Automatic Generation of Interactive Overview Diagrams for the Navigation of Large Graphs

M. Scott Marshall, Ivan Herman, Guy Melançon

CWI

P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Email: {M.S.Marshall, I.Herman, G.Melancon}@cwi.nl

ABSTRACT

A clustered graph can be used to build an abstract view of its non-clustered counterpart and reduce visual complexity. The classic approach to interaction with a clustered graph is limited in scalability and efficacy, underlining the need for an overview diagram. We present a technique for the automatic generation of an overview diagram based on hierarchical clustering and discuss its application to graphs. Hierarchical clustering induces a tree structure that is useful as a map to navigate the original data set. Because the resulting overview diagram is itself a graph, it can be manipulated by the same tools that are available for graphs.

1998 ACM Computing Classification System: H.5.2, I.3.4, I.3.8

Keywords and Phrases: Overview diagram, hierarchical clustering, interactive clustering, exploratory data analysis

Note: The work was carried out under the project INS3.1 “Information Visualization”

1. INTRODUCTION

The analysis of large graphs is one of the challenges of graph visualization. The display of a large relational data set quickly uses up the available pixels and challenges the perceptive abilities of the user. How can we represent a large graph with an abstract image? A fundamental technique in graph visualization is to find groups or clusters within the graph that can be represented by a single visual element called a *meta-node* (more generally known as *visual elision*). When such groups are defined, the graph is said to be *clustered* and the graph can be displayed with fewer elements. In the classic approach to navigation, the user can then control the level of detail in the display by “opening” and “closing” the meta-nodes. The act of opening a meta-node triggers the display of the elements contained in the meta-node. See Schaffer *et al*.[1], Eades and Feng[2], and Bartram[3] for a few of the many applications of this principle.

When the process of clustering is applied to the clusters produced in a previous step, it is called *hierarchical clustering*. The clustered graph produced by such a process will have multiple levels of nested subgraphs. This process can be advantageous when applied to large data sets because it is more powerful at dividing the data into subsets than a single partitioning can be.

Many different strategies involve hierarchical clustering. Some strategies use application dependent attributes of the data [4, 5] while others are based on graph properties such as connectivity or even a given layout of the graph [6-8]. In this paper, we discuss clustering based on *metric values* for nodes. A metric is some measure of data that is associated with a node. If the metric is solely based on structural information about the graph, then it requires no domain knowledge and can be used on any graph.

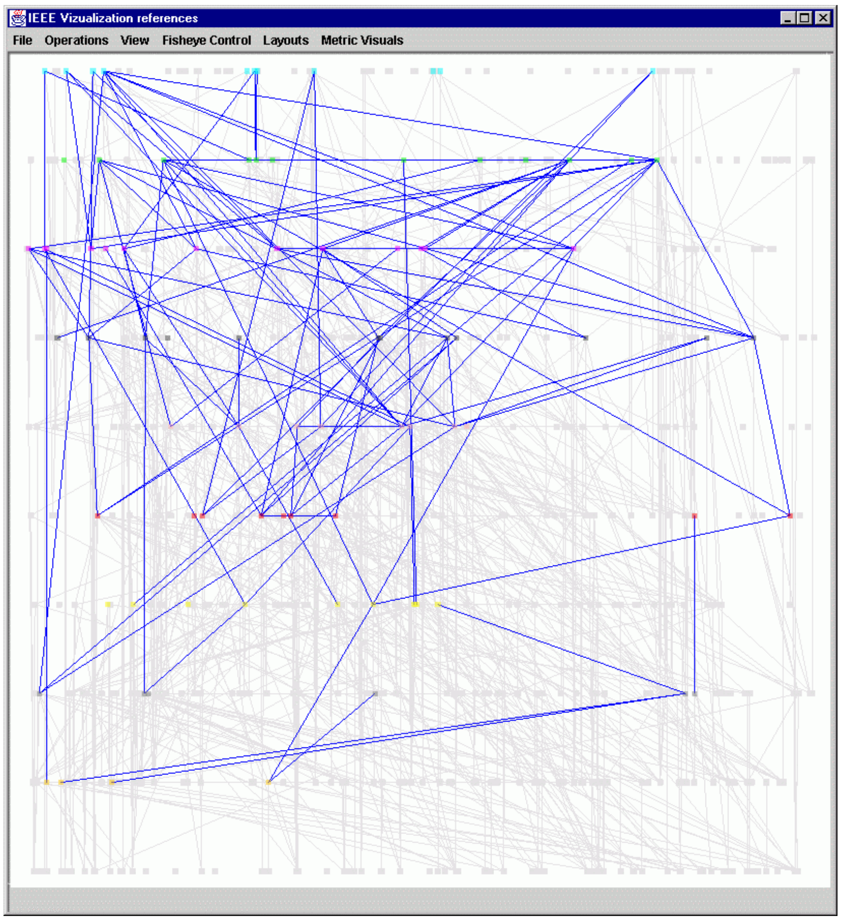
IEEE Vizualization references
File Operations View Fisheye Control Layouts Metric Visuals

Figure 1 The cluster selected in Fig. 2 is emphasized, with the containing graph in the background (see the color plates).

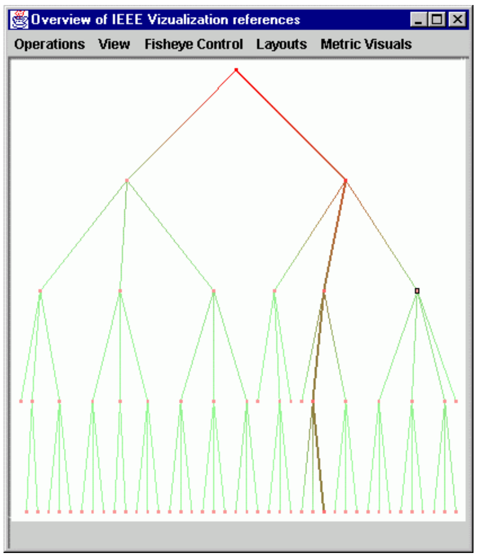
Overview of IEEE Vizualization references
Operations View Fisheye Control Layouts Metric Visuals

Figure 2 Overview diagram with metric coloring (the single highlighted node indicates the selected cluster in Fig. 1)

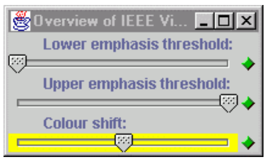

Figure 3 Slider for control of cutoff filter. (The color shift slider plays no role in the techniques described in this paper.)

The results of hierarchically clustering a graph based on node metrics depend on several factors including: the metric that is used during partitioning, the manner in which partitioning is done, and the stop condition. The first step in partitioning is to divide the range of metric values into distinct and consecutive sub-intervals[*]. An element of the graph is assigned to a partition if its value falls within the boundaries defined for the sub-interval. Recursively applying this method on each sub-interval leads to a hierarchical partitioning of the graph elements. Partitioning can be as simple as a linear division of the range of available values (although this approach produces inconsistent results. In our experience, statistical knowledge about the metric values should be used during partitioning, see [9]). It is important to note that the goal of partitioning in this case is to divide the data set into manageable-sized chunks rather than to discover classes (although classes could become apparent as a result of the process). The stop condition for hierarchical clustering will typically use some logic that attempts to balance the size of the cluster with the number of potential subgroups that can be found. A simpler stop condition is to limit the depth of clustering (i.e., how many recursive steps the clustering uses), although this provides less control over the resulting cluster size.

The classic approach to navigating clustered graphs is limited in scalability. For example, in the case of hierarchical clustering, as the depth of nesting increases, the user must open an increasing number of meta-nodes in the resulting graph in order to navigate. This makes it difficult to remember the current location in the overall data set and requires a large number of navigational actions in order to reach a particular location in the graph. A solution to this problem is to provide an overview diagram.

[*] We use contiguous intervals although non-contiguous intervals could also be useful in some applications.

2. INTERACTIVE OVERVIEW DIAGRAM

Many different forms of overview diagrams have been documented in the literature. Mukherjea *et al.*[10] and Fua *et al.*[11] are just a few examples. The overview diagram is an abstract view that serves as a map into the data, although some varieties are limited in purpose to displaying the current location rather than controlling it. However, creating an abstract view of a graph is a challenging problem. One approach has been proposed that works well on trees[12] but we would like a method that works on any type of graph and does not require a layout of the entire graph in order to work.

Hierarchies or trees have several advantages when used to represent data: they are familiar to most users and the relations between elements are clearly visible from their position (at least, when a layout algorithm such as the one described by Reingold and Tilford[13] is used). There are also several established techniques for drawing trees. In fact, trees are so useful that it is tempting to extract a tree from a graph in order to draw it with a familiar and predictable algorithm that is immediately tangible to most users[14]. For these reasons, we would like a way to represent our data as a tree in the overview diagram.

The process of hierarchical clustering can be represented as a tree, which makes it a simple matter to create an overview diagram if graph drawing tools are available. In our graph visualization application[*], we create an overview diagram in the form of a tree from the hierarchical clustering process (shown in Figure 2). The nodes in the overview diagram correspond to clusters in the original graph. Note that because the cluster structure is based on a partitioning of the range of metric values, nodes on the right side of the overview diagram correspond to clusters containing elements with higher metric values. The granularity of the clusters increases with the depth of the tree. The population of a cluster decreases with the depth of the cluster in the tree and there is a containment relation from parent to child.

Using a tree as the overview diagram provides the user with a map of discrete entry points into the data set. By selecting a node in the overview diagram the user selects a cluster of the original graph, whose elements are then displayed in a separate window (shown in Figure 1). The depth of the selected cluster in the hierarchy determines the level of detail (i.e., the number of elements in the view). It is also possible to select a cluster in the overview diagram to be used as the background context (grayed out background) in order to compare clusters. Although it is possible to display the full graph in the background for smaller graphs, this is less practical when the graph is large: the background would display a complex, gray cloud of lines. Finally, information and statistics on the metric values for the current cluster can be called up in an information window. These facilities allow the user to assess both the metric and the current partitioning.[†]

In an earlier paper[12], we described a simple form of clustering and corresponding interaction using sliders (shown in Figure 3). Using a slider one could interactively control a cut-off value: only elements with a metric value larger than the cut-off value would be displayed on the screen. This corresponds to the selection of a single sub-interval within the range of all metric values. A discrete version of the slider can be simulated by selecting a subset of consecutive leaf nodes including the rightmost leaf. The behavior of this mechanism will approach that of the slider as the partitions are refined. The full use of the overview graph gives an even more flexible selection mechanism. The collection of selected nodes in the overview diagram corresponds to a collection of (not necessarily consecutive) sub-intervals on the range of metric values. A finer sub-interval is selected using a node positioned deeper in the tree. Multiple-brushing is achieved by selecting multiple nodes in the overview diagram. Multiple clusters can also be chosen as the background context in the same manner.

The discrete nature of a tree as the overview diagram has several advantages. First, it is possible to quickly jump to another side of the spectrum. It is also possible to combine non-adjacent selections of clusters (i.e., from different parts of the spectrum) to create views of the data. Neither of these activities is possible with a continuous interface such as a slider or the structure-based brushes of Fua *et al.*[11]. Furthermore, the user can return to a particular view of the data by remembering a discrete position within the hierarchy. Overview diagrams without discrete interactive components make such actions more difficult.

Occasionally, the user may want to see a particular cluster further partitioned. For example, in the overview diagram of Figure 2, the leaf node at the end of the emphasized edge has a relatively large population. Interactive clustering is

[*] Our graph visualization application framework (in Java) and an application built using it are available at http://www.cwi.nl/InfoVisu

[†] Note that it would be possible to adapt the layout algorithm of the tree so that it would reflect the clusters it describes. See, for example, Wilson and Bergeron[5].

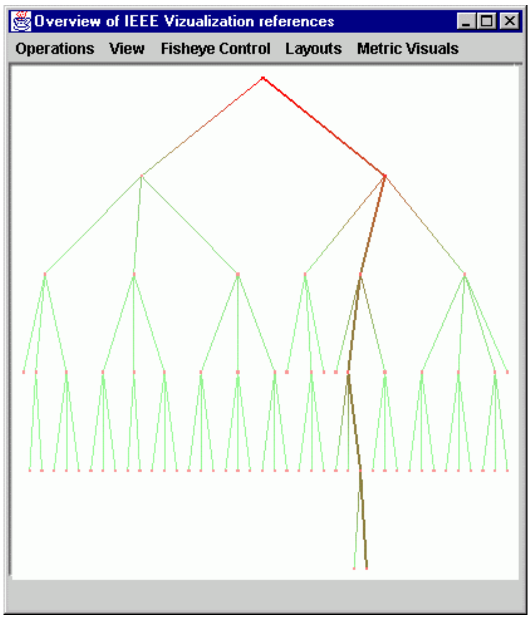

Figure 4 User triggers additional partitioning in cluster with highest population (see also Fig. 2)

triggered by activating the cluster of interest (it must be a leaf node in the overview diagram). This results in renewed partitioning of that particular cluster and is reflected in a new overview diagram shown in Figure 4. With the new partitions, the user can discover trends within the cluster.

3. Overview Diagrams as Graphs

If the overview diagrams are used within a graph visualization system, there is an added advantage to making an overview diagram from a graph: all interactive facilities provided by the system are also available for the overview graph including: zoom, pan, fisheye distortions, alternative layout, and even hierarchical clustering. This fact becomes particularly interesting if we apply metric-related techniques to the overview graph itself. The metric that seems to be useful in this case is one that uses the number of nodes contained in the cluster as its value. Consider the technique described in [12] (see also [9]) which consists of coloring the nodes and edges with continuously shaded color that reflects the metric values of the nodes at its endpoints. If we apply this coloring strategy to the overview diagram, we can easily determine the distribution of elements in the clusters. For example, in Figure 2, we can see that one branch of the tree stands out from the rest. This branch leads to the cluster with the highest population. From this image, it is also easy to see that none of the other clusters has a comparable population.

Another facility is the use of the slider, but used this time on the overview graph rather than the original graph (see Figure 3). This tool allows us to filter the overview diagram based on, for example, the population of elements in the clusters. A slider with a low cutoff and one with a high cutoff value are available, which makes it possible to search for clusters with low values, high values, or a range, if both are combined.

4. Conclusions

The approach to building an overview diagram described here requires no interaction. The automation is possible because of the application of structural metrics. Of course, this approach is not limited to graphs because hierarchical clustering can induce a tree structure on any data set, whether the data is relational or not. The fact that we can treat the overview diagram as a full-featured graph in our application permits effective combinations of techniques to be used to find the elements of interest.

A few issues remain to be explored. How can we change the overview diagram if we choose a different partitioning strategy such as non-contiguous boundaries for adjacent partitions? Are there situations where a second-order overview diagram (produced by hierarchically clustering the overview diagram itself) is particularly helpful? Are there metrics that will aid in finding exceptional areas of the overview diagram, for instance, where the partitioning was different? These are some of the issues that we would like to explore in future work.

REFERENCES

[1] D. Schaffer, Z. Zuo, S. Greenberg, L. Bartram, J. Dill, S. Dubs, and M. Roseman, “Navigating Hierarchically Clustered Networks through Fisheye and Full–zoom Methods”, *ACM Transactions on Computer–Human Interaction*, vol. 3, pp. 162–188, 1996.

[2] P. Eades and Q.-W. Feng, “Multilevel Visualization of Clustered Graphs”, in Proceedings of 4th International Symposium on Graph Drawing, Berkeley, California USA, 1996.

[3] L. Bartram, “Perceptual and interpretative properties of motion for information visualization”, in Proceedings of Workshop on New Paradigms in Information Visualization and Manipulation, Las Vegas, 1998.

[4] S. Mukherjea and J. D. Foley, “Visualizing the World-Wide Web with the Navigational View Builder”, *Computer Networks and ISDN Systems*, vol. 27, pp. 1075-1087, 1995.

[5] R. M. Wilson and R. D. Bergeron, “Dynamic Hierarchy Specification and Visualization”, in Proceedings of IEEE Symposium on Information Visualization (InfoVis '99), 1999.

[6] C. J. Alpert and A. B. Kahng, “Recent Developments in Netlist Partitioning: A Survey”, *Integration: the VLSI Journal*, vol. 19, pp. 1-81, 1995.

[7] V. Batagelj, A. Mrvar, and M. Zaveršnik, “Partitioning Approach to Visualization of Large Graphs”, in Proceedings of Symposium on Graph Drawing GD '98, Czech Republic, 1999.

[8] C. A. Duncan, M. T. Goodrich, and S. G. Kobourov, “Balanced Aspect Ratio Trees and Their Use for Drawing Very Large Graphs”, in Proceedings of Symposium on Graph Drawing GD '98, Berlin, 1998.

[9] 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), 2000.

[10] S. Mukherjea, J. D. Foley, and S. Hudson, “Visualizing Complex Hypermedia Networks through Multiple Hierarchical Views”, in Proceedings of CHI'95, 1995.

[11] Y. H. Fua, M. O. Ward, and E. A. Rundensteiner, “Navigating Hierarchies with Structure–Based Brushes”, in Proceedings of IEEE Symposium on Information Visualization (InfoVis '99), 1999.

[12] 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 *Data Visualization '99, Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization*, E. Gröller, H. Löffelmann, and W. Ribarsky, Eds. Wien: Springer–Verlag, 1999, pp. 13–22.

[13] E. M. Reingold and J. S. Tilford, “Tidier Drawing of Trees”, *IEEE Transactions on Software Engineering*, vol. 7, pp. 223-228, 1981.

[14] I. Herman, M. S. Marshall, and G. Melançon, “Graph Visualisation and Navigation in Information Visualisation: A Survey”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, pp. 1-20, 2000.

[15] G. W. Furnas, “Generalized Fisheye Views”, in Proceedings of Human Factors in Computing Systems CHI '86, 1986.