Graph Visualization and Navigation as an Interface to Data Exploration

M. Delest[1], B. Leblanc[2], M. S. Marshall[3], G. Melançon[4]

Keywords:

Information Visualization, Relational Data, Visual Cues, Graph, Cognition (EN)

Visualisation d'information, Données relationnelles, Indices visuels, Graphes, Cognition (FR)

Visualización De la Información, Datos Emparentados, Señales Visuales, Gráfico, Cognición (ES)

Graph Visualization – a Type of Information Visualization

The visualization of data has a simple motivation: it allows humans to take advantage of the visual processing system of the brain (70% of all receptors and more than 40% of the cortex is devoted to vision) [1]. In contrast, attempting to find patterns in textual lists of alphanumeric data can require large amounts of inference and memory. The advantages of visualizing data extend from the realm of data visualization to graph (relational data) visualization. When successfully approached, graph visualization makes relations and values apparent to pre-attentive processing, freeing up the mind for higher-level cognitive tasks.

This paper argues that graph visualization serves the goals of strategic, scientific and technology watch (SSTW). Researchers and practitioners in SSTW seek methods and techniques to build proper data space. Conversely, graph visualization methods and techniques build from specific data and support tasks designed around this data.

Information visualization has become a field of its own and "sub-fields" are beginning to emerge (see, for example, Card *et al.* [2] for a collection of papers from the last decade, or [3, 9] for an overview). A simple way to determine the applicability of graph visualization is to consider the following question: *Is there an inherent relation among the data elements to be visualized*? If the answer to the question is "no", then data elements are "unstructured" and the goal of the information visualization system might be to help discover relations among data. If, however, the answer to the question is "yes", then the data can be represented by the nodes of a graph, with the edges representing the relations.

The process of discovery through visualization is an interactive process that usually begins with attempts to view some aspect of the entire data set in order to discover how to proceed with the investigation. Through a process of selection and refined views, the user tries to locate feature spaces of interest. This is actually the only reasonable scenario when dealing with massive data spaces, which is the average situation in SSTW. As Schneiderman stated it in the *visualization mantra*: "Overview first, zoom-in, then filter, and detail-on-demand; efficient focusing; as well as effective linking." In the case of graph visualization, the view of the data usually involves the spatialization of one feature of the data, i.e., a graph layout, possibly influenced by some additional data attribute other

[1] LaBRI CNRS UMR 5800, Université Bordeaux I, Bordeaux, France. E(mail: maylis@labri.fr

[2] ISPED, Université Bordeaux 2, Bordeaux, France. E-mail: Benoit.Leblanc@dim.u-bordeaux2.fr

[3] CWI, Amsterdam, The Netherlands. E-mail: Scott.Marshall@cwi.nl

[4] LIRMM CNRS UMR 5506, Université Montpellier 2, Montpellier, France. E-mail : Guy.Melancon@lirmm.fr

than edge information. Other features can then affect the choice of visual attributes such as color and size. The user can possibly discover patterns in the visualization that can be used to relate a particular data attribute to the structure of the graph. One efficient way of achieving this is by viewing how a visual attribute is distributed over the layout of the graph, although the user will eventually focus on the content associated with the graph elements.

Graph Visualization Application Domains

Graph visualization has many areas of application. Most people have encountered a file hierarchy on a computer system. A file hierarchy can be represented as a tree (a special type of graph). It is often necessary to navigate through the file hierarchy in order to find a particular file. Anyone who has done this has probably experienced a few of the problems involved in graph visualization: "Where am I?" "Where is the file that I'm looking for?" Other familiar types of graphs include the hierarchy illustrated in an organisational chart and taxonomies that portray the relations between species. Web site maps are another application of graphs as well as browsing history. In biology and chemistry, graphs are applied to evolutionary trees, phylogenetic trees, molecular maps, genetic maps, biochemical pathways, and protein functions. In the social sciences, social networks are studied for insight into the relations between people and organizations and how these relations might affect the outcome of interaction. Other areas of application include object-oriented systems (class browsers), data structures (compiler data structures in particular), real-time systems (state-transition diagrams, Petri nets), data flow diagrams, subroutine-call graphs, entity relationship diagrams (e.g. UML and database structures), semantic networks and knowledge-representation diagrams, both artificial and physical neural networks, Bayesian networks, project management (PERT diagrams), logic programming (SLD-trees), VLSI (circuit schematics), virtual reality (scene graphs), and document management systems. Note that the information isn't always guaranteed to be in a purely hierarchical format — this necessitates techniques that can deal with more general graphs than trees.

In all of the domains where graph visualization applies, the challenge remains the same: how can we choose an abstraction of a graph that will provide a beginning for exploration? How can we reduce the amount of data displayed without omitting key information from the image? How can we discover the areas of data that are relevant for our analysis? The problem is even more challenging when the data to be viewed is intended at a wide audience of non-experts.

Our group has developed methods that provide answers to these questions. This paper, in a sense, presents the research program we are conducting on different aspects of graph visualization. The automation of some of the tasks involved in graph visualization often requires the analysis of the structure of the graph underlying the information space. Because of this, we believe that combinatorial mathematics and statistical methods offer valuable tools [10,14]. Other issues that have been looked at concern the design of graph visualization systems and relate to software engineering issues [12]. Incidentally, all the techniques we will describe have been incorporated into a graph visualization system built from a generic Java API[5]. Finally, our research also covers cognitive issues concerned with the visualization of information on a computer [14]. Graphs are already used for the visualization

[5] See the website www.cwi.nl/InfoVisu/GVF.

of web search results, for example. However, it is still unclear how a non-expert user uses this type of representation to navigate the information space. A better understanding of computer-human interactions involved in graph visualization can only help design better visualization techniques and systems.

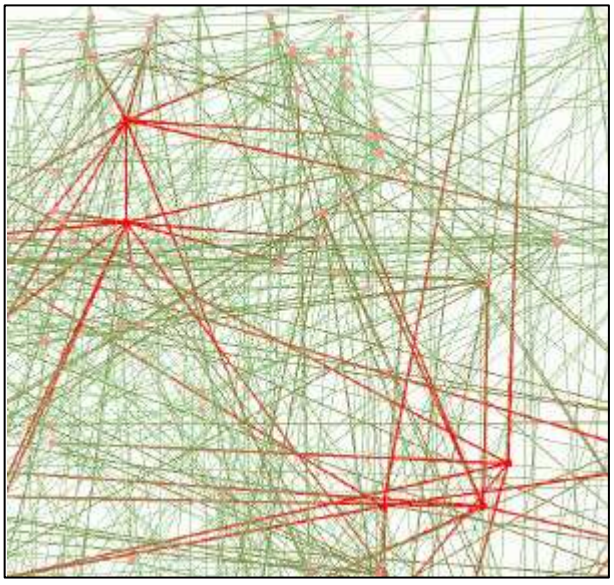
Using Metrics to affect Coloring

Two tasks in Graph Visualization require partitioning: the assignment of visual attributes and divisive clustering. Often, we would like to assign a color or other visual attributes to a node or edge that indicates an associated value. In an application involving divisive clustering, we would like to partition the graph into subsets of graph elements based on metric values in such a way that all subsets are evenly populated. Probability density functions derived from statistics about a metric can help systems succeed at these tasks. Assuming a uniform distribution of metric values during either partitioning or coloring can have undesired effects such as empty clusters or only one level of emphasis for the entire graph. This note will only focus on the assignment of visual attributes, although this approach may also be applied in the context of data partitioning [10].

Assigning a visual attribute (e.g., color, brightness, color saturation, or line width) consists of two steps:

1. Assign an abstract value, usually between zero and one, to each displayable element based on the element's metric value. We will refer to this abstract value as the *emphasis* of the element, and we will also refer to this mapping as the *emphasis mapping*.
2. Map the emphasis to a visual attribute. We will refer to this mapping as the *attribute mapping*.

The two mappings have different characteristics. It is therefore important to conceptually separate them. The mapping that creates the final visual attributes, is closely related to issues of perception and cognition, lighting conditions, display gamma values, and underlying graphics systems (see Ware's book [4], for example). In some cases, a simple linear mapping from an emphasis value to, for example, color saturation is acceptable. In other cases, a non-linear mapping is necessary. In our view, a visualization system should give the end-user some means of controlling the mapping used in order to adjust for his/her viewing conditions.



The image on the right is a zoomed view of a complex network of nodes. The high number of edge crossings makes it difficult to rely solely on the layout in order to get a structured view of the network. The technique used to produce the image involves mapping emphasis values to three visual attributes: color, saturation, and line width. The nodes of the graph have been assigned a metric value measuring

the intensity of a flow going through the graph[6] [11]. We interpolate between two colors as well as between low and high saturation based on the emphasis value. Similarly, high emphasis values map to thicker line widths.

A straightforward and naïve emphasis mapping sometimes fails to produce the desired effect because it does not take into consideration how the metric values are spread over the available interval. For instance, applying a linear emphasis to the graph in the image would result in a plain image because most of the edges would have very low color saturation. This is *not* the result of artifacts in the attribute mapping: modifying the distortions of the attribute mapping would not improve the picture -- just make *all* edges uniformly darker, for example.

Design and Analysis of Metrics

Metrics can be used for many different purposes, and, in our view, not all applications have been fully explored. For instance, metrics can be used to govern a subgraph extraction procedure. These ideas actually lie at the basis of search engines such as Google [**Erreur ! Source du renvoi introuvable.**]. In another application, metrics are used to influence layout [5]. Yet another use of metrics is the creation of fisheye views, as presented in the seminal paper of Furnas [6][7], where he computes the *Degree of Interest (DOI)* of elements in a tree. Elements with low values are hidden to improve the display of the structure (sometimes referred to as *semantic fisheye*) and help emphasize the more important elements in the tree. In semantic fisheye, the value used to determine the distance from the focus is a *semantic distance* rather than a geometric distance.

Key Issues in Graph Visualization

The size of the graph to view is a key issue in graph visualization. Large graphs pose several difficult problems. If the number of elements is large it can compromise performance or even reach the limits of the viewing platform. Even if it is possible to layout and display all the elements, the issue of viewability or usability arises, because it will become impossible to discern between nodes and edges. In fact, usability becomes an issue even before the problem of discernability is reached. It is well known that comprehension and detailed analysis of data in graph structures is easiest when the size of the displayed graph is small. In general, displaying an entire large graph may give an indication of the overall structure or a location within it but makes it difficult to comprehend.

Other than the usual reference to information overload and the occasional reference to some of the gestalt principles, papers in information visualization rarely apply cognitive science and human factors. This is for no lack of trying; very few of the findings in cognitive science have practical applications at

---

[6] The graph in the example is a directed acyclic graph (no circuit), equipped with the *Flow Metric*. Assign to every source node (no predecessor) the metric value *M (t)* = 1. Then compute values for every other node the following way: divide the value at a node by the number of its successors to find its contribution to each of them. A node receiving a set of values from its ancestors sums them up. More precisely, the value *M(v)* for a node *v* is obtained by summing contributions over the set of all its ancestors $a_1, \ldots, a_q$ $(q \geq 1)$. That is, $M(v) = \sum_j M(a_j) / number\ of\ successors\ of\ a_j$.

[7] Furnas used the term “degree of interest” but, in our terminology, his DOI function could be considered a type of metric.

this time and very few usability studies have been done. Cognitive aspects are undoubtedly a subject for continued research.

Scalability - The Limits of the Display and the User

The *size* of the graph to be viewed is a determining factor in graph visualization. Scalability is therefore of the utmost importance and should act as a guiding heuristic for the design of graph visualization systems. Few systems can claim to deal effectively with thousands of nodes, although graphs with this order of magnitude appear in a wide variety of applications. NicheWorks [7], H3Viewer [8] are among the few systems that claim to handle data sets with thousands of elements. The size of a graph can make a normally good layout algorithm completely unusable. In fact, a layout algorithm may produce good layouts for graphs of several hundred nodes, but this does not guarantee that it will scale up to several thousand nodes. When the layout is too dense, interaction with the graph becomes difficult. Occlusions in the picture make it impossible to navigate and query about particular nodes. The use of 3D or of non-Euclidean geometry have also been proposed to alleviate these problems. However, beyond a certain limit, no algorithm will guarantee a proper layout of large graphs. There is simply not enough space on the screen. In fact, from a cognitive perspective, it does not even make sense to display a very large amount of data. Consequently, a first step in the visualization process is often to reduce the size of the graph that is actually displayed. As a result, classical layout algorithms remain usable tools for visualization, but only when combined with these techniques.

GVF – Graph Visualization Framework in Java

When faced with the task of analyzing a set of data with relations we are immediately faced with some of the challenges in graph visualization. For instance, we must choose a type of layout before we can draw the graph. If the graph is too large to fit on the screen, we must choose an abstract view of the graph that exposes certain types of information about the graph, yet reduces the amount of information displayed. Both choices influence what we can discover about the graph, because they determine which information is presented and how it is presented. However, our needs don't stop here; we would also like to interact with the graph, changing the view in order to gain insight into the data. All these features require a system that can easily adapt to our needs and quickly change the way a graph is presented. Simple filtering techniques based on attributes don't always suffice, so more sophisticated techniques such as clustering are often used. The process of clustering involves discovering groups in the data. A fundamental technique in graph visualization displays the groups, or clusters, of a graph using a special type of node called a *meta-node* to represent clusters or subgraphs in the graph. This technique makes it possible to represent a graph by displaying fewer elements, allowing the user to control the level of detail by "opening" and "closing" meta-nodes. Such an approach requires a way to store and manipulate graphs whose nodes may represent subgraphs.

Other tasks besides clustering make support for constantly changing graphs necessary. For example, the ability to edit a graph can be an important part of a graph visualization system. A user experimenting with layout may want to add or delete parts of the graph to see the effect it has on a particular layout, or simply edit the properties of a given element to see the resulting effect. Systems that update graphs with real-time information also require a way to handle constantly changing

structures. All of these tasks make support for dynamic graphs an important requirement for graph visualization systems.

In 1999, we started looking for an environment in which it was possible to experiment with a variety of graph visualization techniques. The experimentation would include the interactive definition of nested clusters and the use of visual elements to represent graphs and their properties. Although we have looked at other class libraries for graphs in addition to many complete systems, none fulfilled our needs and so we decided to develop our own. We wanted to create a system that is general enough that it can be embedded in information visualization applications but also be used to create a standalone application. Because our goal was also portability and ease of maintenance, we decided to implement it in Java. Although the development of the Graph Visualization Framework (GVF) was done in Java, we believe that the solutions we have found are of general interest for object-oriented programming. The GVF is designed around an original object-oriented node-centered data structure for representing a graph, which makes it scalable and suitable for a wide range of applications [12].

References

1. C. Ware, “Invited Talk: The Visual Representation of Information Structures”, in: Proceedings of Symposium on Graph Drawing (GD 2000), Colonial Williamsburg, Virginia, USA, pp. 1-4, 2000.
2. S. K. Card, J. D. Mackinlay, and B. Shneiderman, *Readings in Information Visualization* San Francisco, Morgan Kaufmann Publishers, 1999.
3. R. Spence, *Information Visualization* Harlow, England, ACM Press/Addison-Wesley, 2001.
4. C. Ware, *Information Visualization: Perception for Design* Orlando, FL, Morgan Kaufmann Publishers, 2000.
5. R. M. Wilson and R. D. Bergeron, “Dynamic Hierarchy Specification and Visualization”, in: Proceedings of IEEE Symposium on Information Visualization (InfoVis '99), pp. 65-72, 1999.
6. G. W. Furnas, “Generalized Fisheye Views”, in: Proceedings of Human Factors in Computing Systems CHI '86, pp. 16-23, 1986.
7. G. J. Wills, “Niche Works - Interactive Visualization of Very Large Graphs”, *Journal of Computational and Graphical Statistics*, vol. 8, pp. 190-212, 1999.
8. T. Munzner, “Drawing Large Graphs with H3Viewer and Site Manager”, in: Proceedings of Symposium on Graph Drawing GD '98, Berlin, pp. 384-393, 1998.
9. I. Herman, M. S. Marshall, and G. Melançon, “Graph Visualization and Navigation in Information Visualization: A Survey”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, pp. 24-43, 2000.
10. I. Herman, M. S. Marshall, and G. Melançon, “Density Functions for Visual Attributes and Effective Partitioning in Graph Visualization”, in: Proceedings of IEEE Symposium on Information Visualization (InfoVis'2000), Salt Lake City, Utah, U.S., pp. 49-56, 2000.
11. I. Herman, M. S. Marshall, G. Melançon, D. J. Duke, M. Delest, and J.-P. Domenger, “Skeletal Images as Visual Cues in Graph Visualization”, in: Proceedings of Joint Eurographics and IEEE TCVG Symposium on Visualization (Data Visualization '99), Wien, pp. 13-22, 1999.
12. M. S. Marshall, I. Herman, and G. Melançon. An object-oriented design for graph visualization. Software - Practice and Experience, vol. 31, pp. 739-756, 2001.
13. S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In: WWW7 Seventh International World Wide Web Conference, April 1998, Brisbane, Australia.
14. I. Herman, M. Delest, G. Melançon. Tree Visualization and Navigational Clues for Information Visualization, in: *Computer Graphics Forum*, volume 17, number 2, 1998, pp. 153 – 166.
15. B. Leblanc, D. Dion, D. Auber, G. Melançon, Constitution et visualisation de deux réseaux d'associations verbales, Colloque ALCAA « Agents logiciels, coopération, apprentissage & activité humaine », Biarritz, France, September 2001.