graph may contain both directed and undirected edges; it generalizes both directed and undirected graphs. When stated without any qualification, a graph is almost always assumed to be undirected.

A digraph is called simple if it has no loops and at most one arc between any pair of vertices. When stated without any qualification, a digraph is usually assumed to be simple. A quiver is a directed graph which is specifically allowed, but not required, to have loops and more than one arc between any pair of vertices.

2.4 Quality Measures of Graph Drawings

Many different quality measures have been defined for graph drawings, in an attempt to find objective means of evaluating their aesthetics and usability [1,3,4]. In addition to guiding the choice between different layout methods for the same graph, some layout methods attempt to directly optimize these measures.

1. The crossing number of a drawing is the number of pairs of edges that cross each other. If the graph is planar, then it is often convenient to draw it without any edge intersections; that is, in this case, a graph drawing represents a graph embedding.
2. The area of a drawing is the size of its smallest bounding box, relative to the closest distance between any two vertices. Drawings with smaller area are generally preferable to those with larger area, because they allow the features of the drawing to be shown at greater size and therefore more legibly. The aspect ratio of the bounding box may also be important.
3. Symmetry display is the problem of finding symmetry groups within a given graph [4], and finding a drawing that displays as much of the symmetry as possible. Some layout methods automatically lead to symmetric drawings; alternatively, some drawing methods start by finding symmetries in the input graph and using them to construct a drawing.
4. It is important that edges have shapes that are as simple as possible, to make it easier for the eye to follow them. [1] In polyline drawings, the complexity of an edge may be measured by its number of bends, and many methods aim to provide drawings with few total bends or few bends per edge. Similarly for spline curves the complexity of an edge may be measured by the number of control points on the edge.
5. Angular resolution is a measure of the sharpest angles in a graph drawing. If a graph has vertices with high degree then it necessarily will have small angular resolution, but the angular resolution can be bounded below by a function of the degree.

3 Graph Visualization Layouts

This section describes traditional layouts strategies [7,10] as well as modern approaches [8,9,12,13] that are used to display complex graph data. Many of these layouts originate at different knowledge domains where they solve specific problems. These layouts can be used to display any kind of relational data from other domains as well.

3.1 Force Directed Graphs

Force-directed graph drawings are a family of algorithms that position the nodes of a graph in two-dimensional space so that all the edges are of more or less equal length and there are as few crossing edges as possible by assigning forces among the set of edges and the set of nodes based on their relative positions [7]. These

forces are then used either to simulate the motion of the edges and nodes or to minimize their energy. Force-directed algorithms calculate the layout of a graph using only in formation contained within the structure of the graph itself, rather than relying on domain-specific knowledge.

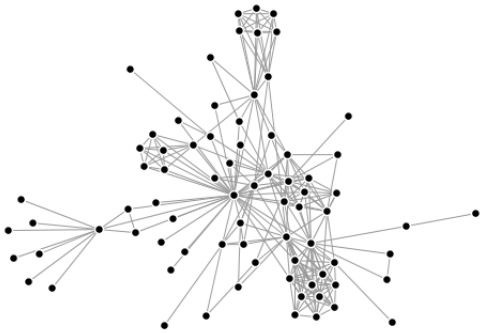

Fig. 2. Force-directed Graph

3.2 Hive Plots

In hive plot vertices are mapped to and positioned on radially distributed linear axes [8]. This mapping is based on network structural properties. Edges are drawn as curved links. The purpose of the hive plot is to establish a new baseline for visualization of large networks. A method that is both general and tunable and useful as a starting point in visually exploring network structure.

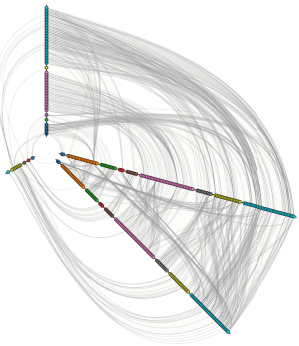

Fig. 3. Hive Plot

3.3 ARC Diagrams

An arc diagram uses a one-dimensional layout of vertices, with circular arcs to represent edges. Though an arc diagram may not convey the overall structure of the graph as effectively as a two-dimensional layout, with a good ordering of vertices it is easy to identify cliques and bridges. Further, as with the indented tree, multivariate data can easily be displayed alongside nodes.

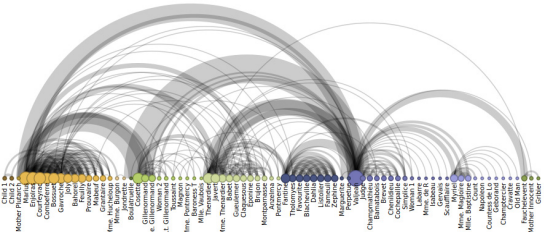

Fig. 4. Arc Diagram

3.4 Sankey Flow Diagrams

Sankey diagram is a specific type of flow diagram, where the width of the arrows is proportional to the flow quantity. [10] Sankey diagrams typically used to visualize energy or material or cost transfers between processes or the network flows [11].

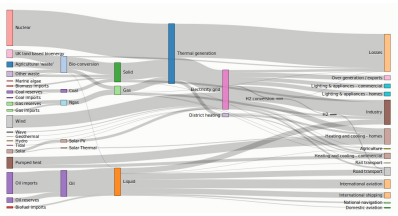

Fig. 5. Sankey Flow Diagram

3.5 Chord Diagrams

In chord diagram the vertices are arranged radially around a circle with the edges between them typically drawn as arcs. Chord diagrams are useful for exploring relationships between groups of entities. They have been heavily adopted by the biological scientific community for visualizing genomic data, but they have also been featured as part of infographics in numerous publications including Wired, New York Times, and American Scientist.

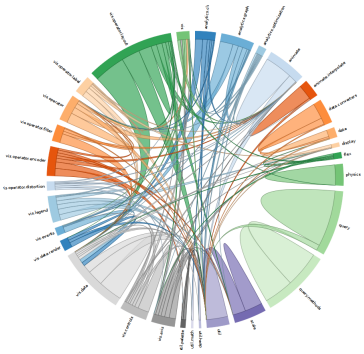

Fig. 6. Chord Diagram

3.6 Hierarchical Edge Bundles

While a small amount of data could be represented in a circular diagram using straight lines to show the interconnections, a diagram featuring numerous lines would quickly become illegible. To reduce the visual complexity, chord diagrams employ a technique called hierarchical edge bundling. There is a large class of data sets that contain both hierarchical components, i.e., parent-child relations between data items, as well as non-hierarchical components representing additional relations between data items. [9] If there is need for more insight in the hierarchical organization of graphed data it can be visualized by the hierarchical structure using any of the tree visualization methods. However, if we want to visualize additional adjacency edges on top of this by adding edges in a straightforward way, this generally leads to visual clutter. A possible way to alleviate this problem is to treat the tree and the adjacency graph as a single graph.

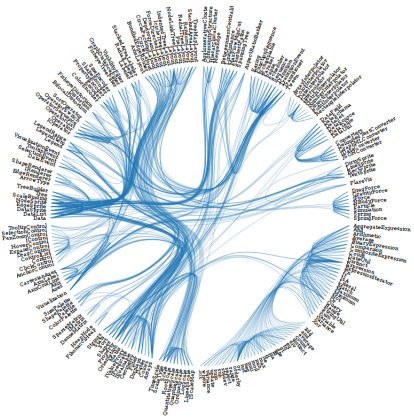

Fig. 7. Hierarchical Edge Bundle

3.7 Adjacency Matrices

An adjacency matrix represents a graph, where each value in row i and column j in the matrix corresponds to the edge from vertex i to vertex j in the graph. [12] Use of colors or saturation instead of text allows patterns to be perceived more easily. The effectiveness of a matrix visualization is heavily dependent on the order of rows and columns. If related vertices are placed closed to each other, it is easier to identify clusters.

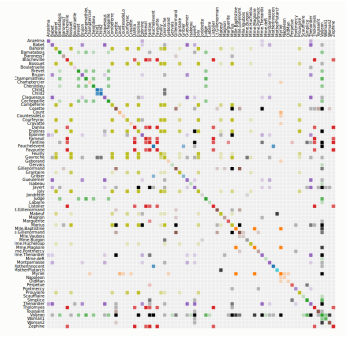

Fig. 8. Adjacency Matrix

3.8 Pivot Graphs

Pivot graph is technique for visualizing and analysing graph structures. [13] The technique is designed specifically for graphs that are multivariate, where each node is associated with several attributes. Unlike visualizations which emphasize global graph topology, pivot graph uses a simple grid-based approach to focus on the relationship between node attributes and connections.

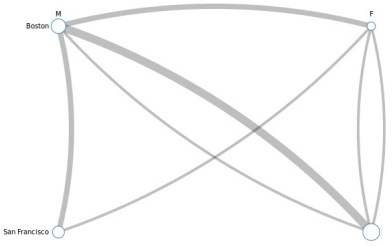

Fig. 9. Pivot Graph

4 Visualization Comparison

This section summarises some of the qualitative properties of different graph layouts. All parameters for evaluating graphs are in the following list.

Requires Domain Specific Data. This property states whether graph layout requires additional data to get displayed.
Maximum Number of Vertices. Maximum number of vertices where visualisation remains aesthetic and readable.
Maximum Number of Edges. Maximum number of displayed graph edges where visualisation remains aesthetic and readable.
Amount of Edge Crossings. How many edge crossings does the graph layout produce.
Multivariate Data Support. Does the graph layout support display of additional information.
Global Graph Topology. This property shows if drawing covers whole topology or just specific subgraphs.

All of these parameters were evaluated, tested agains a d3js graph implementation. The results of the tests are summarised in the table 1.

Table 1. Graph Drawing Layouts Comparison

Layout	Domain	Vertices	Edges	Crossings	Multivariete	Global topo.
Force-directed	no	1.000s	2.000s	high	no	yes
Hive plot	yes	2.000s+	5.000s+	medium	yes	yes
Adjacency matrix	no	100s	1.000s	no	yes	yes
Arc diagram	no	100s	1.000s	low	no	yes
Sankey diagram	no	100s	1.000s	low	yes	yes
Chord diagrams	yes	100s	1.000s	medium	yes	both
Pivot graph	yes	10s	10s	low	yes	no

Some values are estimates as the largest graph data set contained 2000 vertices and 5000 edges. Force directed graphs started to exhibit a hairball effect with growing number of vertices [8]. Other layouts did not loose understandability in such a rate, but other reasons prevented to get larger.

5 Conclusion

We have covered modern layout methods used for graph drawing in our paper. Each layout has it's area of use. Some visualisations like arc diagram and chord diagram display edge weights and directions better than others. Some visualisations like hive plots or hierarchical edge bundles require additional domain specific knowledge definitions that make them harder to maintain but provide more aesthetic and usable layouts for datasets where traditional layout approaches fail.

Graph layouts are located at http://lab.newt.cz/vis/relational/. The graphs are implemented in D3 javascript framework for data driven graphics.

References

1. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.: Algorithms for Drawing Graphs: an Annotated Bibliography. ACM (1994). doi:10.1016/0925-7721(94)00014-x
2. Graph, H.: Graph visualization and navigation in information visualization: A survey. Visualization and Computer Graphics. IEEE doi:10.1109/2945.841119
3. Purchase, H., Carrington, D., Allder, J.-A.: Empirical Evaluation of Aesthetics-based Graph Layout. Empirical Software Engineering. Kluwer (2002)
4. McGrath, C., Blythe, J., Krackhardt, D.: The effect of spatial arrangement on judgements and errors in interpreting graphs. Social Networks **19**. Elsevier (1997)
5. Purchase, H.C., Hoggan, E., Görg, C.: How important is the mental map? – an empirical investigation of a dynamic graph layout algorithm. In: Kaufmann, M., Wagner, D. (eds.) GD 2006. LNCS, vol. 4372, pp. 184–195. Springer, Heidelberg (2007)
6. Di Battista, G., Gargb, A., Liottab, G., Tamassiab, R. ,Tassinaric, E. Vargiuc, F.: An experimental comparison of four graph drawing algorithms. Elsevier (1997). doi:10.1016/S0925-7721(96)00005-3
7. Kobourov, S.: Handbook of Graph Drawing and Visualization: Force-directed drawing algorithms (2013). http://cs.brown.edu/rt/gdhandbook/chapters/force-directed.pdf
8. Krzywinski, M.: Hive Plots - Linear Layout for Network Visualization - Visually Interpreting Network Structure and Content Made Possible (2011). http://www.hiveplot.net/
9. Holten, D.: Hierarchical, Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data (2006)
10. Riehmann, P., Hanfler, M., Froehlich, B.: Interactive Sankey Diagrams (2005)
11. Holik, F., Horalek, J., Marik, O., Neradova, S., Zitta, S.: The methodology of measuring throughput of a wireless network. In: CINTI 2014 (2014)
12. Godsil, C., Royle, G.: Algebraic Graph Theory. Springer (2001). ISBN 0-387-95241-1
13. Wattenberg, M.: Visual Exploration of Multivariate Graphs. ACM (2006). ISBN 1-59593-372-7
14. Bostock, M.: D3 Data-Driven Documents Documentations (2015). https://github.com/mbostock/d3/wiki