

VISUALIZATION OF NETWORK STRUCTURES

Aaron Kershenbaum
IBM T.J. Watson Research Center
Yorktown Heights, NY, 10598
aaronk@us.ibm.com

Keitha Murray
Iona College
New Rochelle, NY, 10801
kmurray@iona.edu

ABSTRACT

Visualization of large networks, especially with annotations describing node and edge properties, is a challenging task. The challenge is heightened when it is necessary for the user to have both global and local views of the network. While there is no general solution to this problem, it can be successfully approached in several ways. By taking advantage of hierarchical structure that exists in the network and by judiciously using color and symbols to explicitly represent node and edge properties, it is often possible to convey the necessary information. We explore these techniques and their application to the visualization of networks.

1 INTRODUCTION AND PROBLEM DESCRIPTION

Networks have long been recognized as effective tool for visualizing relationships, both qualitative and quantitative, among objects [1, 2, 3, 4, 10]. By a network, we mean a graph whose nodes and edges are annotated by properties specific to the topic at hand. For example, in chemistry, the network might be a diagram of a molecule where the nodes are atoms, annotated to indicate the type of atom, and the edges are bonds, annotated to indicate the type of bond (such as covalent). In a management course, the network might be a PERT chart, with the nodes indicating interdependent tasks, annotated to identify them and their durations, with directed edges representing the dependencies (e.g. task C can only begin after tasks A and B are completed). Networks are widely used in computer science to illustrate data structures, program component interdependencies, object relationships, parsing strategies, syntactic structures, hardware architecture, control flow and data flow. Indeed, almost any course could benefit from the use of networks as tools for visualization.

Although anyone can draw a network and many software tools exist to assist in this task [3, 5, 6, 7, 8], significant problems remain. The first and most obvious problem is that if the network is large, it is difficult to present it in a way that can actually be seen, especially when the number of pixels on a screen is limited.

Other more subtle problems exist for such tools [2, 4, 10]. For example, for certain applications, the need for precision might necessitate annotating an edge

with a numerical length to 5 decimal places. However, such annotation occupies a lot of screen space and may obscure a more important fact that some edges are “long” while others are “short.” Thus, assigning colors to different edge length categories may be the better approach when the precise length is not important or where it can be obtained in another way, e.g., by temporarily opening up a box with additional annotation when the mouse hovers over a node.

The notion of standardization of software tools is also a consideration in the setting of preparing classroom materials. The instructor would prefer using simple tools to prepare lectures and would like to be able to easily share materials across courses. Uniformity in presentation also makes it easier for the student to understand the material being presented. This is not to say that all material can or should be presented in the same way, but rather, that by preparing visual materials within a unified framework and using a unified set of authoring tools, the task becomes more efficient and effective. Even if everyone uses their favorite tools, an understanding of the principles for visualizing networks is useful.

This paper explores the basic issues in representing networks, the underlying principles for their display and some of the tools currently available for creating and presenting network displays.

2 MAPS - AN ILLUSTRATIVE EXAMPLE

To serve as an example of some of the issues that arise in representing networks and some of the approaches to dealing with these issues, we begin by considering the problem of presenting maps; i.e., visualizations of geographic areas. The first question is, “What is the purpose of displaying the map?” The answer will help us to decide how best to display the map. Suppose the purpose is to drive from one place to another. Mapquest (www.mapquest.com) is a common application which deals with this problem and their approach is instructive. The underlying network of streets has millions of nodes and edges. Thus, it is impossible to display it all at once. On the other hand, just displaying the actual driving route is not satisfactory either as the driver also usually wants some perspective on where the route is in relation to other familiar roads. This perspective varies, however, based on the type of the road. When on a highway, details of local roads are relatively unimportant. The display solution to these dual needs is a hierarchical display, precisely what Mapquest uses. One can zoom and pan to see different parts of the route, and at different scales, different levels of detail are presented. The user can get all the detail desired, along with the overview of the route, but not all at the same time.

Several things should be observed. First, the problem of displaying the entire network at once with all its details is not possible. Fortunately, it is not necessary to do so. Second, since the basic purpose of doing the display is to give the driver perspective, it is not sufficient to just pan around; at some point it is necessary to see the entire route from end to end. Fortunately, the network is hierarchical and can be displayed globally at each level in the hierarchy at appropriate levels of detail. At different points in the trip, the driver may be interested in different levels of detail. It is possible that the driver may want to exit from a highway onto local streets in order to eat or rest. Thus, it is necessary to give the user control over the level of detail. So, the complete solution to this problem relies on taking advantage

of many things. Note that strictly speaking the map is not actually necessary at all since Mapquest gives verbal directions as well. This takes some pressure off of the graphical display and makes the problem more tractable. All in all, the overall solution is generally acceptable and Mapquest is widely used.

Now consider another map example, the New York City Subway map. Here too, the purpose of displaying the network is to assist people in getting from one place to another. But the display solution chosen is different in many ways from the one above. This network has many hundreds of nodes and edges, still too many to display on a computer screen, but it can, and is, shown in its entirety. This is done by printing a map at higher resolution than could be obtained on a computer screen and making the map large enough to hold the entire network. Nevertheless, the representation would be unreadable if it were necessary to draw the map to precise geographic scale. This condition is relaxed and the map is distorted somewhat to give congested areas more space. Most notably, Manhattan is enlarged in order to show the many subway lines passing through it more clearly. This is a reasonable compromise in this case as the user is not driving and does not need to know precise distances. "Distance" is essentially measured in subway stops, and the representation does well presenting these. One of the primary issues in network visualization is understanding to what extent the placement of the nodes contributes to (or impedes) the intent of the visualization. In many cases there is no "geography," but rather a logical relationship among the nodes.

If the subway system were an order of magnitude larger or if we wished to present it on a computer screen, which is an order of magnitude smaller than the printed map, we would have to add additional visualization functions to keep the representation intelligible. One possibility would be to pan about in a virtual window which is several times larger than the screen. Another solution would be to temporarily display only express stops, or display only transfer points along with a few other key stations such as the origin and destination of the trip. It would also be possible to display all the subway lines but only annotate the ones actually being used. As we remove detail in terms of stations, we might add simpler annotation such as the number of stops or estimated times, to keep perspective on the trip. User control would become more important in this instance.

A third example of a map is an airline route map. Here again, if we were to try to display all routes from all sources to all destinations by all carriers at once, the situation would be hopeless. In this case it would also be undesirable even if it were possible. It would be confusing to include flights that took place at substantially different times in a single display, for example a Tuesday flight from London to San Francisco alongside a Thursday flight from San Francisco to Honolulu. Likewise, in planning a trip from New York to San Francisco, it is unlikely that one would be interested in either of the flights above, at any time. Once we recognize that significant filtering of the map based on the user's goals is necessary, the problem becomes tractable again.

The actual representation of an airline route map is more interesting. Since there will be many parallel edges in this network, there is a need to represent edges by something other than straight lines. There are also several different types of annotation for the edges which may convey valuable information, including carrier, time, and price. Displaying all these at once on the map may result in

clutter. Color coding (e.g. for carrier), line type coding (e.g. dotted lines) to separate day flights from night flights, and line thickness to indicate cost level groupings, might help, but there is clearly a limit to how much could be done with such coding before the display became again unintelligible. User filtering and optional detailed display (e.g. in a text window), would probably be necessary.

This list of issues and approaches to network visualization is by no means exhaustive, but it starts to give the flavor of what needs to be considered. We will explore additional issues and techniques below.

3 NETWORK VISUALIZATION FUNCTIONS

Functionality required in a network visualization system will vary with the particular application and it is important to be able to trade off among requirements to do an acceptable job for a particular situation [1, 2]. In other words, we want to define a feasible problem in each case and then solve it as opposed to doing the best job possible failing at an intractable problem. Some of the functions that need to be considered when designing a network visualization tool are listed.

3.1 User control

It is very helpful to give the user as much control as possible over the display. The user is in the best position to make tradeoffs. On the other hand, the user does not want to be burdened with making complex decisions. Giving the user control over zooming and panning is good. It is easy to do (if the controls are implemented sensibly) and it takes burden off the system. The ability to relocate nodes to relieve congestion and make a more meaningful display can likewise be very useful to the user, but relying too much on this could place an overwhelming burden on the user. For large networks, the system should bear most of the responsibility in this regard. Giving the user control over the level of annotation, including enhanced annotation on a single node or edge can take a large burden off the system. Clearly, the user should have control over what is being displayed, where information being display will come from and where the displays will be stored for future use.

3.2 Node and edge selection

Many of the user control functions discussed above rely on the user being able to select specific nodes and edges. Most packages used to create display software include functions which allow the user to detect where the mouse is and to determine whether a button has been clicked or unclicked. Given this and the knowledge of where nodes and edges are on the screen, it is possible to select them and reposition them. The ability to select nodes and edges via menus or text input may be important, especially when the display is too crowded to make them visible or where panning has removed them from the current display.

3.3 Layout

The ability to position nodes in a way that makes the display readable and meaningful is central to network visualization [11]. Giving the user some manual

control over this is positive, but there is also a need to do so automatically, especially when the network is large. The first question is how does the location of the nodes relate to what is being visualized. In the map examples above, the actual location of the nodes is significant. The same is true if we were displaying anatomical or physical structures (e.g. 3D protein structure). In other situations, there is a logical relationship among the nodes which should be reflected in their location, e.g., we may be displaying an organizational hierarchy or semantic relationships in an ontology. There may be natural clustering within the nodes which may be significant to the visualization. In other cases, there is no particular significance to the location of the nodes and the primary objectives are readability and esthetics. Layout algorithms [1, 11] differ in each of these cases. A good visualization tool should include them all and put them under the user's control.

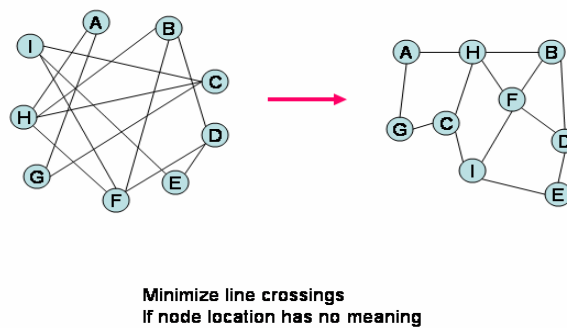


Figure 1 - Different layouts for the same graph

3.4 Annotation

Annotation on the nodes and edges gives meaning to the display. The appropriate amount and type of annotation is closely related to the purpose of the visualization. If the primary purpose is to simply present a logical relationship among the nodes (e.g. in presenting a data structure) then relatively little annotation is necessary. Another such example is the display of 3D protein structure where it is usually not possible to display any annotation if the entire protein is being displayed at once. In other cases, it is necessary to identify the nodes but not any other information about them. Sometimes, we need only present relative information, such as “big” and “small” or functional type. In these cases color, size and shape may be effective. Finally, it is sometimes necessary to display quantitative information, such as length. However, annotation takes space and thus we trade readability and scalability off against it. A good compromise, if possible, is to give the user control of the level of annotation. This includes globally turning annotation on and off and locally adjusting the level of annotation on a specific node or edge. It may be useful to devote a portion of the screen to annotation, and further, to allow panning through the annotation independently. A good example is to graphically display a program's control flow in one window while displaying its source code in another window.

3.5 Perspective

The ability to change perspective can help in clarifying the visualization, especially when space is limited [2]. Thus, one may center the display on a specific node. Another possibility is to change the scale locally in order to make an area more clearly visible, e.g., by opening a window with an inset. One may also want to shift back and forth between geographic and schematic perspectives correlating to the purpose of the display. In 3D displays, one might want to consider a 2D perspective. It is valuable to be able to change a perspective, e.g., by rotating what is being displayed.

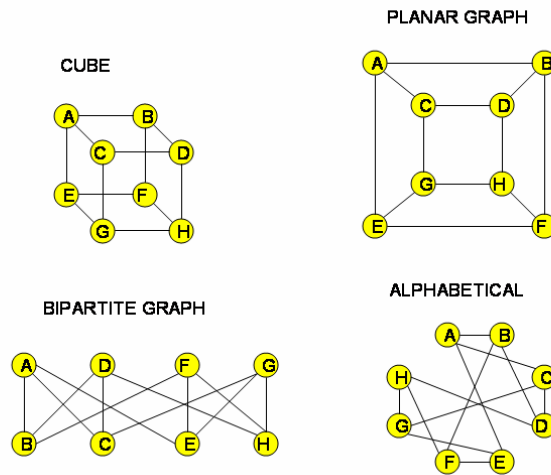


Figure 2 - Different perspectives of same graph

3.6 Hierarchy

If the network has a natural hierarchical structure, it is very useful to be able to reflect this in the display. Indeed, the entire purpose of the visualization may be to display this relationship. Thus, we may want to place nodes close in the hierarchy close to one another in the display. We may also want to zoom in and out hierarchically, both globally as in the mapping examples above and locally using insets. We may even wish to impose a hierarchy in order to increase the size of the network we can display, e.g. by temporarily compressing a group of nodes which are too close together to be simultaneously visible at the current scale. Using hierarchical compression may be a far better solution (indeed, maybe the only one) when the layout algorithm cannot produce a readable display. Simpler and more reliable layout algorithms can be used if compression is available [1].

3.7 Dynamic display

Much of the functionality discussed above implies the ability to dynamically alter the display under user control. This offers many advantages to the user and likewise decreases the requirements of the display software. More importantly, it allows the user to explore what is being shown, which is central to visualization. Given the memory and computing power available today and the presence of software libraries implementing interactive user controls, it is desirable to make the display dynamic and to provide a complete (if possibly simple) graphical user

interface (GUI). On the other hand, we might still want to do as much as possible automatically. For example, laying out a large network manually is at best tedious and at worst beyond most people's ability. In addition to responding dynamically to user inputs, visualization software may be integrated with other software, such as analysis algorithms, with the output of the analysis also being displayed graphically. Such software has the added challenge of altering not only the specific display and annotation from one analysis to another, but also possibly changing the layout and perspective, for example the "best" nodes may change for different input values of parameters. In some cases, animation may be appropriate, for example, in visualizing the progress of an algorithm or sequence of chemical reactions.

3.8 Decomposition

Another way of increasing the size of the network we can display is to decompose it and display one part at a time. Panning is the simplest example of this, where the decomposition is "geographic." It is also possible to decompose the network into clusters that are logically related (e.g. in displaying an organization chart), or functionally related (e.g. in displaying subsystems of a complex system). Decomposing into clusters is attractive when a natural basis for decomposition exists. Sometimes we may impose such a basis by only showing the "best" or most important nodes or edges.

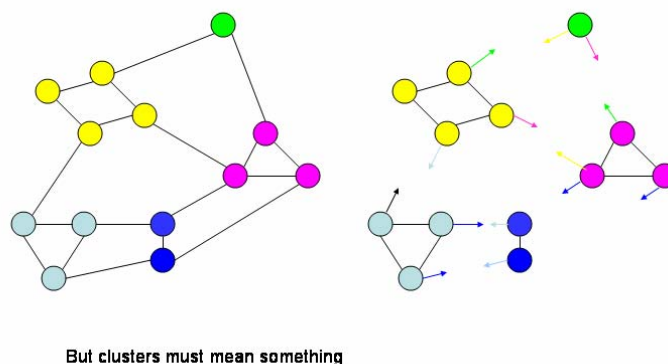


Figure 3 – Decomposition into clusters by color

3.9 Refinement by type of graph

Good visualization software can take advantage of the type of graph underlying the network. If it is known that the graph is a tree, the layout problem becomes much easier as there is no issue of crossing edges (unless other constraints create this problem). If the graph is acyclic, it is often possible (and possibly necessary) to lay nodes out in vertical layers. We can often increase scalability by taking advantage of restrictions on the type of graph. Good visualization software will do this without limiting its scope.

4 DESIGN AND IMPLEMENTATION OF NETWORK VISUALIZATION SOFTWARE

There are many issues in the design and implementation of network visualization software that involve tradeoffs. These include:

- The quality of the display produced
- Versatility across multiple applications
- The effort to create and maintain the software
- Ease of use

Perhaps counter intuitively, efficiency is not likely to be an issue because we will usually run out of display space before we run out of computer time or memory. The exceptions to this are complex layout algorithms and animation.

All of the above issues are important in design and implementation of network visualization software. There is a minimum quality required for intelligible display and the more challenging the visualization task (e.g. discriminating between similar objects), the more this issue is important. Similarly, if the software is too hard to use, it simply won't be used, or if it is, it won't be used reliably. This is particularly important in an academic setting where instructors will not be inclined to use any tool which impedes their ability to prepare lectures and which hinders students more than it helps them. The more general the applicability of the software, the more it will be used, resulting in increased reliability of the software, increased skill using it, and better understanding of the visual display because of common, repeated experience across applications. Nevertheless, requirements, quality and ease of use involve tradeoffs. To have more of one, we must accept less of another. The trick is not to trade a lot for a little; i.e., we seek first to satisfy the minimum requirements and then see how much more we can get. This sometimes forces us to limit the size and complexity of the visualizations.

A central question regarding the implementation of visualization software is whether to use:

- A complete, application specific visualization tool
- A general purpose visualization tool, possibly part of a more general authoring tool
- A toolkit of visualization functions on top of which an application is constructed, possibly in conjunction with other software which produces the data to be displayed
- Special purpose software modules

All except for the most specific application software will not require the last option, but any or combinations of the others may be of use. In preparing lectures, it is often not necessary, or even desirable, to present large, complex networks. Ease of use may be the most important consideration and a general purpose package may be best. Thus, the visualization tool that is part of the word processor, browser or authoring tool is often used to prepare the visual part of the lecture. Many of these tools offer great flexibility in the type of display and some offer animation. They also often offer the ability to include images in the

presentation, thereby affording the potential for high quality, specialized network visualizations. Word processors and browsers are widely used and thus offer opportunities to become more experienced using them and to conveniently share materials across multiple courses. One consideration, however, is they offer very limited dynamic user control over the display.

For larger and more complex networks, however, tools more specific to network visualization may be required. In addition to the basic functions, another feature expected for very large networks would be the ability to read and write files in a simple format. After putting great effort into manually laying out a network, one would want to be able to save it, ideally in the form that can be read back into the tool. One would certainly want to save the display itself, ideally in a format widely accepted by word processors and browsers. The ability to choose colors, sizes and symbols for nodes and edges is also an important feature in the tool since, as well as adding meaning to the visualization; the feature also allows displaying larger networks in a limited amount of space. The ability to select nodes and edges and move them is likewise very useful. The ability to interface directly with other running programs is a more sophisticated feature but clearly important if one is creating sophisticated demonstrations.

5 NETWORK VISUALIZATION SOFTWARE

There are many network visualization tools available, both as freeware and as commercial products. Academic licenses are often quite reasonable. There is a wide variety of functionality. There are libraries in C, C++ and Java which can be used to build interactive applications. As well as standalone display tools which allow the user to read in files to display and also some which allow the user to edit the network interactively, there are full function tools with a variety of layout algorithms and interfaces to other authoring tools. There are authoring tools which integrate network display capability. Table 1 lists some network visualization tools, their sources, cost and types [7]. The reference contains a more complete list and links to the individual tools listed in the table [6, 8]. Another good place to search for graph drawing tools on the Internet is at http://directory.google.com/Top/Science/Math/Combinatorics/Software/Graph_Drawings [5].

Table 1: Network visualization tools

Name	Source	Cost	Dependencies	Summary
ADVIZOR	Visual Insights	commercial (contact company for pricing)	components for custom tools	ADVIZOR enables companies to add visual query and analysis solutions to their existing decision support infrastructure, and create and deploy interactive visual analysis applications and templates.

Graphviz	Stephen North, AT&T Research	free, src available	components for tool building	A set of graph drawing tools for Unix or MS- Windows. Designed for visualizing structural information by constructing geometric representations of abstract graphs and networks.
Otter	CAIDA	free, src available	standalone tool	A general purpose Java applet that can display any type of networking data, using various layout methods/modules.
Pajek	Vladimir Batagelj, Andrej Mrvar University of Ljubljani	free	standalone tool	A program package, for Windows 95/NT, for analyzing large networks (thousands of vertices).
Tom Sawyer Toolkits	Tom Sawyer Software	commercial (contact company for pricing)	components for custom tools	Customizable graph layout and diagramming toolkits for integration into other applications. A number of graph layout algorithms and graph editing tools are available.
Tulip	Auber David University Bordeaux I France	Free, src available	standalone tool	System dedicated to the visualization of hugh graphs. Manages graphs with nodes and edges up to 500,000 on a PC.

6 NETWORK VISUALIZATION TOOLS FOR EDUCATION

Network visualization software is also readily available, both as freeware and as commercial products, for educational purposes for a range of disciplines. The most obvious educational tools are used for instruction in computer science and include algorithm visualization and animations, displays of electronic circuitry, website maps, data structures, etc. Scientific visualization software is in abundance for physics, biology, chemistry, mathematics and bioinformatics. Network visualization software also exists for liberal arts, from timelines in history, to word ontology in linguistics. Listed below are some urls for visualization software for specified disciplines and selected illustrations of their displays.

Mathematics: <http://www.geom.uiuc.edu/software/download> and
http://www.math.nyu.edu/AML/ViSLab_software.html

Scientific applications: <http://www.physlink.com/Education/Software.cfm>

7 CONCLUSIONS

A fundamental goal of network visualization is to display relationships among objects in a manner that facilitates an understating of these relationships. We have discussed the issues and challenges in visually representing networks and the underlying principles for their display. We have surveyed a number of software tools that are currently available for creating and displaying networks. Additionally, we have shown tools for network visualization that can be used in an educational setting for a variety of disciplines.

The main conclusion that we have come to is that while there are many ways to improve the utility of network visualization software, the single most important capability, especially for large networks, is the ability for the user to control what part of the network is visible at any given time. This includes the ability to zoom and pan as well as the ability to display a subset of the nodes and edges in the network based on geographic decomposition, functional decomposition and compression.

References

- [1] G. Di Battista, P. Eades, R. Tamassia, I. G Tollis, Graph Drawing: Algorithms for the Visualization of Graphs, Upper Saddle River, NJ: Prentice Hall, 1999.
- [2] Stephen G. Eick, Aspects of Network Visualization, Computer Graphics and Applications, Vol. 16, No. 2, March 1996, 69-72.
- [3] Emden R. Gansner, Stephen C. North, An open graph visualization system and its applications to software engineering, Software – Practice and Experience, Vol. 30, Issue 11, 1-5, 1999, 1203-1233.
- [4] Ivan Herman, Guy Melancon, M. Scott Marshall, Graph Visualization and Navigation in Information Visualization: A Survey, IEEE Transactions on Visualization and Computer Graphics, Volume 6, Issue 1 (January 2000), 24 – 43.
- [5]http://directory.google.com/Top/Science/Math/Combinatorics/Software/Graph_Drawings
- [6] [http:// vlado.fmf.uni-lj.si/pub/networks/pajek/](http://vlado.fmf.uni-lj.si/pub/networks/pajek/)
- [7] <http://www.caida.org/projects/internetatlas/viz/viztools.html>
- [8] [http:// www.research.att.com/sw/tools/graphviz/](http://www.research.att.com/sw/tools/graphviz/)

- [9] Thomas L. Naps, Laura L. Norton, and James R. Eagan, JHAVÉ -- An Environment to Actively Engage Students in Web-based Algorithm Visualizations, in Proceedings of the SIGCSE Session, ACM Meetings (Austin, Texas, March 2000).
- [10] H.C. Purchase, R.F. Cohen, M.I. James, An experimental study of the basis for graph drawing algorithms, Journal of Experimental Algorithms, Vol. 2, No. 4, 1997.
- [11] Manojit Sarkar, Marc H. Brown, Graphical fisheye views of graphs, Proceedings of the SIGCHI Conference on Human factors in Computing Systems, Monterey, California, United States, 1992, 83-91.