RINGS : A Technique for Visualizing Large Hierarchies

Soon Tee Teoh and Kwan-Liu Ma

Computer Science Department,
University of California, Davis
{teoh, ma}@cs.ucdavis.edu

Abstract. We present RINGS, a technique for visualizing large trees with hundreds of thousands of nodes. We introduce a new ringed circular layout of nodes to make more efficient use of limited display space. RINGS provides the user with the means to specify areas of primary and secondary focus, and is able to show multiple foci without compromising understanding of the graph. The user can also control the context shown. The strength of RINGS is its ability to show more area in focus and more contextual information than existing techniques using a fixed resolution display. We demonstrate the effectiveness of RINGS by applying it to the visualization of a Unix file directory.

1 Introduction

Ringed **I**nteractive-**N**avigation **G**raph **S**ystem (RINGS) is a technique for visualizing large hierarchical data, as well as large graphs if a spanning tree of the graph is provided as input. RINGS is based on a new ringed circular tree layout algorithm described in more detail in Section 2. Existing methods such as [3, 5, 7] tend to leave large regions of space empty. Shneiderman's Tree-maps [8] method makes efficient use of screen space to show a large number of nodes, but a common criticism is that structural information of the visualized trees is unclear in the Tree-map layout. Three-dimensional visualization methods such as Munzner's hyperbolic tree [5] and Kleiberg's botanical tree [2] can be used to navigate through very large graphs and trees respectively. However, overlapping of edges due to projecting a three-dimensional graph into two dimensions can lead to occlusion and ambiguity, limiting their effectiveness.

The main advantage of the ringed circular method is its efficient use of limited display sizes by showing more distinguishable nodes at one time, without any of the other methods' shortcomings listed above. Another important feature of RINGS is its capability to show more contextual information without compromising the clarity of the area in focus. Besides providing interactive navigation and focus+context solutions common in many existing systems, RINGS also has a unique ability to effectively show areas in secondary focus. These and other features of RINGS are described in detail in later sections.

Besides visualizing the topology of graphs, RINGS can also be used to visualize other information embedded in the graph. For example, in Section 5, we demonstrate how RINGS can be used to visualize file sizes, locate specific files and types of files in a directory, and reveal other aspects of the file directory structure.

2 Layout Algorithm

Recent graph-drawing algorithms such as Kreuseler et. al's *tree on a hemisphere* [3] and Munzner's *hyperbolic tree* [5] provide good navigation and focus+context solutions, but they are often less effective in using the available screen space. With a 512×512 display, typically less than 100 nodes and edges are distinguisable at one time while the rest of the graph appears clumped together. In RINGS, we make better use of the available screen space by showing more distinguishable edges and also allowing more structural information to be shown in context.

In RINGS, a node and all its children are placed in a circle. In this way, the branch any node belongs to is easily identified. Representing each node by a circle has been described in [1, 4, 6]; however, they place children only on the circumference of the parent circle, wasting a lot of empty space in the interior. In the new ringed circular layout used in RINGS, equal-sized circles corresponding to children are placed in concentric rings around the center of the parent circle, as shown in Figure 1.

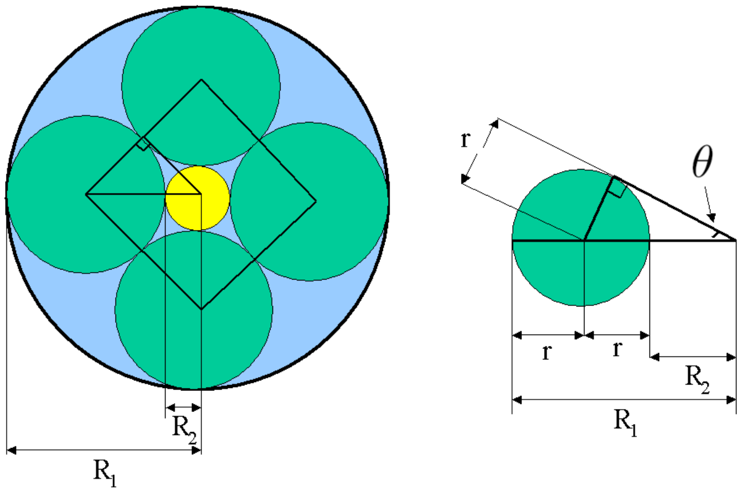

Fig. 1. Ringed circular layout of nodes: The four green circles represent four largest children of the parent node, represented by the large blue circle. The small yellow inner circle is the area left for the remaining children.

Connecting the centers of n equal circles placed in a ring makes an n-sided regular polygon. In Figure 1, $\theta = \pi/n$, where n is the number of circles in a ring,

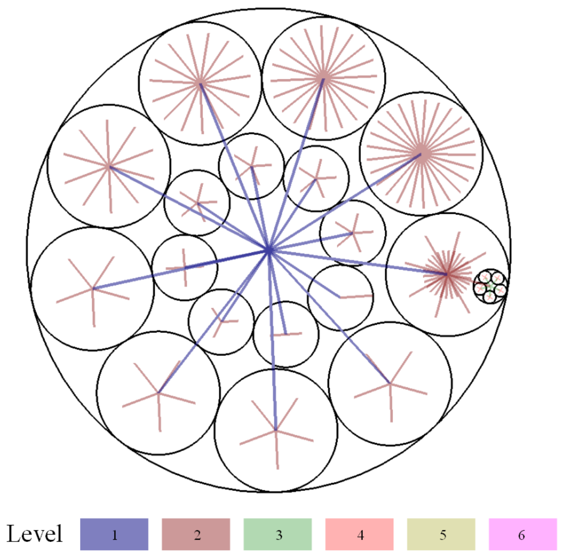

Fig. 2. Each node is placed at the center of its circle, and edges are drawn between each pair of connected nodes. Edges are colored according to their distance from the root. Level 1 edges are those directly connected to the root.

and θ is in radians. A simple relationship can be derived between the number of children circles in the outermost ring and the percentage of area taken up by the ring. *f(n)*, the fraction of the area left after n circles have been placed in the ring is given by:

$$f(n) = \frac{(R_2)^2}{(R_1)^2} = \frac{(1 - sin(\theta))^2}{(1 + sin(\theta))^2} = \frac{(1 - sin(\frac{\pi}{n}))^2}{(1 + sin(\frac{\pi}{n}))^2} \ . \tag{1}$$

This relationship is used to determine the number of children to be placed in each concentric ring. The algorithm first sorts the children by their number of children. Next, the algorithm finds the smallest k for which the sum of the number of children of the first k children expressed as a fraction of the total number of grandchildren is greater or equal to *f(k)*. The first k children are placed in the outermost ring. The rest of the children are placed in the same way in the inner rings. A node is positioned in the center of its circle. Edges are drawn as straight lines connecting nodes, as shown in Figure 2. In this figure, the root has 17 children, and they are arranged by the algorithm in order of their *size*, where the size of a node is defined by the number of children it has. The largest child itself has 35 children, and one of those children has descendents, whose circles are outlined in the figure. By placing the largest children in the outermost ring, the number of children in the outer rings is minimized, thereby minimizing the amount of occlusion due to edges drawn from the parent to the children.

3 Interactive Navigation

Like other large graph visualization systems, RINGS allows the user to interactively explore the large data. The primary focus is changed simply by clicking on the child to focus on. The child will be moved to the center and the parent will be moved to the side as a smooth animation. Figure 3 demonstrates the results of a change of focus performed for the visualization of a 20,000-node Unix file directory. The child in focus becomes the *pictorial root* because of its central position, and its *true parent* becomes a *pictorial child*, since it is placed in a circle on a ring just as any child is. To distinguish the *true ancestors* from the descendents of the pictorial root, the path from the pictorial root back to the *true root* is colored black.

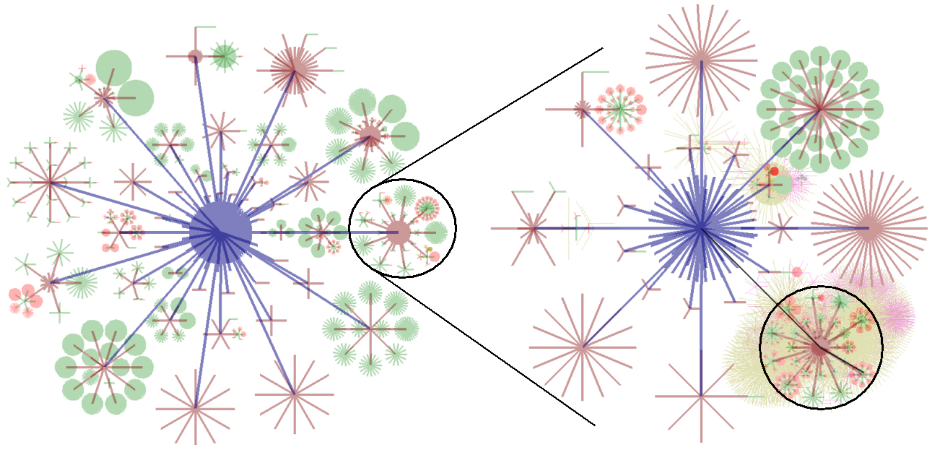

Fig. 3. Primary focus is changed by moving a child to the center. The circled child in the left drawing is expanded to the center in the right drawing, while the rest of the graph is moved to the circle in the right drawing. The edge from the focus to its true parent is colored black to distinguish it from the other edges.

Secondary foci can be set by clicking on a child to focus on and dragging it to another child currently occupying a larger area. The positions of the two children are swapped, again with smooth animation (see Figure 4). If the secondary focus is not large enough, the user can reduce the number of children in the outermost ring so that the secondary focus can be enhanced, also shown in Figure 4. The ability to effectively show secondary foci is generally not found in existing methods.

4 Other Features

Nodes at larger topological distances from the pictorial root occupy very little screen area, so they may become indistiguisable from neighboring nodes. Many

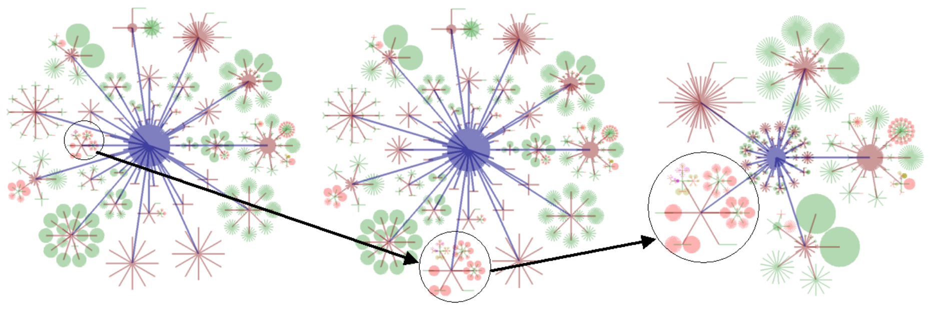

Fig. 4. Secondary focus is changed by swapping two children and, additionally, by reducing the number of children in the outermost ring. RINGS smoothly animates the changes.

edges far away from the foci map to the same pixels on the screen. Figure 5 shows how RINGS provides the option of expanding those nodes out of the confines of their circle. This additional contextual information gives the user a sense of the depth and topology of particular branches without compromising the clarity of the regions in focus.

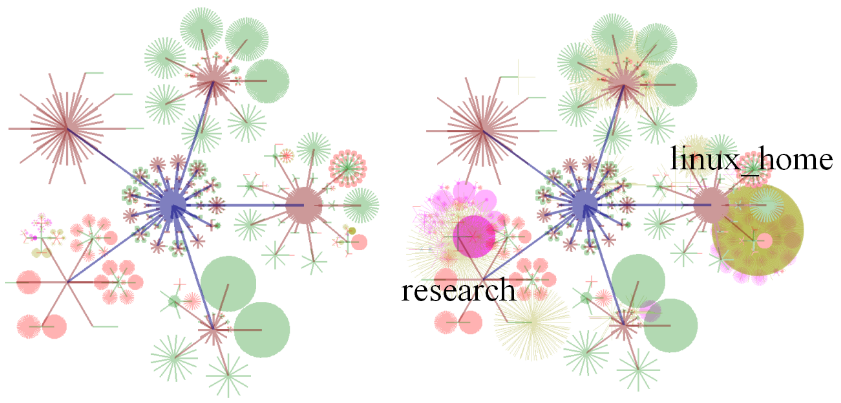

Fig. 5. The drawing on the right shows circles at topological distance 4 from the root expanded 200 times. Better sense of topology is gained without losing focus. The depth of certain sub-directories under "research" and "linux_home" is revealed.

Visual cues like color and transparency are also used to effectively enhance structural information. A simple way to improve perception of graph topology is to use different colors for edges with different topological distances from the root. In the color map (shown in Figure 2) used in our examples, darker, more saturated colors are used for edges closer to the root to highlight them. With the RINGS layout, some edges overlap. To overcome the inevitable occlusion,

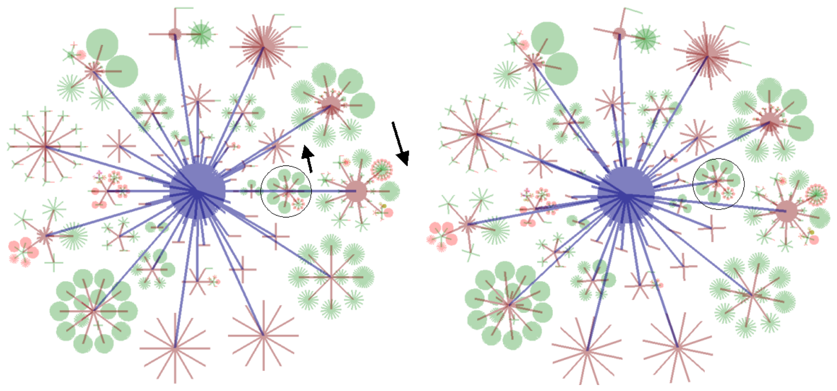

Fig. 6. Rotation: Occlusion of particular nodes can be eliminated by rotating adjacent rings in opposite directions.

edges are drawn as semi-transparent lines to reveal the parts of the graph they block. In addition, RINGS allows the user to rotate adjacent concentric rings in opposite directions (see figure 6) to eliminate the occlusion of a particular branch.

When visualizing very large graphs, it becomes computationally very expensive to traverse every node in the graph and render individual edges. To maintain navigation at interactive rate, nodes topologically further from the root are shown at reduced resolution where possible. For example, a dense node may have many adjacent edges mapping to the same pixels, and the resulting image formed from all the lines drawn is a circle. Instead, by simply rendering a single circle representing all the edges, rendering time and artifacts are both reduced.

5 Visualizing Information Beyond Topology

Figure 7 shows the application of RINGS to the visualization of a Unix file directory. Edges in the graph are colored according to the size of the file or directory. The two pictures show how varying the color map can be used to explore the data.

Other information in the data can also be visualized. Figure 8 shows the result of searching for a file “texture.tga”. The entire path to the file is highlighted in red. In this picture, the user can see the location of the file in relation to the whole directory. The branch and sub-branches the file belongs to are also revealed. Figure 9 highlights all the files in the directory tree with “.cpp”, “.cc”, “.c” or “.h” extensions. This identifies all the sub-directories containing those C++ source files. Examining individual sub-directories of Figure 9 gives insight into their internal structure. Examples of interpreting the visualization are explained in Figure 10.

By visualizing typical ringed circular graphs, we can classify the patterns observed. This classification is shown and explained in Figure 11. Understanding

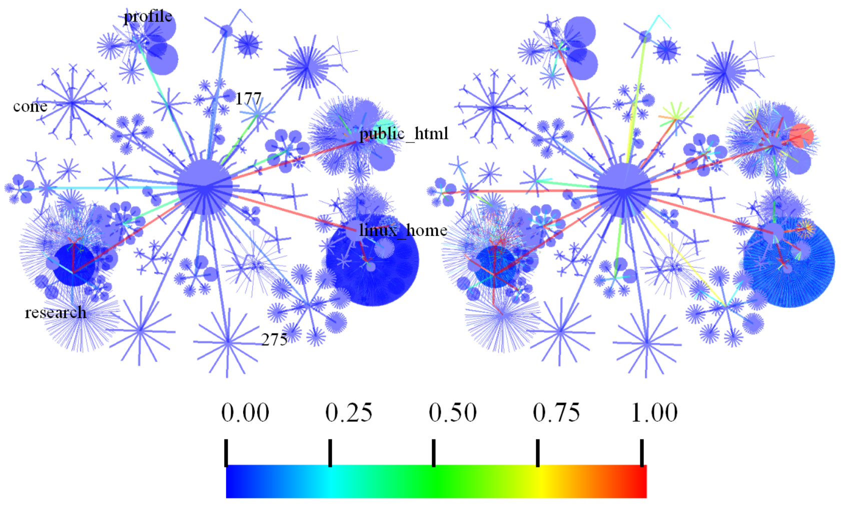

Fig. 7. Visualizing a Unix directory with RINGS : A color map is used to show files and directories of different sizes. In the left picture, the value 1.0 is set to be 0.1 of the total size of all files in the directory. In the right picture, 1.0 is set to 0.01. All nodes of sizes greater than 0.01 of the total size of the entire directory are colored red.

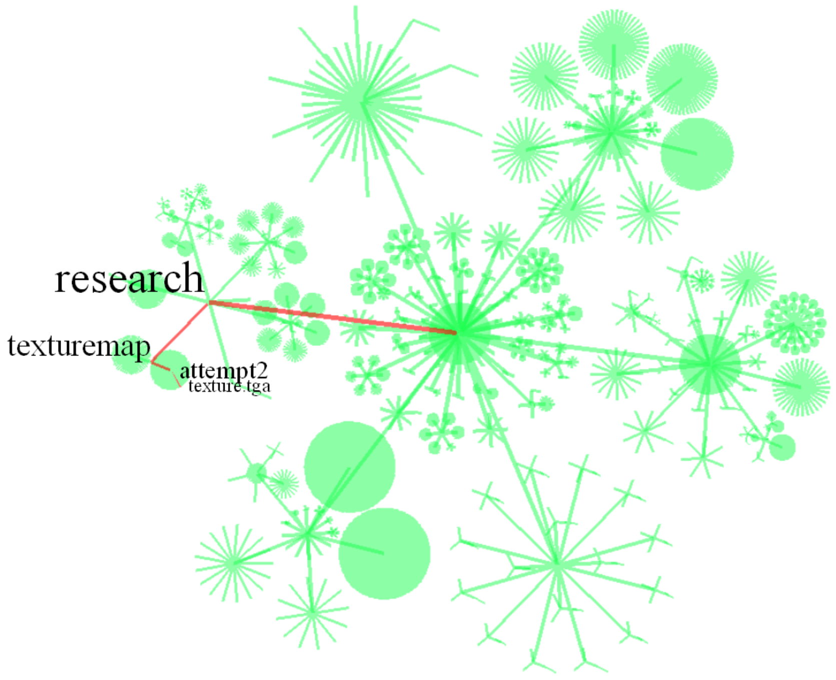

Fig. 8. Visualizing the location of a particular file with respect to its neighbors, the branch it belongs to, and the entire directory tree.

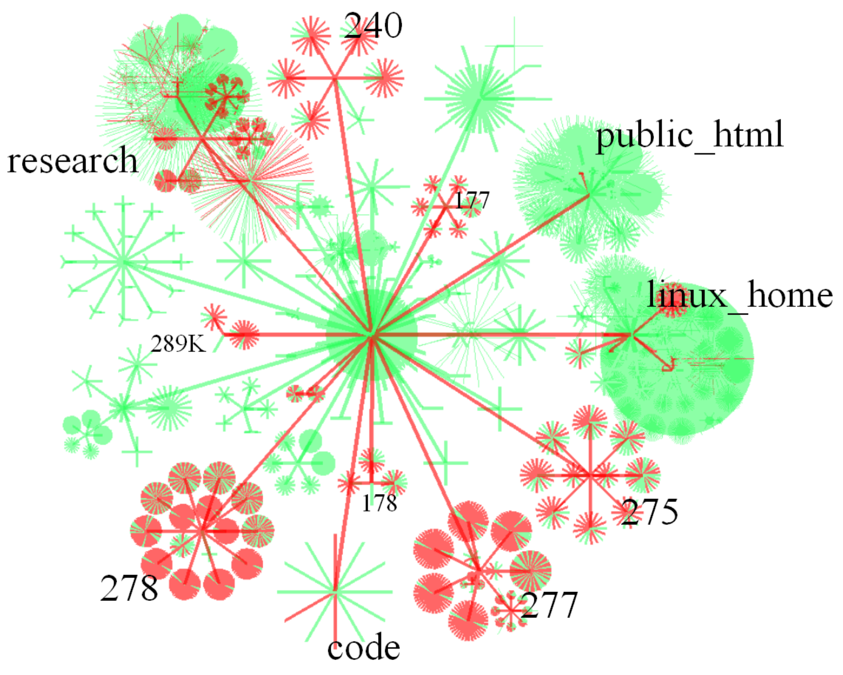

Fig. 9. Highlighting all files ending with ".cpp", ".cc", ".c" or ".h" and all the directories containing them. This shows the directories containing C++ code.

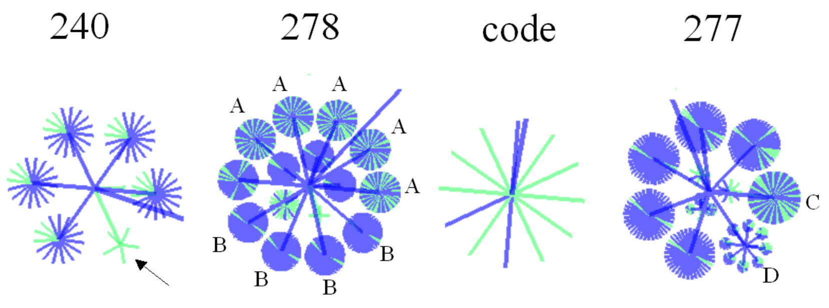

Fig. 10. Interpretation of sub-directories from Figure 9: Sub-directory "240" has 6 sub-directories, 5 of which have very similar structure, the arrowed one is the exception. "278" has many more sub-directories than "240" and some of them are similar, for example those labelled "A" and "B". "code" has children which are leaves (files). "277" has children of different sizes. 5 of the largest children are similar, but the child labelled "C" is slightly different in the composition of C++ source files and the child labelled "D" is topologically different from the rest.

this classification aids the understanding of the visualization of graphs using RINGS.

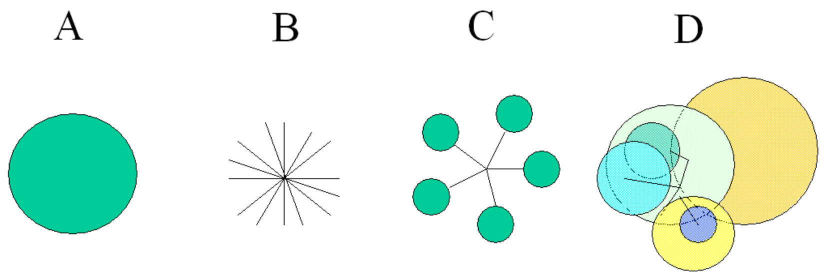

Fig. 11. Classification of basic patterns: A) A Solid circle indicates that this node has numerous children of similar size. B) A pattern of radiating lines indicates that the children are leaves. C) Radial lines with large end-points indicate that children are not leaves. D) Many overlapping circles and lines in the expanded view indicate that this node has many very deep descendents. In practice, as observed in all the graphs presented, most nodes in a graph exhibit mixed patterns. However, this simple classification leads to more intuitive interpretation of the graphs.

6 Conclusion and Future Work

We have presented a new ringed circular layout algorithm to visualize large graphs. The main contribution of RINGS is its efficient use of limited display sizes by showing more distinguishable nodes at one time. Another important feature of RINGS is its capability to show more contextual information without compromising the clarity of the area in focus. A useful feature of RINGS is to show areas in secondary focus.

RINGS uses appropriate visual cues for the user to effectively understand the topology of the graph. We have demonstrated the effectiveness of RINGS in visualizing specific information embedded in the Unix file directory. RINGS is able to highlight sub-directories with large files and/or large number of files. RINGS is also able to locate specific files and show where they are in relation to the rest of the directory.

RINGS is a general graph visualization technique, not limited to the visualization of directory trees. In the future, we plan to apply the technique to the visualization of a variety of hierarchical data, as well as general graphs with spanning trees. For example, in our current work, we are also applying RINGS to the visualization of network connectivity information. By mapping traffic and network data flow information onto the network topology, we believe that RINGS will provide insights into complex, possibly hidden, network behavior. Finally, we plan to conduct a user study and present the results in the final paper.

References

1. E.H. Chi and S.K. Card. Sensemaking of Evolving Web Sites Using Visualization Spreadsheets. *Proceedings of the Symposium on Information Visualization (InfoViz'99)*, pp. 18–25, 142. IEEE Press, 1999.
2. E. Kleiberg, H. van de Wetering and J.J. van Wijk. Botanical Visualization of Huge Hierarchies. *Proceedings of the Symposium on Information Visualization (InfoViz'01)*, pp. 87–94. IEEE Press, 2001.
3. M. Kreuseler, N. Lopez and H. Schumann. A Scalable Framework for Information Visualization. *Proceedings of the Symposium on Information Visualization (InfoViz'00)*, pp. 27–36. IEEE Press, 2000.
4. G. Melancon and I. Herman. Circular Drawing of Rooted Trees. *Reports of the Center for Mathematics and Computer Sciences*, Report number INS-9817, available at: http://www.cwi.nl/InfoVisu/papers/circular.pdf, 1998.
5. T. Munzner. H3: Laying out Large Directed Graphs in 3D Hyperbolic Space. *Proceedings of the IEEE Symposium on Information Visualization (InfoViz'97)*, IEEE CS Press, pp. 2–10, 1997.
6. G.G. Robertson, J.D. Mackinlay and S.K. Card. Cone Trees: Animated 3D Visualization of Hierarchical Information. *Human Factors in Computing Systems, CHI'91 Conference Proceedings*, ACM Press, pp. 189–194, 1991.
7. M. Sarkar and M.H. Brown. Graphical Fisheye View of Graphs. *Human Factors in Computing Systems (CHI'92) Conference Proceedings*, ACM Press, pp. 83–91, 1992.
8. B. Shneiderman. Tree Visualization with Tree-maps: A 2D space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, September 1990.