Navigation and Interaction in Graph Visualizations

Andre Suslik Spritzer[1]

Carla M. D. S. Freitas[1]

Resumo: Grafos são bastante utilizados em muitas áreas e diversas aplicações requerem sua visualização. A visualização de grafos é baseada em técnicas para desenho de grafos, interação e navegação, de forma que se auxilie o usuário a encontrar e manipular informações de forma eficiente. Essas técnicas, que podem ser bi ou tridimensionais, dependendo da metáfora espacial usada para representar o grafo, podem ser combinadas de muitas maneiras diferentes de forma a se adequar às necessidades de uma aplicação em particular. Este artigo apresenta uma visão geral do tema "visualização de grafos".

Abstract: Graphs are widely utilized in many fields and several applications require their visualization. Graph visualization is based on techniques for graph drawing, interaction and navigation in such a way that helps the user in finding and manipulating information efficiently. These techniques, which can be two or three-dimensional, depending on the spatial metaphor used to represent the graph, can be combined in many different ways in order to fit a particular application's needs. This paper presents an overview of the field of graph visualization.

1 Introduction

Due to its intuitive use, simple, easy to understand definition and power as a modeling and representational tool for relational data, graphs have become the structure of choice for representing information in many fields, ranging from software engineering, databases and artificial intelligence to medical science, biology, chemistry and many others.

Many of the applications that use graphs require them to be visualized and manipulated so that the information they represent can be better comprehended and utilized. Thus, graph visualization plays an important role in the field of Information Visualization.

One of the most important aspects of a graph visualization technique is the drawing of the graph itself [8]. The development of a drawing algorithm is a very complex and challenging problem and there is a large community of researchers dedicated to it. Much effort

[1] Instituto de Informática, UFRGS, Caixa Postal 15064, 91.501-970 Porto Alegre
{spritzer, carla@inf.ufrgs.br }

has already been directed to this issue, with many techniques available especially for two-dimensional (2D) drawings of graphs, which are used in many 2D visualization tools.

Most techniques for visualizing graphs fall into the two-dimensional category. Such visualizations rely on a very solid base of techniques for drawing, interaction and navigation. For small graphs, these techniques are suitable, having been used in practice in many applications – from the visualization of a file system, such as Microsoft's Windows Explorer, to software engineering, network visualization tools and even animation packages, such as Alias's Maya. However, the same cannot be said for large graphs.

The visualization of a graph is only as useful as its readability, and with 2D techniques it is often the case that with large and too complex information sets the screen results in a cluttered visual representation. There are several approaches to solving this scalability problem, with one of the most natural, but yet least used in practice, being the use of three-dimensional (3D) visualizations.

On the contrary of 2D techniques, there are no well established standards for interaction with 3D visualizations. The development of user interfaces for 3D applications is still largely experimental, with only a few generically applicable guidelines and recommendations. Therefore, although there has been some progress in the development and research of 3D techniques for the visualization of graphs, advances in this subject are still far from its 2D counterpart.

Taking all of this into account, we present here an overview of graph visualization and introduce a prototype for a tool that is currently being developed, based on force-directed algorithms. Throughout the overview, special attention is given to interaction and navigation techniques, and emphasis is placed more to the results obtained with a technique than on its technical details, which would require a longer description.

We start with a brief background on graph drawing (Section 2), which is followed by overviews of 2D and 3D techniques (Sections 3 and 4, respectively). We then cover some toolkits that were created to aid the development of graph visualization software while providing some conclusions (Section 5).

2 Graph Drawing

Graph drawing algorithms are of course essential to graph visualization techniques. Here we present a short background on the subject. For a more detailed view of graph drawing, the reader can refer to Di Battista et al.'s book [8].

The problem that graph drawing techniques attempt to solve can be very simply stated: given a graph, calculate the position of its nodes and the curve to be drawn for each edge [33]. Finding a solution to this problem, though, is not as trivial[2].

For 2D drawings, many algorithms have been developed and are widely used in graph visualizations. Each algorithm takes into account different aesthetic criteria, such as number of bends in an edge, amount of line crossings and the size of the area occupied by the drawing, and very often will only work on specific types of graphs.

Graph drawing algorithms usually define a rule, or drawing convention, that the generated drawing must adhere to. These conventions might be very complex requirements, but they are usually quite simple, with some of the most often used being the polyline, straight-line and orthogonal drawings (Fig. 1). Often constraints are placed on the drawing, such as forcing the algorithm to always place a certain vertex at a certain position, clustering (placing a subset of nodes close together according to certain criteria) or drawing a certain subgraph with a predefined shape.

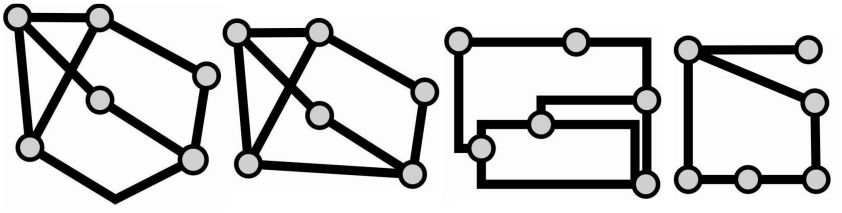

Figure 1. Common graph layouts. From left to right: polylines, line segments, orthogonal and grid layout.

It is very important that a graph drawing algorithm be efficient, so the user can interact with the graph in real-time. It is also desirable that a graph drawing algorithm be predictable, so that two different runs with the same or similar graphs should produce similar visual representations [33]. This way the user can maintain a consistent "mental map" of the drawing.

There is no graph drawing algorithm that could be considered the best – as said before graph drawings effectiveness depends on their readability, and that changes depending on how (and where) the technique is being applied. User studies show that a drawing is more readable if it has no edge-bends, no edge-crossing and small edge-lengths, occupies small screen-space, displays symmetries and satisfies application-specific constraints such as clustering and

[2] A large and active community has evolved around graph drawing. There is a yearly Graph Drawing conference, several books on the subject and sessions in major conferences, such as the Information IEEE Information Visualization symposium.

alignment [7][13][50]. It is usually the case that those readability criteria cannot all be satisfied simultaneously so trade-offs must be made.

Graph drawing is usually associated to 2D layouts and, consequently, to 2D visualization techniques. However, when the graphs are too large, 2D visualizations often have problems with screen clutter. One possible solution is to use 3D visualizations, where the layouts are based on 3D geometrical primitives. With the extra third dimension, problems such as edge crossings, for example, are easily avoided. Nevertheless, even though 3D visualizations may seem to solve many of the issues faced by 2D techniques, they come along with their own set of problems. The next two sections surveys 2D and 3D graph visualization techniques.

3 2D Graph Visualization

There is a great variety of 2D layouts for graph drawings to fit many different aesthetic needs. Techniques range from tree-specific and hierarchical approaches to layouts generated by a physics-based metaphor. An interesting classification of layouts is by Mutzel et al. [43]. Detailed descriptions of the algorithms of most of the below mentioned techniques can be found in [8].

Layouts specific for tree visualizations are among the most used in practice, due to the frequency such structures are found in many diverse applications, ranging from a file system to a website's structure. Examples are the *indentation* layout, the *Reingold-Tilford* layout, the *radial* layout, the *Balloon Trees*, the *Treemaps* and the *hyberbolic* layout.

The indentation layout (Figure 2) is one of the most commonly found, being used typically to represent file structures. All items are placed along vertically distributed rows, with indentation used to show parent/child relationships. They are often used with texts, being Microsoft's Windows Explorer one of its most prominent examples. The major problem of the indentation layout is that it fails to provide an overview of the system and often requires a lot of scrolling from the user. Nodes can be expanded and contracted, allowing hiding child nodes, when contracted, and showing the immediate children, when expanded. This can minimize that latter problem, but will not solve it.

The Reingold-Tilford layout is one of the most well known tree drawing algorithms designed for binary trees [52] and later extended for trees of any degree [62] (Figure 3). The algorithm maps the depth level to the Cartesian y-coordinate and aims at keeping the layout free of edge crossings, preserving the symmetry and ordering of the graph. It produces a compact layout, and draws isomorphic subtrees identically (isomorphic subtrees are kept isomorphic in layout).

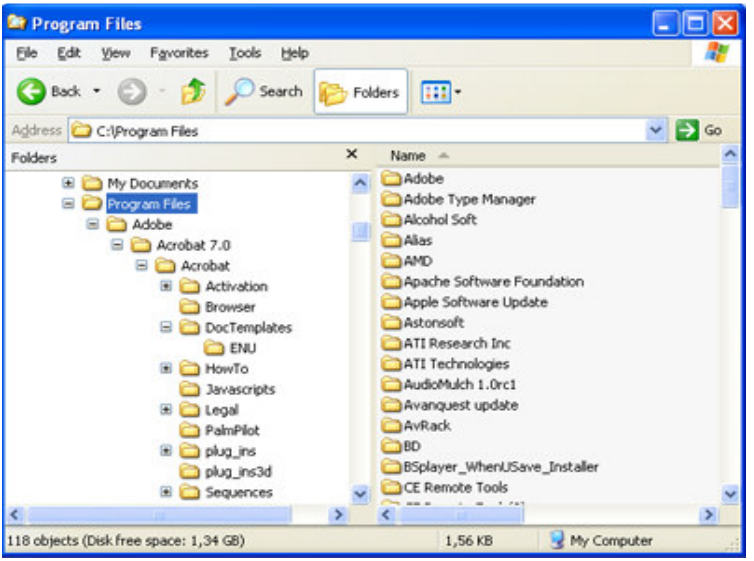

Figure 2. Indentation layout.

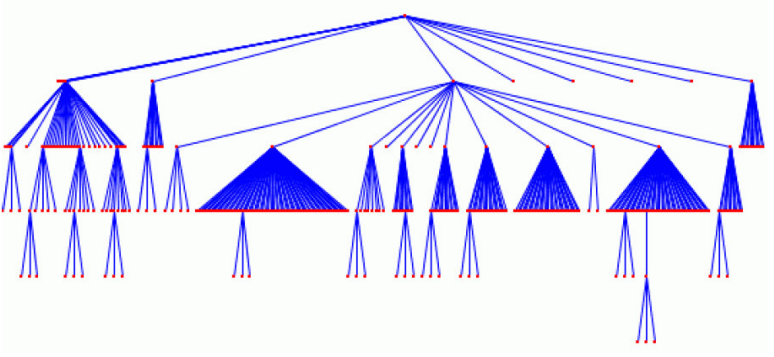

Figure 3. Reingold-Tilford algorithm.

The radial layout (Figure 4a) was proposed by Eades [17] as a variation of the H-tree layout [56], which was designed for general graphs instead of just binary trees. This layout places nodes in concentric circles according to their depths in the tree, with each circle's radius representing one level and the root always located in the center of all circles. The radial layout can be thought of as a node-link diagram laid out in polar coordinates.

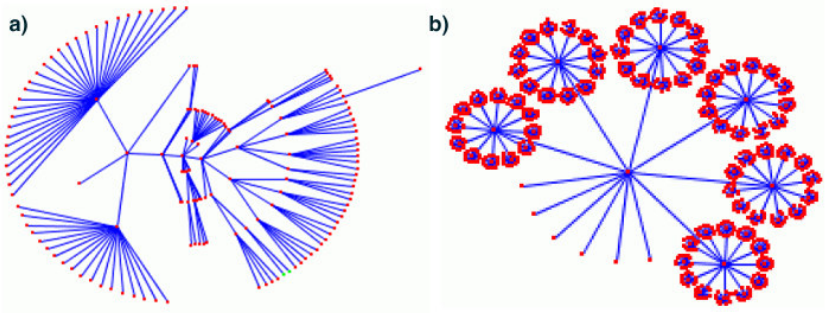

Figure 4. a) Radial layout; b) Balloon Trees layout.

The Balloon Trees layout (Figure 4b) is especially interesting for being one of the few occasions in which a 3D algorithm was adapted to 2D [12]. This technique consists of a 2D version of the classic Cone Trees algorithm [53], consisting basically of a planar projection of a cone tree, with some modifications made so that the projected circles do not overlap.

One of the most well known techniques in the field of Information Visualization, the Treemaps layout [57] is very interesting for many reasons. It intuitively communicates the hierarchy of a tree without using the regular node-link approach and is able to give an overview of an entire large tree to the user. It has been used in many applications, from visualizations of file systems [57] (figure 5) to stock market data [63][35], and used as basis for many new techniques[3].

Perhaps one of the most interesting tree visualizations due to the effects of interacting with it is the hyperbolic layout (Figure 6), which builds the layout on a hyperplane and project the result back into Euclidean geometry. It results in a view of the graph [38][39] with fish-eye effect [23]. Hyperbolic visualization allows for visualizing more data than regular techniques, thus being more suitable for large graphs. In this technique, nodes far from the center are shown in small size and lines are projected as curves. Other works with similar results include Robinson's [54] and Wilson and Bergeron's [64]. There are also 3D extensions of this approach.

[3] The University of Maryland still maintain a website with the latest versions and adaptations, history and publications at http://www.cs.umd.edu/hcil/treemap.

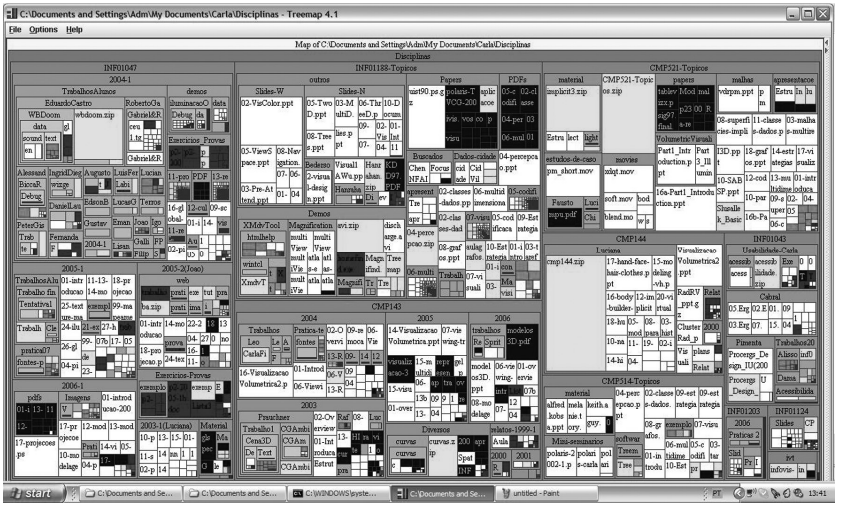

Figure 5. Treemaps view of a directory.

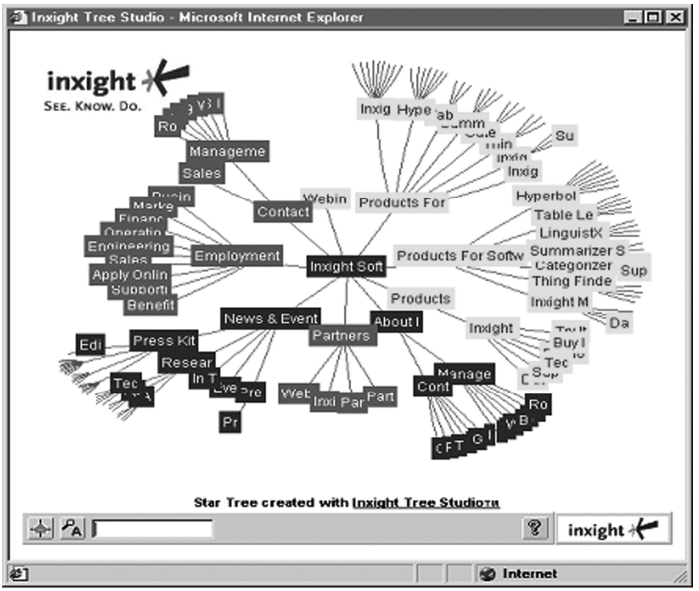

Figure 6. Hyperbolic Layout provided by Visual Inxight.

For the drawing of general graphs there are several approaches that can be taken. Some of which being the application of a tree layout on a spanning tree, the *hierarchical* or *Sugiyama* layout, the *optimization-based* layouts, the *adjacency matrix* layout and the *attribute-driven* (or *structurally-independent*) layout.

Sugiyama's hierarchical layout for directed graphs (Figure 7), also known as the "dot" algorithm, is one of the most important works in the field of graph drawing [60]. In this algorithm, nodes are assigned to hierarchically organized layers according to a chosen layering technique. Once the nodes have been assigned to layers, they are positioned within the layer in a way that minimizes the edge-crossings between two consecutive layers. This minimization of edge-crossings makes the Sugiyama algorithm computationally expensive and was proven to be NP-hard [25]. The Sugiyama layout is not adequate for cyclic graphs, but shows very good results with directed acyclic graphs. One way to minimize this problem is to use a layering technique that removes cycles.

Figure 7. Sugiyama layout

The optimization-based layout algorithms rely on a series of mathematical equations that act as constraints for the drawing and are passed to a "solver", which tries to optimize them. Such constraints might be the minimization of edge crossings and edge bends, for instance. Optimization-based layouts tend to generate organic-looking layouts that can be applied on all types of graphs and are very customizable. They are also interesting for allowing the user to see the graph being built in a real-time animation that can be obtained by rendering the intermediary steps of the layout building process. The main drawbacks of this type of layout are that it tends to be very computationally expensive (especially with larger graphs) and that it is not deterministic. Also, sometimes the organic look is not desirable, depending on the application.

By far the most popular and widely used class of techniques following the optimization-based approach is the *force-directed* or *spring* layout (Figure 8), which was first proposed by Eades [16]. The force-directed layout used a physics metaphor, with nodes acting as physical bodies and edges as springs. Nodes exert a repulsive force and a drag force or some amount of friction is applied to the model. This algorithm results in an optimization process, since it minimizes the energy within the physical system. One of the main advantages of this technique is that it is easy to implement and generally applicable. Its main drawback is that it is potentially very slow and tends to produce unpredictable results, although Bertault managed to make a more deterministic version by preserving edge crossings [9].

Many approaches, versions and adaptations of force-directed algorithms exist, including Kamada and Kawai's algorithm [36], the Fruchterman-Reingold algorithm [21], Frick et al.'s layout [20], Davidson and Harel's method [14], and more recently the DiG-CoLa technique by Dwyer and Koren [15], which allows the application of force-directed algorithms to directed graphs.

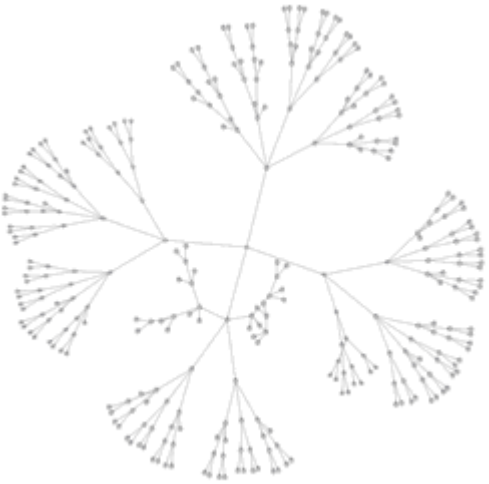

Figure 8. Force-directed layout. Reproduced from aiSee gallery (http://www.aisee.com/)

The attribute-driven or structurally-independent layout is based on simply ignoring the graph's node-link structure and using the attributes of the data instead. These algorithms are often fast and make the communication of particular features of the graph quite clear, but they often do not present its structure very well. An example of a structurally-independent layout is CAIDA's Skitter tool [10], which has a radial scatterplot of internet connectivity, with the angle representing the longitude and the radius, the degree (number of connections).

The adjacency matrix approach consists of showing the graph as a table with the nodes as its rows and columns and the edges as filled table cells. Matrix visualizations are not as intuitive as node-link diagrams, but they have proven themselves very promising, since they eliminate all occlusion and scalability problems that are often found in node-link diagrams. When the nodes are ordered following some criteria, the visualization as an adjacency matrix allows a rapid perception of the relationships between nodes, since there are no edge crossings. This technique might be extremely helpful in visualizing large graphs.

Ghonien *et al.* [26] provide an interesting comparison of these two types of graph visualization. Matrix visualization can also be used in conjunction to node-link diagrams (Figure 9), in order to help the user with filtering and searching while also providing the traditional and intuitive view of the graph [30]. Other applications of matrix visualization include the system developed by Abello and Korn for visualizing multidigraphs [1], Abello and van Ham's Matrix Zoom [2] and Ghonien, Jussien and Fekete's VISEXP [27]

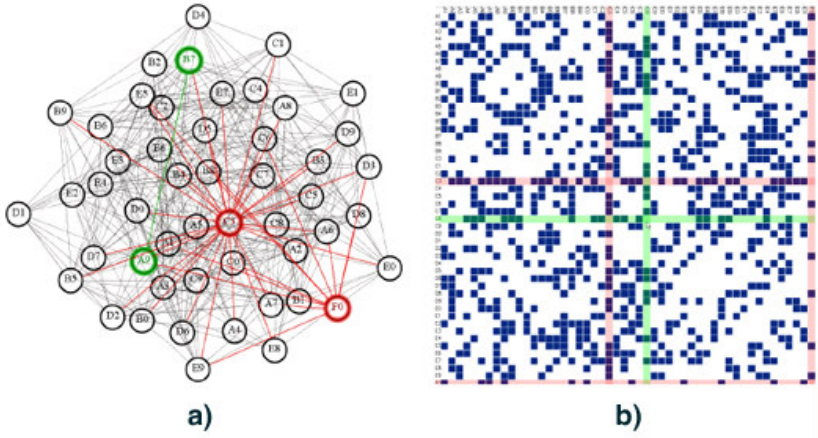

Figure 9. A traditional node-link diagram (a) and its matrix representation (b) [26]

Another interesting approach to 2D graph drawing is the *spectral layout* technique, which use as coordinates the eigenvectors of a transformation matrix derived from the adjacency matrix of the graph.

There are also methods known as *hybrid techniques*, which take elements from several techniques and combine them in order to create a new layout that uses the advantages of one approach to deal with the disadvantages of another. Some recent techniques that fit into this category are the *Voronoi Treemaps* [6], the *Elastic Hierarchies* [65] and the previously mentioned DiG-CoLa.

For relatively small graphs, traditional 2D techniques work quite well, producing satisfactory results. As the graphs grow larger, however, the screen tends to be cluttered with visual information and the user has to do a great amount of scrolling, panning and zooming to explore the graph. Also, it becomes harder for the user to get a general overview of the graph and to maintain a solid "mental map". The more nodes, too, the more computationally expensive an algorithm tends to be, and more cluttered scenes result. One possible solution to this problem is the use of 3D graphs.

4 3D Graph Visualization

Characteristics of 2D graph layouts deemed problematic often become non-issues with 3D visualizations. The minimization of edge crossings, for instance, is a very important concern in 2D techniques, but is much less important in 3D. In 3D, edges are a lot less likely to intersect and even if some appear as intersecting, they will not be perceived as such due to motion parallax, shading or stereoscopic disparities that allow the perception of depth. However, even though 3D visualizations may seem to solve many of the issues faced by 2D techniques, they come along with their own set of problems.

On the contrary to 2D visualizations, there are not any well established standards and techniques for display, interaction and navigation that 3D visualizations can count with. Also, 3D visualizations cannot count on the 2D standards. Therefore, the development of user interfaces for 3D applications is largely experimental, based mostly on a few guidelines and recommendations that can be applied to some cases, but not in others.

One of the most common approaches to creating 3D visualization techniques is the extension of 2D techniques. Among these techniques, are 3DCube [48], Information Cube [51], Narcissus [29], NV3D [47] and Pajek [46].

3DCube (Figure 10) is an extensible platform aimed at the evaluation, development and refinement of 3D graph drawing algorithms. Supporting many different algorithms and being easily extensible, the platform provides a 3D environment that the user can navigate by rotation, shifting and zooming. It is a relatively old platform, being one of the first dedicated to 3D graphs and its focus is mostly on graph drawing and not graph visualization, so interaction and navigation are not well designed. Thus, even though quite interesting from a graph drawing perspective, 3DCube is not adequate for end-user applications.

Information Cube (Figure 11) is a technique for visualization of hierarchical information [51]. In this technique, information is represented in translucent, nested cubes, with the outermost cube corresponding to the top level data and containing cubes representing the next level, which also contain cubes representing the following level, and so on. The visualization is displayed in 3D, either on a head-mounted display or on a conventional monitor, and navigation and interaction (rotating, moving, selecting and zooming in on cubes) is performed with hand movements detected with the user wearing a DataGlove. If the

necessary hardware is available, the technique can be displayed in stereo, thus providing the user with a better insight of the 3D structure.

NestedVision3D, or NVD3D [47], is a system for three-dimensional visualization of large, hierarchical graphs obtained from the structure of computer programs. Like in Information Cube, nodes are also represented by boxes that can contain subgraphs nested to an arbitrary depth and relations are simply lines. Navigation in NV3D's 3D environment is performed through widgets, which are 3D objects rendered as part of the scene.

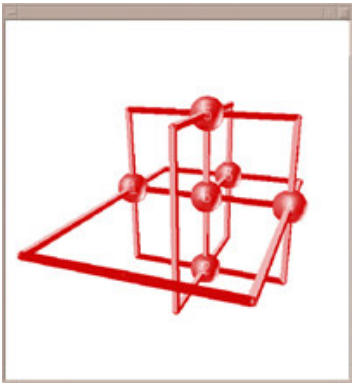

Figure 10. 3D Cube [48].

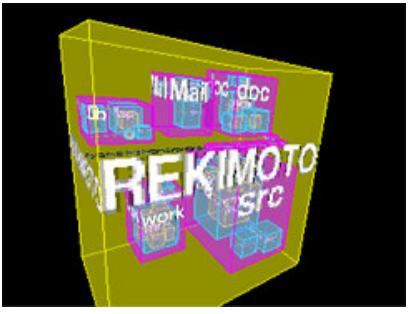

Figure 11. Information Cube [51].

Narcissus [29] is a 3D graph visualization system that aids the user in navigating and exploring the information space of other applications. Narcissus was implemented as a process that communicates with applications (such as web browsers or programming environments) and displays a 3D force-directed visualization of the graph generated with the collected data.

Besides these extensions of 2D techniques, there are also techniques that were designed exclusively for 3D, even though adopting some ideas tailored for 2D drawings. Among these, we find the classic ConeTrees [53], H3 [40], the SGI File System Navigator [58], Nodes3D [61] and Walrus [34]. It is interesting to note that all of these techniques deal with tree-like structures instead of general graphs.

Originally proposed in 1991 by Robertson et al, the Cone Tree [53] is one of the classic techniques in Information Visualization. In this technique, nodes are placed at the apex of a cone and its children are placed evenly spaced along its base. When a node is selected by a mouse click, the cone tree rotates so that this node and each node in the path from this node to the top are brought to the front and highlighted. The rotation is made smooth through animation so the user can maintain the mental map of the tree.

Tamara Munzner's H3 [40][41] is a technique for drawing node-link diagrams of large directed graphs in 3D hyperbolic space, which allows for the visualization of much larger structures than the traditional techniques (Figure 12). The technique works by computing the layout according to the hyperbolic metric, which increases exponentially as opposed to the familiar geometric increase of Euclidean space. The algorithm expands the cone tree layout for 3D hyperbolic space by placing children on a hemisphere around the cone base instead of its perimeter. The effect is a fisheye-like focus+context view of the structure, which allows the visualization of thousands of nodes with minimal screen clutter.

Figure 12. H3 Hyperbolic Layout [41]. Image © IEEE reprinted with permission.

The SGI File System Navigator [58][4] is a 3D file system navigator for IRIX 4.0.1+. Directories are represented by a pedestal with heights proportional to the size of the files in them. The directories are connected by wires, on which the user travels. Files are represented as boxes on top of each directory with the height of the box proportional to the size of the file and the color representing the age.

Nodes3D [61] is an open-source 3D graph visualization application developed to explore neuro-anatomical data although it can be used with general graphs. Nodes are represented as spheres with sizes proportional to their degree. Edges are lines (or arrows, in case of directed graphs) with adjustable thickness. Navigation consists on using the mouse to zoom in and out and rotate the graph and several keyboard shortcuts.

Based on H3 and written in Java and Java3D, Walrus [34] is an open source 3D visualization tool for directed graphs that employs a fisheye-like distortion through a hyperbolic layout to simultaneously display local detail and the global context. The layout is computed based on a spanning tree provided by the user and input graphs ideally should be hierarchical. Graphs are rendered inside a sphere that contains the Euclidean projection of 3D hyperbolic space, with points within the sphere magnified according to their radial distance from the center (objects become larger the closer they are to the center and smaller the closer they are to the boundary).

As could be observed, although 3D graph visualization techniques have been developed along the years to overcome the problems of 2D graph drawing techniques, the problems introduced with objects occlusion and mainly with interaction in 3D space seem to be prevented their widespread use. As one will notice in the next section, even for large graphs, 2D visualization techniques are still far more used than the 3D ones.

5 Visualization of Large Graphs

With the current growing interest in the visualization of massive graphs, such as large social networks, online libraries, encyclopedias and databases, the study and development of new visualization and navigation techniques for large graphs have been receiving much attention. Several authors have had the problem of large graph visualization as motivation for new techniques [3][30][49].

Creating a visualization technique that allows the user to find specific details of information contained in one node of the graph while at the same time giving not only an overview of the entire structure, but also quite possibly views of subgraphs and the relationships between that node and other nodes and clusters of nodes, is far from being trivial. Hybrid layouts and interaction and navigation techniques might provide possible solutions to

[4] FSN usage can be observed in a scene of the film Jurassic Park.

this problem. Other alternatives might be coordinating (Figure 13) or integrating different visualizations, as in MatrixExplorer [30] and MatLink [31], respectively, or adding another dimension to the drawing and rendering it in 3D.

Many techniques for visualization of large graphs rely on methods for grouping together related nodes in order to reduce screen clutter. This process is known as *clustering* and is usually of two types: *structure-based* or *content-based*. Structure-based clustering utilizes the graph structure to group elements, while content-based clustering uses the semantics of the graph [33]. A structure-based clustering technique can be more generally applied since it does not depend on application-specific information, while a content-based technique can generate the best results for its application (possibly even being used in combination with structure-based clustering) but might not be general enough to use in other contexts.

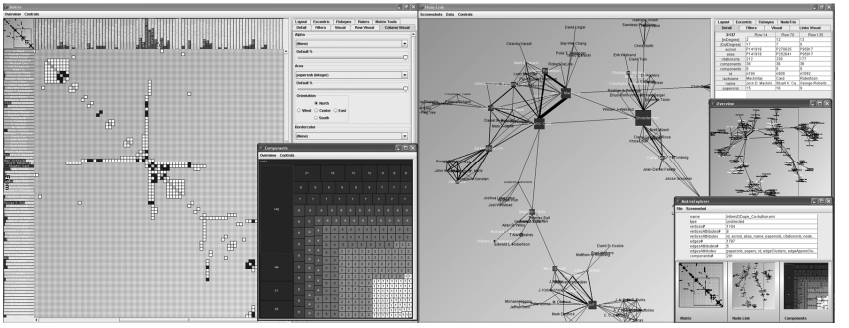

Figura 13. MatrixExplorer [30], integrating matrix and node-link representation. Image courtesy from N. Henry e J-D. Fekete.

Clustering works by allowing the use of a single on-screen element to represent several of the graph's nodes, making possible, for example, the use of different levels of detail that can be accessed by zooming in and out of the graph. Clustering also helps the user identify desired relationships that otherwise could not be seen. Clustering techniques can be either an inherent part of the visualization, such as in MatrixExplorer [30] and MatLink [31] and ASK-GraphView [3] or independent methods that can be used with another technique. They are frequently employed to help with filtering and searching a graph.

Other approaches to large graph visualization include hybrid techniques and hyperbolic visualizations, such as the previously mentioned *Elastic Hierarchies* [65], the works by Munzner [42][40][41], and the utilization of incremental exploration and navigation techniques.

6 Interaction and Navigation in Graphs Visualizations

In Information Visualization, interaction and navigation are as important as the visualization of the data itself, especially when we are dealing with (possibly large) graphs. Finding desired information on a graph can be almost impossible sometimes without the proper navigation and interaction tools.

Techniques for interaction and navigation 2D graph visualization have evolved from generally applicable human-computer interaction (HCI) methods, being thus able rely on all the well known HCI standards. Many different techniques have already been developed, sometimes being an integral part of the visualization technique itself, supporting all user's tasks. It is important to note that both navigation and interaction present a related, but very different problem from graph drawing, with most researchers focusing either in one or the other. In order to produce an efficient visualization, the two areas should be combined, since they are quite dependent on one another to work properly.

Some of the most important categories of interaction techniques are *filtering*, *search*, *selection*, *scrolling/panning* and *scaling/zooming*. Recently, with the surge of interest in visualizing large and complex graphs *incremental exploration and navigation* has also become essential for many applications.

Filtering allows the user to define a set of criteria that will be used by the visualization system to determine what will be displayed on the screen, after filtering the information to a subset that is of interest to the user. Filtering allows for faster finding of desired information by the user due to less visual complexity, being thus very useful to deal with problems generated by visual clutter and occlusion that tend to happen with large datasets. The criteria chosen by the user for the filtering operation might be anything from the graph's structure to a particular attribute of the nodes. North [45] and Gansner's [24] work describe the use of filtering in graph visualization applications.

Searching allows the user to easily find information on the visual representation of the graph by providing some criteria as input to the search mechanism, which usually returns the node that best matches the search parameters by visually focusing on it somehow. *ConeTrees* [53] and *Hyperbolic Broswer* [38] techniques are some of the first techniques that implemented this feature common to all visualization systems nowadays.

Selection provides the user with a mechanism to select certain information from the dataset. When an element is selected, it is usually visually highlighted from the rest of the dataset, with more information often being displayed about it. Selection is one of the most common interaction techniques and is closely related to search mechanisms (which may return a certain element selected). Selection also serves as basis for tools for moving and manipulating elements.

When the drawing of the graph occupies more than the size of the screen, it is often useful to apply the classical zoom and pan tools. Zooming allows a visualization to be viewed in more detail and has two distinct variations: *geometric* and *semantic* [33]. Geometric zooming simply scales up the drawing while semantic zooming shows more detailed information when approaching an area of the graph, thus requiring some sort of level of detail control.

One approach when dealing with large graphs is the use of incremental exploration and navigation techniques, which display only a small part of the graph at a time, showing other parts as they are needed. This approach allows the use of layout techniques that otherwise might be too expensive for real-time interaction, since it would only be applied to a relatively small subgraph at a time. These techniques are useful for exploration of graphs that are not known in their entirety, such as the World Wide Web.

Incremental exploration of large graphs can be enhanced through the user of *focus+context* technique. Usually an intrinsic part of the visualization itself, focus+context techniques allow a detailed view of a desired subset of information while at the same time showing the larger context where that information is located with a reduced level of detailed [11]. As the user's focus of interest changes, the highlighted subset may also change correspondingly.

One of the classical focus+context techniques is the *fisheye view* (Figure 14), which shows a larger, detailed view of the desired information subset (possibly centered on the screen) while also displaying the rest of the information surrounding it (the context) in progressively less detail [23]. The fisheye view is based on mapping the graph to a plane, defining a focus point and applying a degree-of-interest function on every node with their distance from the focus point as parameter.

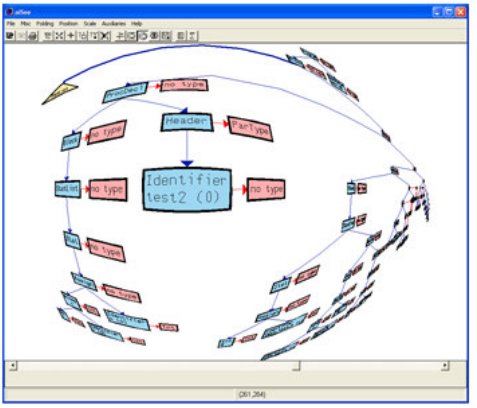

Figure 14. Fisheye view of a graph in AbsInt's aiSee [4]

There are many variants of the fisheye view technique, mostly divided into two categories: filtering and distortion [44]. Filtering fisheye views work by using the degree-of-interest function to display only the most interesting or relevant elements and are closer to what Furnas suggested. By far the most commonly found, the distortion techniques work by using a degree-of-interest function to geometrically distort the visualizations in order to obtain a fisheye lens look. There are also techniques with more than one focus point, such as the one from Storey [59].

The fisheye view can be applied to any visualization independent of its layout algorithm, but that is not the case with all focus+ techniques. The previously mentioned hyperbolic layouts [38][40] merge the fisheye-like visual distortions with the layout itself. This layout technique was developed with focus+context in mind, so it becomes an integral part of the layout algorithm instead of a separate stage applied later to the drawing.

7 Tools and Applications

There is a vast multitude of applications in a great variety of fields that utilize graph visualization. We will focus on a few interesting applications that provide a good idea of the different techniques and their many possible uses and variations.

AbsInt's aiSee is a tool specifically for graph visualization that allows the user to customize the layout of the graph given as input [4]. Some of its features include zooming and panning, subgraph nesting and different layout algorithms (including force-directed layout), cartesian and polar fisheye views, and handling of huge graphs (one million nodes). It has been used for software and network visualization, gene/protein/keyword relationship mining, stack usage analysis for real-time, safety-critical systems, and many other applications, both by commercial users and for academic research.

Developed at the IBM Visual Communication Lab by Abello et al., ASK-GraphView [3] is a system for visualizing large graphs. Sophisticated clustering algorithms are applied to the input graph, which is displayed in the main window as a node-link diagram with cluster nodes (nodes that actually represent groupings of nodes – subgraphs) that can be expanded and collapsed. The generated graph can be interactively explored and filtered by the user. Other works [18][19] have used a similar approach to visualizing large graphs, but they require the input graph to have a hierarchy, which is seldom the case in practice.

MatrixExplorer [30] is a tool for social network analysis based on the coordination of two different graph visualizations: matrix representation and node-link diagram. The matrix representation is used for reorganizing, clustering and filtering graphs while the node-link diagram is used to better visualize exploration results (Figure 13). MatrixExplorer was developed based on detailed user studies with professional social science researchers, who are the target audience for this type of application. In recent works [31][32], the same authors

have provided more powerful tools where they combine matrix representation with node link diagrams. In MatLink [31], edges are drawn superimposed to the matrix representation, linking the nodes represented as headers of columns, while in NodeTrix [32], subgraphs are represented as matrices and linked to other subgraphs by edges, forming a hybrid node(matrix)-link diagrams.

Vizster [28] produces an animated, interactive, incremental, force-directed node-link visualization of a graph of a large-scale online social networking service. Viszter aims at playful end-user exploration and navigation of the Friendster (see http://www.friendster.com) network by the end-user. In Viszter, nodes represent members of the system and links represent the "friendship" links between them.

TouchGraph relied on the implementation of a force-directed layout to create an interesting tool now applied to the visualization of networks and search engine results (see http://www.touchgraph.com/). Figure 15 shows an example of the visualization produced with this tool. It can be used to represent results from search with Google, browse through Facebook as well as to browse through Amazon.

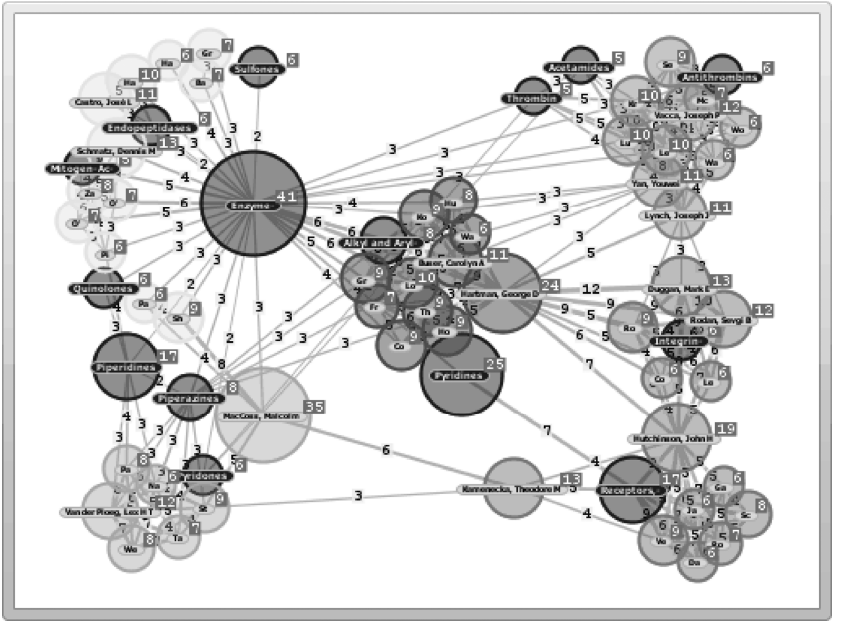

Figure 15. Example of visualization built with TouchGraph (extracted from the site).

AmazNode produces an animated, incremental, force-directed node-link visualization of the Amazon.com online store [22]. After the visitor gives a product as input, the system

creates and displays the graph, recursively searching the products based on the criteria "customers who bought this product also bought that product".

Pajek [46] is a widely used tool for visualization of large networks. It can be utilized in areas ranging from social network analysis to chemistry, biology, citation networks, etc. Pajek aims at allowing large networks to be visualized by recursively decomposing them into many smaller networks while providing powerful tools for visualization and analysis of such structures. The system also supports the visualization of temporal networks.

There is also a technique proposed by Kumar and Garland [37] in which users explore a hierarchical representation of a complex graph constructed by stratifying nodes into different levels so that central and representative nodes in the graph are emphasized. Nodes are organized into interconnected groupings in a tree, enabling tree families to be placed close to each other in the tree layout algorithm. The rendering scheme integrates user interaction with stylistic visual representations to abstract and explore graphs, using the hierarchy to filter edges and nodes and allowing users to drill down into the graph for details. An interesting feature of this technique is its ability to deal with time-varying data, making it particularly useful for the visualization of dynamic graphs. Also, unlike most other graph visualization techniques, modern graphics capabilities such as 3D rendering, shading and alpha-compositing are used in order to achieve clearer results.

Nestor [55] is a web browser based on Internet Explorer that draws a web map (graph) representing the links and documents the user is visiting while navigating. The map provides a visual context so that users know where they are and where they came from, allowing them to easily go back to any previously visited document.

It is interesting to note that several of the above mentioned visualization techniques deal with social networks. In the last few years there has been a growing interesting in visualizing such networks, especially since the popularization of end-user social networking websites such as MySpace, Orkut, Friendster and YouTube.

8 Final Comments

Graph visualization is one of the most promising fields in Information Visualization. Despite the existent long and dense literature on graph drawing, with the recent advances in the volume of information available for diverse users, techniques for visualizing and allowing the extraction of information from such structures are still needed.

Many toolkits and libraries have been developed to aid the creation of graph visualization software, ranging from tools developed as part of academic research to free GPL-licensed libraries and commercial toolkits, supporting many languages and platforms, from C++ and Win32 to Java and .NET. Most allow the drawing of the graphs using several different algorithms and several also support tools for basic interaction and navigation (drag

and dropping nodes, zoom and pan, etc.) as well as graph theory algorithms. A few also provide features specifically for network visualization. Table 1 provides an overview of some toolkits.

The analysis of these tools reveals that there is still room for improvement especially regarding the interactive techniques they provide to the users for navigating in the information space. An important feature also is the possibility of integrating different views of the same dataset in a single or coordinated visualization. This is based on the fact that very often a single layout or visualization is not sufficient for the user to get the necessary insight from the data.

Table 1. Toolkits and libraries.

Toolkit	Availability	Supports	URL / Reference
GDToolkit	Free (for academic use); Commercial	C++	http://www.dia.uniroma3.it/~gdt
GoVisual	Commercial	Win32, .NET, Java, Linux	http://www.oreas.com
Gravisto	Not yet available	Java	http://gravisto.fim.uni-passau.de ; [BACHMAIER 2004]
ILOG Views Graph Layout	Commercial	C++, Java, .NET	http://www.ilog.com/products/views/graphlayout/
Infovis Toolkit	Free for academic use	Java	http://ivtk.sourceforge.net/
JGraph	Free; Commercial	Java	http://www.jgraph.com
JUNG	Free, Open-source	Java	http://jung.sourceforge.net
Tom Sawyer Visualization	Commercial	ActiveX, ASP.NET, Java, JSP, MFC	http://www.tomsawyer.com
Tom Sawyer Layout	Commercial	ActiveX, C++, Java, .NET	http://www.tomsawyer.com
Touchgraph	Free; Commercial		http://sourceforge.net/project/showfiles.php?group_id=30469 http://www.touchgraph.com
Tulip	Free , Open-source (GPL)	C++	http://www.tulip-software.org
yFiles	Commercial	Java	http://www.yworks.com

Acknowledgments

This work has been possible due to the support from CNPq (research grant for C. Freitas, CNPq/INRIA and CNPq/FNRS projects).

References

[1] Abello, J. and Korn, J. "Mgv: a System for Visualizing Massive Multidigraphs" *IEEE Transactions on Visualization and Computer Graphics*, vol. 8(1), pp 21–38, 2002.

[2] Abello, J. and Ham, F. "Matrix Zoom: A Visual Interface to Semi-External Graphs" *Proc. IEEE Symposium on Information Visualization 2004*, pp 183-190, 2004.

[3] Abello, J, Ham, F., Krishnan, N. "ASK-GraphView : A Large Scale Graph Visualization System." In: *Proc. of the IEEE Symposium Information Visualization 2006*, pp. 669-676, 2006.

[4] AbsInt – Angewandle Informatik. "aiSee". At: <http://www.absint.com/aisee/>. Accessed on January 22, 2007.

[5] Bachmaier, C., Brandemburg, J., Forster, M., Holleis, P., and Raitner, M. "Gravisto: Graph visualization toolkit." In: *Proceedings of the International Symposium on Graph Drawing* (GD 2004), 2004.

[6] Balzer, M. and Deussen, O. "Voronoi Treemaps." In *Proc. of the IEEE Symposium on Information Visualization 2005*, pp. 49-56, 2005.

[7] Batini, C., Furlani, L., and Nardelli, E. "What is a Good Diagram? A Pragmatic Approach." In: *Proc. 4th International Conf. on the Entity Relationship Approach*, pp. 312-319, 1985.

[8] Battista, G.; Eades, P; Tamassia, R; Tollis I.G. "*Graph Drawing: Algorithms for the Visualization of Graphs*." Prentice Hall, New Jersey, 1999.

[9] Bertault, F. "A Force–Directed Algorithm that Preserves Edge Crossing Properties". In: *Proceedings of the Symposium on Graph Drawing* (GD'99), pp. 351–358, 1999.

[10] CAIDA – Cooperative Association for Internet Data Analysis, "Skitter." At: <http://www.caida.org/tools/measurement/skitter/>. Accessed on January 24, 2007.

[11] Card, S.K., Mackinlay, J.D., and Shneiderman, B. "*Readings in Information Visualization*", Morgan Kaufmann Publishers, 1999.

[12] Carrière, J., and Kazman, R. "Research Report: Interacting with Huge Hierarchies: Beyond Cone Trees". In: *Proceedings of the IEEE Conference on Information Visualization '95*, pp. 74–81, 1995.

[13] Cohen, R.F., Eades, P., Lin, T. and Ruskey, F. Three-dimensional graph drawing. In: *Proceedings of Graph Drawing* (GD'94), vol. 894 of LNCS, pages 1-11, 1994.

[14] Davidson, R. and Harel, D. "Drawing Graphs Nicely Using Simulated Annealing", *ACM Transaction on Graphics*, 15(4), pp. 301–331, 1996.

[15] Dwyer, T. and Koren, Y. 2005. DIG-COLA: Directed Graph Layout through Constrained Energy Minimization. In: *Proc. of the IEEE Symposium on Information Visualization 2005*, pp. xxx, 2005.

[16] Eades, P. "A Heuristic for Graph Drawing," *Congressus Numerantium*, vol. 42, pp. 149–160, 1984.

[17] Eades, P. "Drawing Free Trees", *Bulletin of the Institute for Combinatorics and its Applications*, pp. 10– 36, 1992.

[18] Eades, P. and Feng, Q.W. "Multilevel Visualization of Clustered Graphs," In: *Proceedings of the 4th Intl. Symposium. on Graph Drawing* (GD'96), LCNS 1190, pp 101-112, 1996.

[19] Eades, P. and Huang, M.L. "Navigating Clustered Graphs using Force-Directed Methods," *Journal of Graph Algorithms and Applications*, 4(3), pp 157-181, 2000.

[20] Frick, A., Ludwig, A., and Mehldau, H. "A Fast Adaptive Layout Algorithm for Undirected Graphs", In: *Proc. of the IEEE Symposium on Graph Drawing* (GD '93), pp. 389–403, 1994.

[21] Fruchterman, T.M.J. and Reingold, E.M. "Graph Drawing by Force–Directed Placement," *Software — Practice & Experience*, Vol. 21, pp. 1129–1164, 1991.

[22] Fukatsu, T. "AmazNode." At: <http://amaznode.fladdict.net>. Accessed on January 22, 2007.

[23] Furnas, G.W. "*The FISHEYE view: a new look at structured files*." Bell Laboratories, 1981.

[24] Gansner, E.R.; North, S.C. "An Open Graph Visualization System and its Applications to Software Engineering." *Software – Practice and Experience*, Vol. 30, n. 11, p. 1203-1233, Sept. 2000.

[25] Garey, M.R. and Johnson, D.S. "Crossing number is NP–complete", *SIAM Journal of Algebraic and Discrete Methods*, Vol. 4 No 3, pp. 312–316, 1983.

[26] Ghonien, M.; Fekete, J.-D. "A Comparison of the Readability of Graphs Using Node-Link and Matrix-Based Representations." In: *Proc. of the IEEE Symposium on Information Visualization 2004*, pp. 17-24.

[27] Ghonien, M., Jussien, N. and Fekete, J.-D., "VISEXP: visualizing constraint solver dynamics using explanations". In: *FLAIRS'04: Seventeenth international Florida Artificial Intelligence Research Society Conference,* 2004, Miami Beach, FL.

[28] Heer, J. and Boyd, D. "Vizster: Visualizing Online Social Networks." In *Proc. of IEEE Symposium on Information Visualization*, 2005.

[29] Hendley, R. J., Drew, N. S., Wood, A. M., and Beale, R. 1995. "Case study: Narcissus: visualising information." In: *Proc. of the IEEE Symposium on Information Visualization 1995*.

[30] Henry, N.; Fekete, J.-D. "MatrixExplorer: a dual-representation system to explore social networks." *IEEE Transactions on Visualization and Computer Graphics* (Proceedings IEEE Information Visualization 2006), September-October 2006.

[31] Henry, N., Fekete, J.-D. "MatLink: Enhanced Matrix Visualization for Analyzing Social Networks". In: *Proceedings of IFIP Interact 2007*, LNCS 4663, p. 288-302.

[32] Henry, N., Fekete, J.-D., McGuffin, M. "NodeTrix: Hybrid representation for analyzing social networks". *IEEE Transactions on Visualization and Computer Graphics*.

[33] Herman, I; Melancon, G.; Marshall, M. S. "Graph visualization and navigation in information visualization: A survey." *IEEE Transactions on Visualization and Computer Graphics*, pp. 24-43, 2000.

[34] Hyun, Y. "Walrus." At: <http://www.caida.org/tools/visualization/walrus/>. Accessed on January 24, 2007.

[35] Jungmeister, W.-A. and Turo, D. "*Adapting treemaps to stock portfolio visualization,*" University of Maryland, Center for Automation Research technical report, 1992.

[36] Kamada, T. and Kawai, S. "An Algorithm for Drawing General Undirected Graphs", *Information Processing Letters*, Vol. 31, pp. 7–15, 1989.

[37] Kumar, G. and Garland, M. "Visual exploration of complex time-varying graphs." *IEEE Transactions on Visualization and Computer Graphics*,

[38] Lamping, J., Rao, R., and Pirolli, P. "A Focus+context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies", In: *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, ACM Press, 1995.

[39] Lamping, J. and Rao, R. "The Hyperbolic Browser: A Focus+context Technique for Visualizing Large Hierarchies", *Journal of Visual Languages and Computing*, Vol. 7 No. 1, pp. 33–55, 1996.

[40] Munzner, T. "H3: Laying out large directed graphs in 3d hyperbolic space." In *Proc. of the IEEE Symposium on Information Visualization 1997*, pp. 2-10, 1997.

[41] Munzner, T. "Drawing Large Graphs with H3Viewer and Site Manager", In: *Proc. of the Symposium on Graph Drawing* (GD '98), pp. 384– 393, 1998.

[42] Munzner, T. and Burchard, P. "Visualizing the Structure of the World Wide Web in 3D Hyperbolic Space", In: *Proc. of the VRML'95 Symposium*, ACM SIGGRAPH, ACM Press, 1995.

[43] Mutzel, P., Gutwengwer, C. , Brockenauer, R., Fialko, S., Klau, G., Kruger, M., Ziegler, T., Naher, S., Alberts, D., Ambras, D. , Koch, G., Junger, M., Bucheim, C., Leipert, S. "A Library of Algorithms for Graph Drawing", In: *Proc. of the Symposium on Graph Drawing* (GD'97), pp. 456– 457, 1998.

[44] Noik, E. G. "Layout-independent Fisheye Views of Nested Graphs." In: Symposium on Visual Language, VL, 1993, Bergen, Norway.

[45] North, S.C. and Koutsofious, E. "Applications of Graph Visualization." In: *Graphics Interface*, Banff, Canada, 1994.

[46] Pajek – Proram for Lare Network Analysis. At: <http://vlado.fmf.uni-lj.si/pub/networks/pajek/>,

[47] Parker, G., Franck G. and Ware, C. "Visualization of Large Nested Graphs in 3D: Navigation and Interaction." *Journal of Visual Languages and Computing*, 9(3) 299–317, 1998.

[48] Patrignani, M. and Vargiu, F. "3DCube: a Tool for Three Dimensional Graph Drawing", in G. Di Battista (editor), *Graph Drawing* (Proc. GD '97), volume 1353 of LNCS, pp. 284-290, 1998.

[49] Pretorius, A.J., Wijk, J.J. "Visual Analysis of Multivariate State Transition Graphs." *IEEE Transactions on Visualization and Computer Graphics*, 12(5): 685-692, 2006.

[50] Purchase, H. C., Cohen, R. F., and James, M. I. "An experimental study of the basis for graph drawing algorithms." *J. Exp. Algorithmics* 2, Jan. 1997.

[51] Rekimoto, J. and Green, M. "The Information Cube: Using Transparency in 3D Information Visualization." In: *Proc. of the Third Annual Workshop on Information Technologies and Systems* (WITS'93), pp. 125–132. 1993.

[52] Reingold and J.S. Tilford, "Tidier Drawing of Trees." *IEEE. Transactions on Software Engineering*, Vol. SE-7, No. 2, pp. 223–228, 1981.

[53] Robertson, G., Mackinlay, J.D., and Card, S.K. "Cone trees: Animated 3-D visualizations of hierarchical information." In: *Proc. of the ACM SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1991.

[54] Robinson, A. "EBI Hyperbolic Viewer,' European Bioinformatics Institute, available at: http://industry.ebi.ac.uk/~alan/components, 1998.

[55] Sawers, J.E. and Zeiliger, R. "NESTOR Navigator: A tool for the collaborative construction of knowledge through constructive navigation." In: R. Debreceny & A. Ellis (eds.) *Proc. of Ausweb99, the Fifth Australian World Wide Web Conference*. Southern Cross University Press, Lismore. p. 396-408, 1999.

[56] Shiloach, Y. "*Arrangements of Planar Graphs on the Planar Lattices*", in Rehovot, Israel: Weizmann Institute of Science, 1976, Ph.D. Thesis.

[57] Shneiderman, B. Tree visualization with treemaps: a 2-d space-filling approach. *ACM Transactions on Graphics*, vol. 11, 92-99, 1992.

[58] Silicon Graphics, Inc. "3D File System Navigator." At <http://www.sgi.com/fun/freeware/3d_navigator.html>, 1992.

[59] Storey, M.–A.D. "On Integrating Visualization Techniques for Effective Software Exploration." In: *Proc. of the IEEE Symposium on Information Visualization*, 1997.

[60] Sugiyama, K., Tagawa, S., and Toda, M. "Methods for Visual Understanding of Hierarchical Systems Structures", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC–11 No. 2, pp. 109–125, 1989.

[61] Trotts, I., Mikula, S., and Jones, E.G. "Nodes3d - Enhanced Interactive Three-Dimensional Directed Graph and Similarity Matrix Visualization" ,2006.

[62] Walker II, J.Q., "A Node-positioning Algorithm for General Trees", *Software — Practice and Experience*, 20(7), pp. 685–705, 1990.

[63] Wattenberg, M. "Visualizing the stock market." In: *CHI '99 Extended Abstracts on Human Factors in Computing Systems*. CHI '99. ACM Press, New York, NY, 188-189, 1999.

[64] Wilson, R.M. and Bergeron, R.D. "Dynamic Hierarchy Specification and Visualization", In: *Proc. of the IEEE Symposium on Information Visualization*, pp. 65–72, 1999.

[65] Zhao, S., McGuffin, M. J., and Chignell, M. H. 2005. Elastic Hierarchies: Combining Treemaps and Node-Link Diagrams. In: *Proc. of the IEEE Symposium on Information Visualization 2005*.