A Survey on Graph Visualization

By
Weiwei Cui

Supervisor
Huamin Qu

Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Table of Contents

Abstract

Graph visualization helps users gain insight into data by turning the data elements and their internal relationships into graphs. Graph visualization leverages the human visual system to support knowledge discovery. It has been widely used in many applications, such as social networks, Internet communications, paper citations, and biochemical pathways. However, as data become very large nowadays, traditional graph visualization techniques may fail to reveal the pattern hidden in data. Massive data pose many challenges for graph visualization, such as visual clutter, layout, navigation, and evaluation criteria.

In this survey, we first give a brief overview on graph representations and their layout algorithms. Then we will focus on various visualization techniques for massive graphs. Specifically, we will describe clutter reduction algorithms and user navigation techniques. After that, we introduce some common visualization tasks and discuss the evaluation schemes in typical applications. At the end of this survey, we point out some future research directions.

Chapter 1

Introduction

1.1 Motivation

Graphs are traditional and powerful tools that visually represent sets of data and the relations among them. They have a very long history of helping people communicate, understand the real world, and solve scientific problems. The concept of graphs can be traced back to Ancient Egypt (Rameses I, 1400-1366 B.C)[132], when they were used for a game called "Morris" (See Figure 1.1).

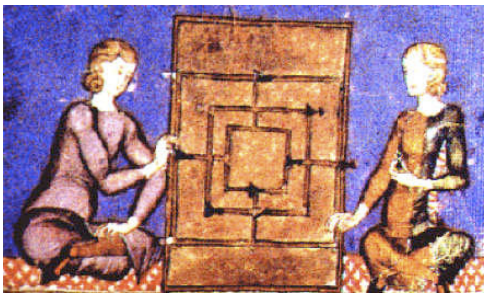
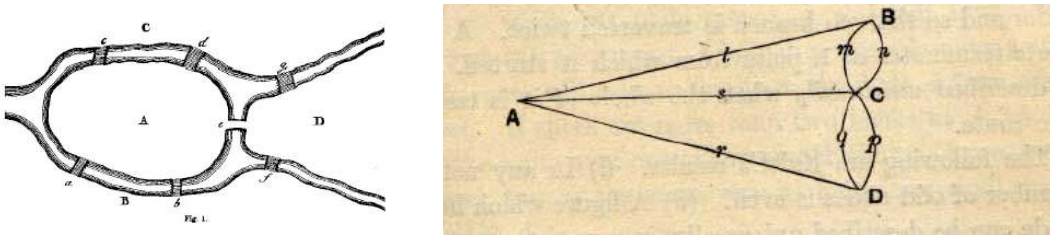


Figure 1.1: Morris Games: Probably the earliest drawing of graph appeared in a book ("Book of Games",13th-centery)

Although graphs have a long history, they were not used for scientific purposes until Euler published his famous Königsberg paper in 1736. In that paper, he solved the path-tracing problem by using the concept of a graph comprising nodes and edges. It marks the transition from early graphs to modern graph drawing. However, Euler's graph is more like a drawing than a graph (See Figure 1.2(a)). Actually, the appearance of the very first abstract graph drawing appeared in Ball's book on mathematical recreations[12], in which he re-drew the Königsberg problem by using node-link diagrams (See Figure 1.2(b)), in

1892, which was over 150 years later.



(a) (b)

Figure 1.2: Graph drawings of Königsberg problem: (a) Euler's drawing in his paper [51], 1736. (b) Ball's abstract drawing, 1892.

There are several reasons for graphs being powerful visualization tools.

- Graphs are very simple models that can be applied to various fields. Any data that has internal relationships can be modeled as a graph. The World-Wide Web is a good example, in which nodes can represent web pages and links can represent hyperlinks among them. There are a lot more examples such as interpersonal relationships, animal species trees, computer file systems and so on.

- Graphs are an abstract concept having a specific definition. After hundreds of years of development, the graph theory field has a very solid foundation and a set of powerful domain independent algorithms for processing graphs efficiently.

- Human beings have strong visual processing abilities. As visualization tools, graphs can change the nature of a task by providing external memory aids, providing information which can be directly perceived and used without being interpreted and formulated explicitly[179].

- Comparing graphs with other common visual design principles, such as proximity, similarity, closure, continuity, symmetry, relative size, and common fate, Palmer and Rock [131] argue that connecting different graphical elements by lines is most powerful to express that there is a relationship between them. From this point of view, users also find graphs preferable to other visual representations when they try to visually explore data with internal relationships.

1.2 Challenge

Graph visualization is a sub-field of information visualization. It usually refers to representation of interconnected nodes arranged in space and navigation through a visual representation to help users understand the global or local original data structures. It is a complex

field, since it draws on ideas from several intellectual domains: computer science, psychology, semiotics, graphic design, cartography, and art. It also makes the task of analyzing a set of data with relations full of challenges.

A major challenge in graph visualization is the size of graphs. Large graphs can cause several difficult problems:

- **Algorithm complexity:** Graph size is crucial to algorithms in some cases, because a lot of useful graph algorithms are NP-complete or NP-hard. Therefore, some layout algorithms can be totally unusable when facing graphs of a large size. Even if the algorithm time complexity is bearable, the long processing time can also make it hard to interact in real-time.

- **Display clutter:** Practical data usually has thousands of elements or even more. When the size of data grows, the corresponding graph becomes cluttered and visually confusing, and users have more and more trouble discerning between nodes and edges. In some extreme cases, the data can be so large it could even exceed the number of display pixels.

- **Readability:** Even if the graph size is still manageable from the point of view of screen space, how effectively a graph can convey the information required by users is still challenging, because human perceptual abilities usually require a small graph size. Ghoniem et al. [65] show that even for a simple task such as locating a node or finding the links between two nodes, node-link diagram performs badly, even for graphs with as few as 20 nodes (See Figure 1.3).

- **Navigation:** People get used to perceiving the world though both local detail and global context. Navigating large information spaces, such as graphs with thousands of nodes, suffers the problems of viewing a large space on a small display. How to navigate a huge graph but not get lost is a challenge problem.

Besides the size of a graph, the complex data structure is another challenge. Current data typically contains more than three attributes. For example, for a social network such as Facebook, in which nodes represent people and links represent interpersonal connections. Nodes may be accompanied by information such as age, gender, and identity. Links also may have different types, such as colleague relationships, classmate relationships, and family relationships. To naturally represent all the information at the same time is really challenging. The most common solution is to use visual cues, such as color, shape, or transparency to encode different attributes.

However, visual encoding can lead to another challenge: what is a better way to encode different attributes? Due to the constraints of the human perceptual system, some visual channels probably have more representational power than others. For example, size and length are more effective for quantitative data, but less useful for ordinal or nominal data. Some pairs of colors are more distinguishable than others. How to make the best of these visual channels to efficiently deliver information to users is still not clear.

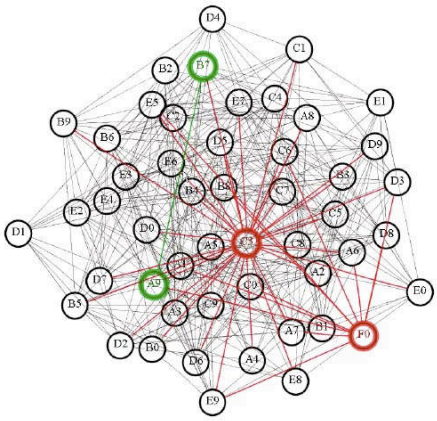


Figure 1.3: A visualization of graph containing 50 vertices and 400 edges[65]

Since the ultimate goal of graph visualization is to help users understand and analyze data. Different layouts can bring different feeling to users, even for the same graph. For example, Figures 1.4 shows two different drawings of the same graph. The only difference is the curvature, but they emphasize different paths. From this point of view, the effectiveness of graph visualization is quite a challenge for researchers.

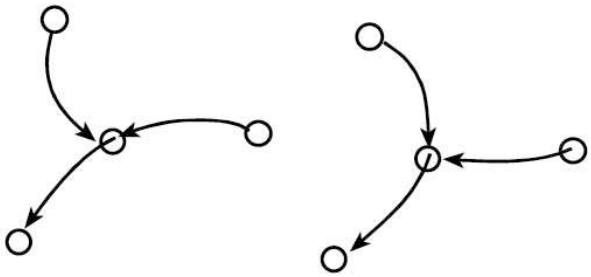


Figure 1.4: Different paths are emphasized by different drawings of the same graph [168]

Various techniques have been proposed for graph visualization for the last two decades. Different aspects of the human perceptual potentials have been explored to help detect patterns in information. However, it is very hard to evaluate their effectiveness and efficiency. Some techniques are so intuitive that people do not question their correctness. However, the intuition is not a good comparison criteria even though it is essential from the point of view of information visualization. Moreover, it is very hard to generalize results from

different techniques, because different techniques usually come along with specific tasks. Therefore, as a fundamental challenge, there needs a technique taxonomy based on which tasks they focus on, and a set of benchmark tasks which are general enough for designers to improve their systems and to evaluators to compare different systems.

1.3 Overview

There are many research issues for graph visualization. The goal of this survey is not to provide a comprehensive review of graph visualization. We mainly focus on large graph visualization and navigation.

A general process of large graph exploration usually starts with choosing a layout to show some aspect of the whole data set so that users can get an impression of a global structure. For example, by viewing the overall structure, users may notice some interesting trends, clusters, or outliers. Thus, they will decide how to proceed with the next investigation. Through dimension reduction, filter, or other navigation methods, they can locate features of interest. To summarize the process, Scheiderman presented the Visual Information Seeking Mantra [153]: "Overview first, zoom and filter, then details-on-demand".

Based on the general process steps of graph visualization, the rest of this survey is organized as follows: In Chapter 2, we try to give an overview of current graph layout solutions, which can help users see the global structure of the whole data set. In this part, these layout methods are looked at from the aesthetic point of view. In other words, we focus on producing an aesthetically pleasing layout instead of theoretically proving graph properties such as planarity. Then, we discuss several approaches to the navigation of large graphs in Chapter 3. In Chapter 4, we categorize the common visualization tasks and popular application fields. Finally, in chapter 5, we conclude and present some future research directions.

Chapter 2

Graph Layout

As we have stated before, node-link graphs are most popular for visualizing inherent relations within data. This representation of graphs includes trees and more general networks. In addition to node-link layout, a few new types of visualization models have been proposed recently, such as space Nested layout [94]. In this chapter, we give an overview of different graphical representations for data associated with networks and their layout algorithms.

2.1 Node-Link Layout

Informally speaking, the basic graph layout problem is very simple. Given a set of nodes with a set of relations(edges), it only needs to calculate the positions of the nodes and draw each edge as curve. However, to make graphical layouts understandable and useful is very hard. Basically there are generally accepted aesthetic rules [138, 139], which include:

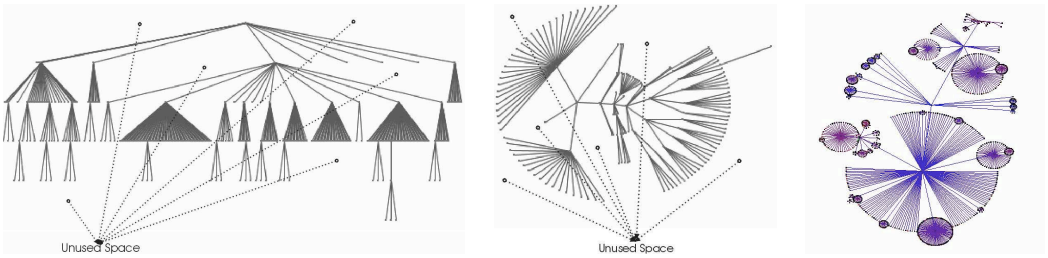
- Distribute nodes and edges evenly.
- Avoid edge crossing.
- Display isomorphic substructures in the same manner.
- Minimize the bends along the edges.

Though these aesthetic rules are generally accepted, they are not equally important. For instance, Purchase et al.[139] show that "reducing the crossings is by far the most important aesthetic, while minimizing the number of bends and maximizing symmetry has a lesser effect". However, most of the time, it is quite impossible to meet all rules at the same time. Some of them conflict with each other. Some of them are very computationally expensive. Thus, practical graphical layouts are usually the results of compromise among the aesthetics.

Another issue about graph layout is predictability. Due to the task of graph visualization, it is important and necessary to make the results of layout algorithm predictable[81], which means two different results of running the same algorithm with the same or similar data inputs should also look the same or alike.

2.1.1 Tree Layout

Tree structures are usually studied separately, because, compared with general network structures, it is more tractable and easier to understand. Another reason is that tree layout problems usually have low complexity (probably sub-quadratic time complexities, such as $O(n)$, $O(n \log n)$, and $O(n\sqrt{n})$).



(a) (b) (c)

Figure 2.1: Node-link layout schemes: (a) Classical hierarchical view for a moderate large tree [126]. (b) Radial view [126]. (c) Balloon view [26].

Node-link layouts use links between nodes to indicate the parent-child relationships. Reingold et al. [143] give a very satisfactory solution for node-link layout as early as 1981. Their classical algorithm is simple, fast, predictable, and produces aesthetically pleasing trees on the plane (See Figure 2.1(a)). However, we can see from Figure 2.1(a), this algorithm has a major problem that it make use of screen space in a very inefficient way. It wastes the root side of the tree and severely clutters the opposite side. Some compact tree layout algorithms are developed for making layouts more dense, while keeping the classical tree looks. For example, Beaudoin et al. [17] overlapped branches of the tree to intensively compress large graphs. However, after the compression, a lot of details are missing. Marriott et al. [114] relaxed the requirement that a parent must be placed exactly midway between its children. Through slightly shifting branches or nodes, they produce a compact and relatively clear compact layout of large trees.

Several variation algorithms have been studied to avoid this problem. for example, Eades proposes another node-link layout called radial layout. Eades's algorithm recursively positions children of a sub-tree into a circular wedge shape according to their depths in the tree (See Figure 2.1(b)). Generally, radial views, including its variations [172, 177, 92], share a common characteristic: the focus node is always placed at the center of the layout, and the other nodes radiate outward on separated circles (the depth of a node from the focus node usually decides which circle that node belongs to). Balloon layout [26] (See Figure 2.1(c)) is similar to radial layout. Balloon layouts are formed where siblings of sub-trees are placed in circles around their father node. This can be obtained by projecting cone tree [148] onto the plane.

Comparing these three node-link tree layouts (classical layout, radial layout, and balloon

layout), the classical layout is the most understandable and well accepted, as it can clearly reflect the intrinsic structure of the data. Though the other two layouts can make better use of the display space, it is much less clear where the root of the tree is, and users may feel confused during the exploration process.

2.1.2 Tree+Link Layout

Since large graphs are much more difficult to handle than trees, tree visualization is often used to help users understand graph structures. A straightforward way to visualize graphs is to directly layout spanning trees for them.

Munzner [120] finds a particular set of graphs called quasi-hierarchical graphs, which are very suitable to be visualized as minimum spanning trees. However, for most graphs, all links are important. It could be very hard to choose a representative spanning tree. Arbitrary spanning trees can also possibly deliver misleading information.

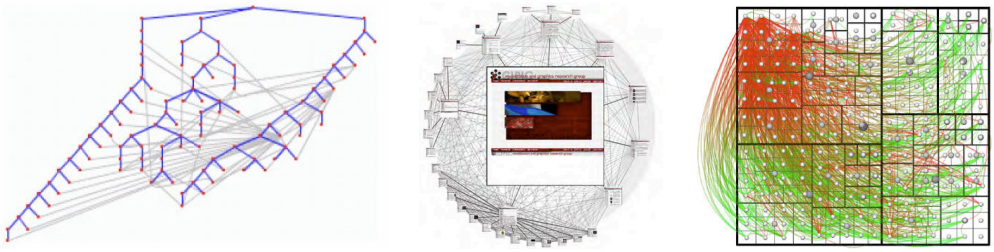
Lee et al. [108] propose another strategy to explore general graphs as trees. Their method, which is called TreePlus, enables users to interactively explore a graph by starting at a node and then incrementally expanding and exploring the graph. Unlike Munzner's algorithm, TreePlus does not have any favorite graph types. However, TreePlus can easily overlook the overall structures, and some local features, such as loops.

Besides visualizing graphs as trees, a more common way to visualize graphs is to use tree+link layouts. After finding spanning trees, we show them by using tree visualization techniques, such as classical tree view [82], radial view [92], or treemaps view [52, 84], and add the rest edges back (See Figure 2.2). Many visualization systems, such as [36, 72, 82, 21, 92, 22, 84], take this approach.

A number of data sets in real world have more than two kinds of relationships. If one of them is hierarchical relationship, tree+link layouts are particularly appropriate. For example, WebMap [36] gives a visualization of user's web browser history. If a webpage is visited for the first time, a new node then is added and connected with its predecessor as part of the underlying tree. However, if the webpage is visited before, a cross link is then created to connect the node with its predecessor. Danny Holten [84] developed this layout further by using the underlying tree as control aid to curve the additional links.

2.1.3 Spring Layout

Spring layout, also known as Force-Directed layout, is another popular strategy for general graph layouts. In spring layout, graphs are modeled as physical systems of rings or springs. The attractive idea about spring layout is that the physical analogy can be very naturally extended to include additional aesthetic information by adjusting the forces between nodes. As one of the first few practical algorithms for drawing general graphs, spring layout is proposed by Eades[45] in 1984. Since then, his method is revisited and improved [57, 61, 43, 59, 40] in different ways. Mathematically, Spring layout is based on a cost (energy) function, which maps different layouts of the same graph to different non-negative numbers. Through



(a) (b) (c)

Figure 2.2: Tree+link layouts: (a) Classical tree with added links [82]. (b) Radial view with added links [92]. (c) Treemaps view with added links [84].

approaching the minimum energy, the layout results reaches better and better aesthetically pleasing results. The main differences between different spring approaches are in the choice of energy functions and the methods for their minimization. For example, Newton-Raphson method used by Kamada and Kawai [95], simulated annealing method used by Davidson and Harel [33], and GEM method used by Frick et al. [56].

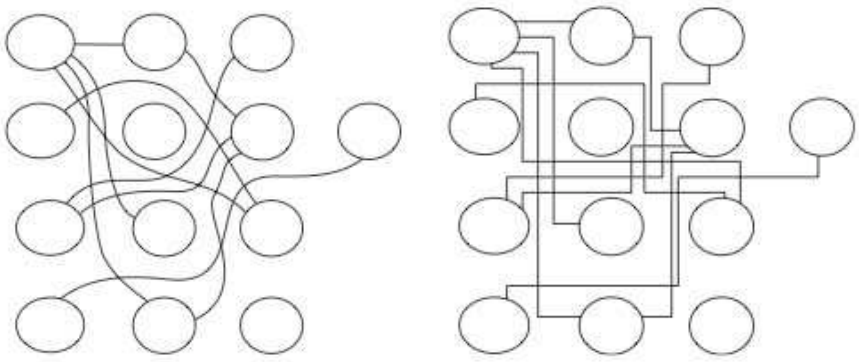
Besides to encode aesthetic information, researchers also try to use spring layouts to avoid node overlapping problems. For example, Eades et al. [44] present force scan algorithm to adjust cluttered layouts. Their algorithm scans horizontally and vertically to find overlapping nodes, move the overlapping nodes to new positions repeatedly until there is no further overlapping. Li et al. [110] also propose two similar spring embedder models to solve the overlapping problems. (It has been proved NP-hard to transform a given overlapping graph into a minimum-area layout without node overlapping which still preserves the orthogonal orders [74].)

In general, force-directed algorithms can successfully produce good results for relatively small graphs, but they do not scale well with size. Large graphs often make the energy function very hard to reach minimum. Furthermore, typical force-directed algorithms run in iterations. During each iteration, all node positions will be renewed based on previous iteration, which probably needs $O(n^2 + m)$ time, where n is the number of vertices and m is the number of of edges. A good layout result usually need $O(n)$ iterations. Therefore, it leads to overall running time of $O(n^3)$ or even $O(n^4)$. Moreover, force-directed algorithms show a lack of predictability, which means two different runs of the same algorithm with a same input graph may be unlike one another. The unpredictability can cause serious problems in some cases, since user navigation heavily depends on the visual representation of graphs.

2.1.4 Node-Link Layout Applications

The layouts mentioned above all assume that the nodes can be placed anywhere without additional constraints. However, There some applications in which the node positions have semantic meanings, such as geographical information. In these applications, nodes are fixed

according to their geographical information. For example, to view the Internet structure and traffic flows, researchers uses arcs or layered structures to help viewers tell apart different links (See Figure 2.4). In fact, as long as the layouts involve links, users always can choose to draw links in different shapes. Table 2.1 gives some examples about how recent applications place nodes and draw edges. We can see that polylines are not preferable in graph visualization. That is because, according to the continuity principle in Gestalt laws [169], humans are more likely to construct visual entities which are smooth, rather than ones with abruptly changed directions (See Figure 2.3).

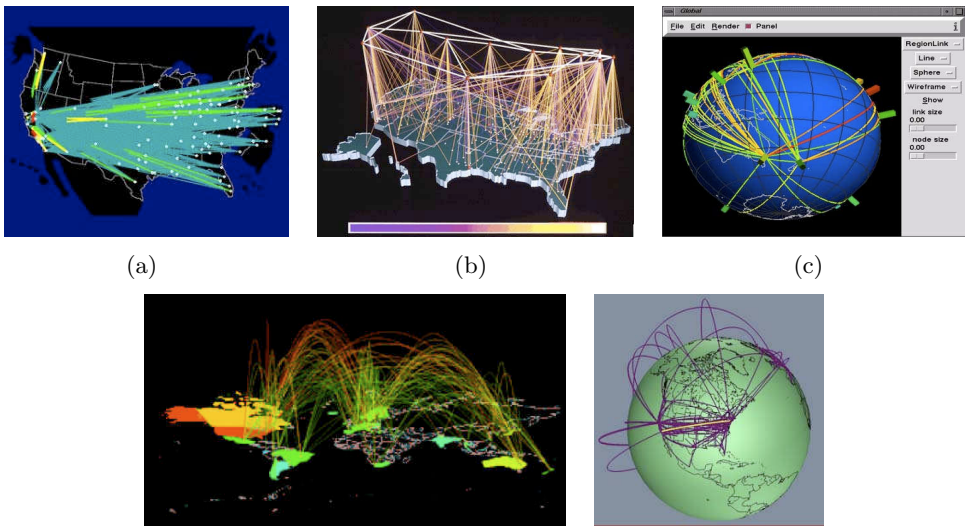


(a) (b)

Figure 2.3: Comparison between smooth links and abruptly changed links. It is easier to perceive the connection relations when nodes are connect smoothly.

Table 2.1: Application categorized by placements of nodes and edges

	Straight line	Polyline	Curve
Force-directed placed	[133, 10, 175, 164, 16, 37, 21, 7, 4, 128, 176, 75, 2]	[40]	[152, 54, 103]
Regular placed	[141, 100, 49, 83, 161, 92, 136, 177, 116, 4, 176]	[48]	[137, 52, 84, 108, 154, 170]
Arbitrarily placed	[8]	[88, 129, 150, 86]	[174, 30, 134, 31, 80]



(a) (b) (c)

(d) (e)

Figure 2.4: Edge drawing in 3D: (a) Node-to-node network overload [18]. (b) Billion-byte inbound traffic on the NSFNET T1 backbone in 1991 [29]. (c) A link visualization system develped at AT&T research lab. [31]. (d) World wide web traffic visualized by SeeNet 3D [31]. (e) Visualization of the Internet's multicast backbone [121].

2.2 Space Division Layout

In space division layouts, the parent-child relationship is indicated by attaching child node(s) to the parent node. Since the parent-child and sibling relationships are both expressed by adjacency, The layout should have a clear orientation cue to differentiate these two relationships. Some researchers [6, 156] think centre-out is a space division layout (See Figure 2.5), because it leaves less boundary area unused. However, the node sizes are difficult to control. Some nodes, which have many children, can be so large that they have a lot of blank spaces in them, while others are too thin to even be labeled or colored.

2.3 Space Nested Layout

Nested layouts, such as Treemaps [94], draw the hierarchical structure in the nested way. They place child nodes within their parent node (See Figure 2.6(a)). In a Treemaps layout, nodes are represented as rectangles. A rectangle can be subdivided horizontally or vertically into smaller rectangles, where each sub-rectangle represents one of its children. The rectangle size is proportional to an attribute of the node. This process is repeated recursively. Treemaps is the most compact display among the three layouts, which leaves no free space in the display. In addition, it is particularly suitable for showing the size of the leaves in a

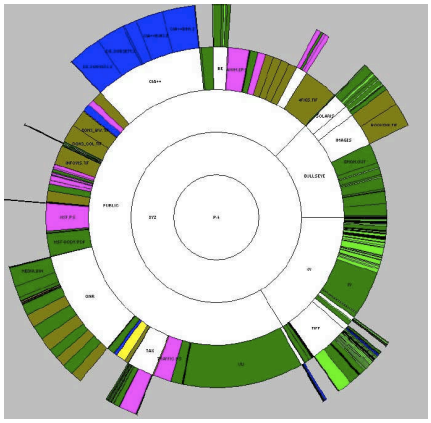


Figure 2.5: SunBurst visualization of a file directory [156]

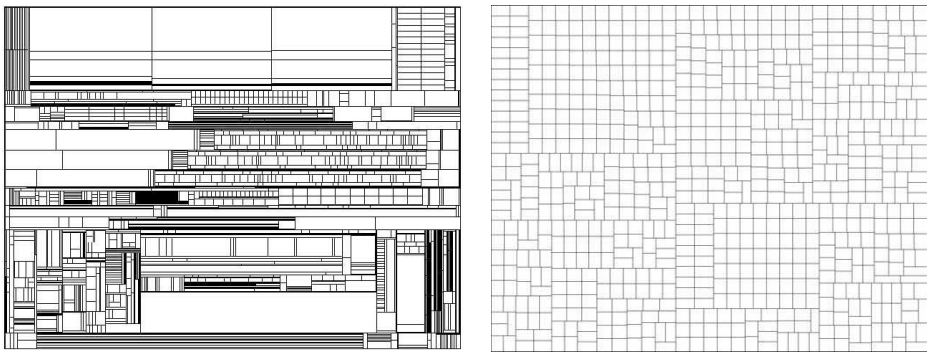
tree. Similar to Treemaps, Circle packing [167], uses nested circles to describe hierarchical structures.

However, the basic Treemaps has several drawbacks. Firstly, due to the subdivision method, long thin rectangles easily appear (Figure 2.6(a) is an example). These long thin rectangles can lead to a number of interaction problems, such as selecting, comparing, labeling and so on. Secondly, the hierarchical structure is harder to discern than in classical node-link tree layout, because the non-leaf nodes are represented implicitly by nesting child nodes in their parent nodes. Furthermore, the spacial relations in Treemap layout may mislead users. Users usually assumes subconsciously that nodes closing to each other in layouts are also closed in the underlying data structure, but it is not always true in Treemaps layout. Figure 2.6(b) shows a Treemaps view of a balanced tree. We can see that the structure information is totally unrecognizable.

The high aspect ratio problem has been studied since Treemaps was designed. Several alternative subdivision algorithms [23, 166, 155] are proposed to generate better aspect ratios. For the hierarchical structure problem, Some possible solutions include: Cushion Treemap [165], which uses shading to enhance the parent-child relations, Beamtrees [163], which keeps the size ratios while leaving gaps between siblings, Voronoi Treemaps [13, 160], which uses arbitrary polygons to give more meaningful visual structures.

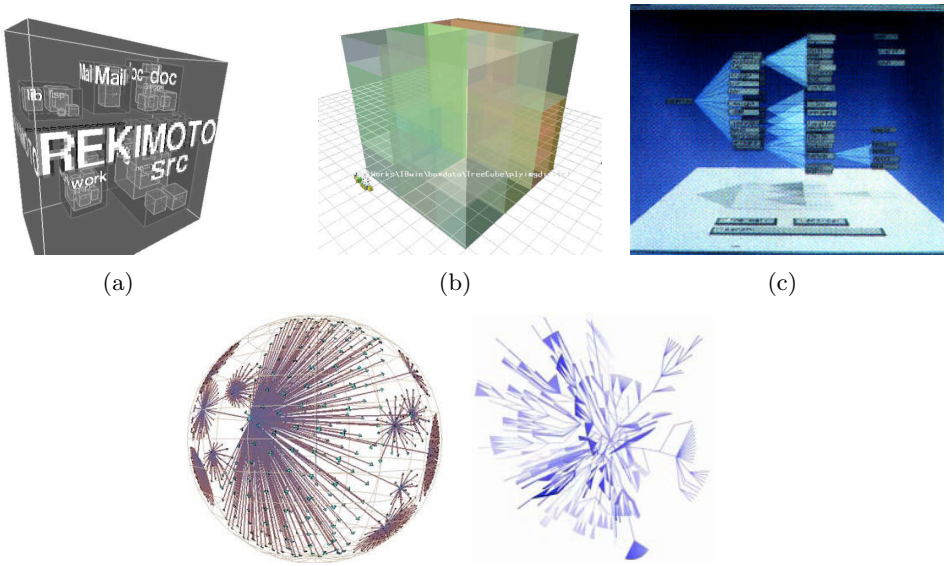
2.4 3D Layout

A lot of 2D layouts have been extended to 3D space (See Figure 2.7(a), 2.7(b)). It is because, informally speaking, the extra dimension can give more space and it would be easier to display large structures.



(a) (b)

Figure 2.6: Treemaps views: (a) Treemaps view of a file directory [165]. (b) Treemaps view of a highly balanced tree [13].



(a) (b) (c)

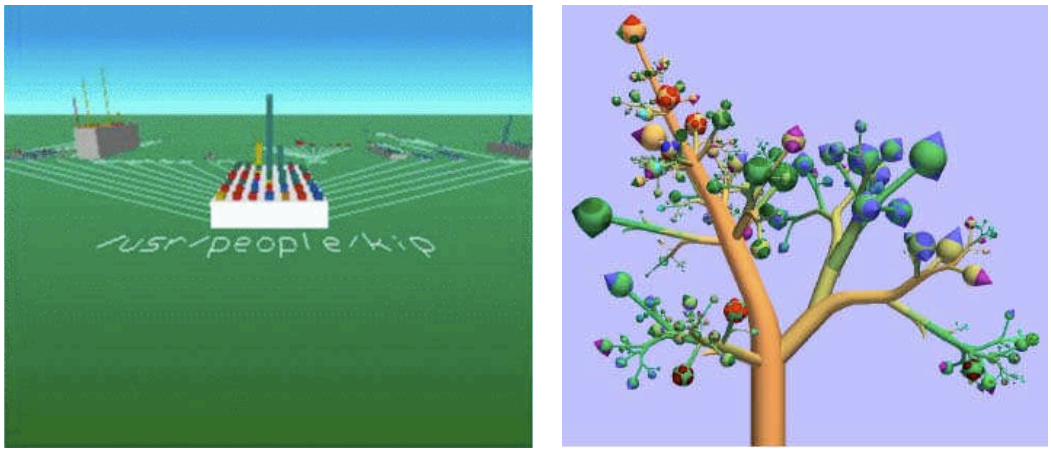
(d) (e)

Figure 2.7: Examples of 3D layouts: (a) 3D space nested layout [144]. (b) 3D treemaps layout [158]. (c) Cone tree layout [148], (d) Hyperbolic view of a tree in 3D space. [106]. (e) Node-link tree (4485 nodes) drawn with the cube polytop (12 subplanes) [85].

For example, Rekimoto implements Information Cube [144], a 3D version of space nested layout. He puts the information cubes inside their parent cubes to represent parent-child relationships. It displays textural labels on semi-transparent cube surfaces. Cone tree [148] (See Figure 2.7(c)) is one of the best-known 3D tree layouts. The parent-child relationships are represented by using a cone, in which the parent is placed at the apex and its children are placed evenly along the cone base. Furthermore, most force-directed methods can be generalized to 3D [178].

Although the extensions are simple and straightforward, displaying graphs in 3D can also encounter new problems. For example, objects in 3D can occlude one another. It is also difficult to find the best view points in 3D space. Therefore, a fundamental need for 3D views is rotation. Through rotating a 3D representation, the hidden structures can be revealed as much as possible.

Gaining more space is not the only advantage of using 3D. Due to the general human familiarity with 3D in the real world, there are some attempts to map hierarchical data to 3D objects we are familiar with. For example, SGI file system navigator uses 3D metaphors to display abstract information (See Figure 2.8), which consists, on the one hand, of adding blocks in the 3D space whose sizes are proportional to the file sizes and, on the other hand, of the ability to "fly" over the virtual landscape created by those blocks. Kleiberg et al. [102] develop a botanical visualization of huge hierarchies, which maps data to a 3D tree (See Figure 2.8(b)).



(a) (b)

Figure 2.8: (a) The SGI file system navigator. (b) Botanical visualization of a unix home directory [102].

2.5 Matrix Layout

An alternative approach to graph visualization is using matrix-based representations (See Figure 2.9). Graphs can be presented by their connectivity matrixes. Each row and each column corresponds to a node. The glyph at the interaction of (i, j) encodes the edge from node i to node j. Edge attributes are encoded as visual characteristics of the glyphs. such as color, shape, and size. The major benefit of adjacency matrices is the scalability. It can easily show a graphs with thousands of nodes. Henry and Fekete [79] describe two matrix layout methods based on approximate traveling salesman problems solutions, which are computed on the similarity of connection patterns but not on the network itself. By reordering rows and columns, they try to reveal outliers, clusters, and patterns underneath the data sets. However, their techniques are poor in time complexity, because they need to compute the full distance matrix between all the vertices. Abello and van Ham [1] use hierarchical aggregation mechanism to visualize and navigate large matrixes. They display the matrix data hiararchically but not ordered. Then users can view and navigate matrixes as trees. However, the graph paths become very hard to detect in this representation.

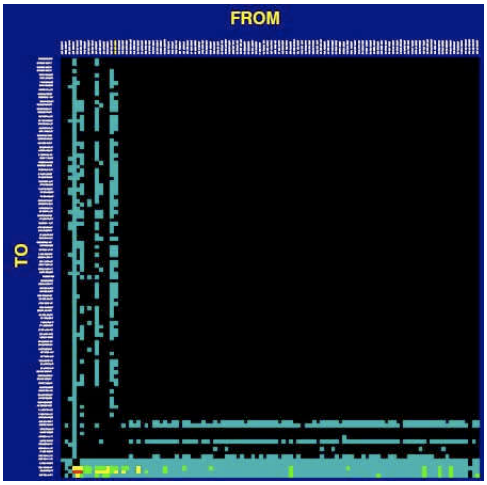


Figure 2.9: An example of matrix views. [18]

Chapter 3

Graph Visualization Techniques

Since none of the static layouts can overcome the problems caused by large graphs, interaction and navigation are essential complements in computerized information visualization. In the last decade, people have developed many visualization techniques to solve the related problems.

3.1 Visual Clutter Reduction

As mentioned above, visual clutter is one of the primary problems caused by large graphs. A good layout should potentially minimize visual clutter. However, trying to find a good layout for reducing visual clutter is not practical, since this kind of problem usually involves optimization, which is usually difficult for large graphs. Researchers have made many different attempts to address or at least minimize this problem. Here we look at three widely used ways: edge displacement, node clustering, and sampling. More details about clutter reduction can be found in [47], which is a taxonomy of clutter reduction schemes given by Ellis and Dix.

3.1.1 Edge Displacement

We can also draw edges in different shapes to reduce visual clutter. Through drawing edges as splines and polylines, we can reduce the edge crossing, which is generally considered to be the most important cause of visual clutter. Edge drawing is especially important for those graphs whose nodes have preassigned positions, such as geographical positions, since we can not let users move nodes around to see how they are connected or to alter the layout.

Dwyer et al. [40] integrate edge routing into force-directed layout algorithms based on constrained stress majorization. They propose several constraints to move nodes and edge bend points for preventing node from overlapping edges and edges crossing each other. However, directly minimizing the number of edge crossings is not a good approach. First of all, to minimize the edge-crossing is NP-Complete [62]. It needs a lot of time to find a optimized solution for large graphs, which happen to be the majority of cases in information

visualization field. However, even in the optimum solution, there are still a lot of crossings. Furthermore, even if we can layer a graph without any edge crossing very quickly, it may involve a lot of zig-zag links in the layout, which is totally aesthetically unacceptable from the point of view of information visualization. Some compromised methods for reducing edge crossing [15, 50, 55, 125] has been proposed recently.

Confluent drawing[49] is another way to alleviate the influence of edge crossing. It does not directly reduce the number of crossings, but draws lines as curves to smooth the intersecting areas, and make them more natural to viewers. The idea behind it is quite simple: for each edge crossing, we make the edges to merge together and drawn as tracks under this constraint: two nodes are connected if and only if there is a smooth curve path connects them without any sharp turns or double backs. Although the constraint is quite clear and determinate. Views still can easily get lost in the complex drawing without any training (See Figure 3.2).

In the confluent drawing layout, we can still distinguish individual links. If we merge links more, we have anther approach called edge clustering. Edge clustering aims to reduce the edge covering area instead of edge crossing. Through merging edges together, more free space will be released and visual clutter will be reduced. Moreover, edge clustering gives a more simple and clear picture of the whole graph. Phan et al. [134] propose an algorithm to generate flow map layouts. However, flow map layout can only be applied to a set of edges which share a common end point, and draws them as a "free-style" binary tree layout (See Figure 3.1). Considering the common end point as the tree root, the algorithm will automatically generate a hierarchical structure based on the leaf positions. There is no strict requirement for the location of the "root". The relative positions of leaves are preserved. Through making the line widths proportional to the edge weights, a flow map can provide a clear flow distribution and reduce the visual clutter. However, it is only good at dealing with a small sub-set of graph visualization problems, such as migration flow from a single source. When dealing with multiple sources in a graph, overlapping flow maps will distract from each other and make the pattern too difficult to read.

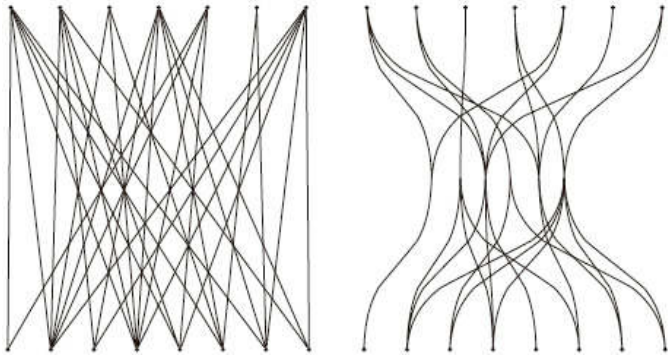
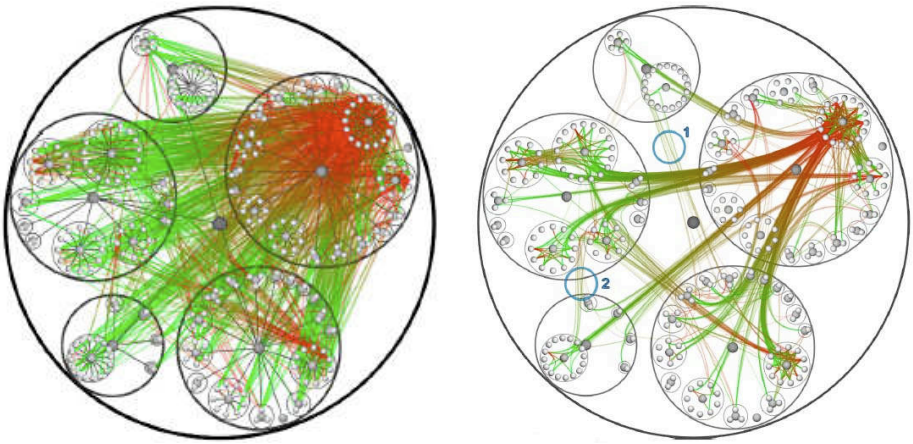


Figure 3.1: An example of flow map: migration flow form California form 1995-2000 [134]

Figure 3.2: An example of confluent drawing of a layered drawings[49]

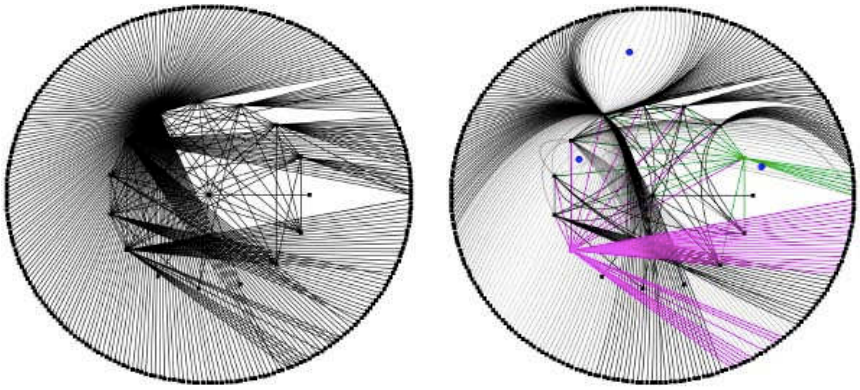
Edge bundles [84] is another edge clustering approach. Compared with flow map layout, it is focusing on clustering links in the tree+link layouts. This kind of problem is showing the links pattern among the elements in a hierarchical structure, such as the reference relations among the elements of a file directory tree. Briefly, in this algorithm, every link is curved by the tree path which connects the link's two end points. If two links share some segment of tree path, they will likely be clustered at that segment (See Figure 3.3). Therefore, the bundle width can intuitively indicate how many links are connecting different parts of the hierarchical structure.

Wong et al. [174] propose a local strategy for removing edge clutter: EdgeLens. Based on users' requests, an EdgeLens can interactively curves graph edges away from their focus of areas without changing the node positions. Therefore, the underlying node and edge relationships can be easily disambiguated (See Figure 3.4). Wong and Carpendale further introduced Edge Plucking [173], an interesting interactive technique which allows users to



(a) (b)

Figure 3.3: Example of hierarchical edge bundles: (a) and (b) show a balloon layout of a sftware system and its associated call relations and its bundled result. [84]



(a) (b)

Figure 3.4: Example of EdgeLens: (a) A simple radial layout with dense edges. (b) EdgeLens views with color and transparency enhancement. [174]

temporarily pluck edges apart to clarify node-edge relationships.

3.1.2 Node Clustering

Node (or data element) clustering is a very general research issue appearing in many domains. It can be known as many names according to the context it is used in, such as cluster analysis, grouping, classification, and pattern recognition.

In the context of visual clutter reduction, clustering means to divide the whole structure into a number of subgraphs (here we call them clusters) and draw those subgraphs as single nodes or small regions. Grouping similar or irrelevant visual elements and releasing more free space can help to reduce the visual clutter. Furthermore, an abstraction of a graph also can help to emphasize the global structure and ease the perception tasks.

Different clustering techniques have been applied applied by researchers to reduce the visual complexity. A major difference between them is the definition of distance or similarity between two items in a data set. There are two basic categories about the similarity measures.

The first one is content-based, which uses semantic data associated with the graph elements to do the clustering. Although it can usually produce very meaningful clustering results [119, 145]. For example, Wattenberg [170] describes a method for aggregating networks according to attributes on their vertices. His method only computes the attribute values, which is much like pivot tables in spreadsheet calculators or data cubes in OLAP databases. This method only works better when the values are categorical with a low cardinality.

The content-based approach is less studied because it is very application relevant. A content-based approach which is specialized to a certain problem usually is not general enough for other problems. Therefore, in this survey we focus on the structure-based approach, which is more general and only uses structural information (relational information such as connectivity information and hierarchy information) to cluster nodes.

In structure-based clustering, well defined clusters usually refer to the graph components who have more intra-link connections than to elements outside them. Various heuristics, such as connectivity, cluster size, geometric proximity and statistical variation [123, 149, 157], have been studied to develop different approaches. Auber et al. [10] present effective algorithms for clustering and visualizing an important class of networks called small-world networks, which have three characteristics: high clustering coefficients, power-law degree distribution and small diameters.

Various techniques can achieve structure-based clustering, such as graph growing, greedy clustering, spectral clustering and multilevel clustering. They can be categorized into three methodologies [28, 87]:

- **Graph theoretical:** Graph theoretical algorithms rely on a similarity matrix representing the similarity between individual nodes. Clusters are formed by closely related nodes, according to a similarity threshold. Each cluster can be represented as a connected graph depending on how these nodes are separated. There is a variety of

techniques in this approach, including spectral bisection [66], spectral quadrisection and octasection [97], and multilevel spectral bisection [14].

- **Single-pass:** Single-pass algorithms can find clusters by growing them from individual data points, called cluster seeds. For example, graph growing and greedy cluster [101, 27] work simply by choosing a starting node, and then add other nodes one by one until the cluster is big enough. More specifically, at each added step, greedy algorithms look for the best node in some measurement while graph growing algorithms look for a node whose resulting sub-graph grows in a certain way.
- **Iterative algorithms:** Iterative algorithms [77, 98, 34, 113] can use clusters generated by other clustering algorithms, such as single-pass clustering, as a starting point. Hierarchical clustering algorithms merge smaller clusters into large ones. Through applying the clustering methods to the clusters generated in a previous step, we can produce layers in a hierarchical drawing of the graph, whose depth is determined by the depth of the recursive clustering hierarchy. After finding a sequence of cluster structures with more and more details, it transfers partitioning information between levels to improve the partition quality on each level. There are three major steps for implementing this method including coarsening graph, partitioning graph, and projecting and refining graph.

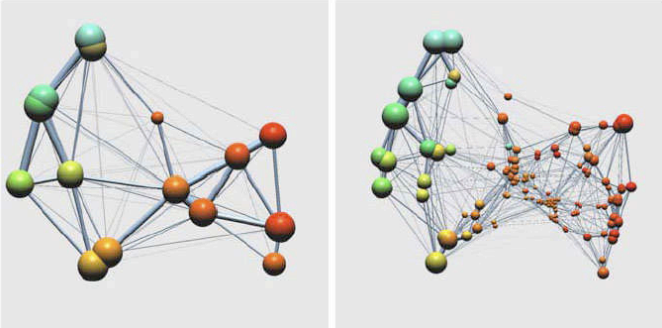


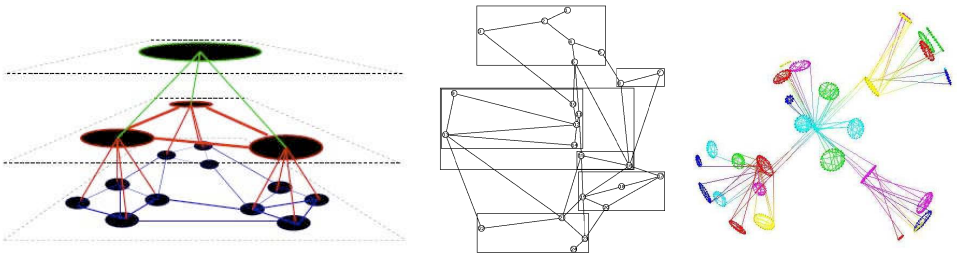
Figure 3.5: Different clustering levels of the same graph [164]

Time complexity is another challenge to node clustering algorithms. In fact, finding the optimized clusters of a graph is still believed to be an NP-complete problem [87]. The classical heuristic clustering algorithm proposed by Kernighan and Lin [101] in 1970 requires $O(m^2n)$ for some graphs, where m is the number of links and n is the number of nodes. Although their algorithm can produce good results, the scalability is rather poor. Recently, the fast clustering algorithm has receives a lot of attention since large data sets are becoming more and more common. Newman [124] proposes a new fast algorithm for detecting community structure in networks, which can runs in time $O((m+n)n)$.

Although the quality of an embedded graph drawing algorithm is highly dependent on its application domain, aesthetics is still one of the most important quality factors in clustered graph drawing in which the readability of a graph is measured or justified. There are some general aesthetic goals that a good clustering algorithm should achieve [39] for the human perceptual needs:

- **Balanced cluster:** In each level of the hierarchy, the size of the clusters should be about the same size, and the distribution of nodes should be as even as possible.
- **Small cluster depth:** There should be a small number of layers in the recursive decomposition.
- **Convex cluster drawings:** The drawing of each cluster should fit in a simple convex region, which we call the cluster region for that subgraph.
- **Balanced aspect ratio:** Cluster regions should not be too "skinny".
- **Efficiency:** Computing the clustering and its associated drawing should not take too long.
- **Symmetry:** Display symmetry should be maximized.

A good layout of clusters can provide users with a clear view of the clustered graph and thus makes it easy for users to visualize and navigate large graphs. Some algorithms [53, 41, 42, 83] assume that the clusters of a graph are given as input along with the graph itself, and just focus on displaying these clusters in two or three dimensions (See Figure 3.6).



(a) (b) (c)

Figure 3.6: Clustered graph layouts: (a) layered layout [41]. (b) 2D layout [42]. (c) 3D layout [83].

3.1.3 Sampling

Sampling is a new approach to reduce visual clutter. Krishnamurthy et al. [104] show that most graph properties, such as the shape of the degree distribution, can be preserved

by random-node selection with sample sizes down to 30%. Random sampling is the most common approach. Rafiei and Curial illustrate the sampling approach in [140]. In their paper, they study several sampling-based schemes, and showed that in most cases the topological properties of a network can be successfully preserved after sampling. They also introduce a notation "focus" to involve human decision and improve sampling results. A "focus" is an area which can be assigned by users. Once it is set, the sampling is biased toward that focal area, thus the visualization emphasizes the focal area and its neighborhood in the graph.

Sampling obviously is a good way to reduce the visual clutter. However, sampling is an unpredictable approach, which means different sampling results can possibly give different impressions to users. Therefore, how to define a good sampling strategy is a challenging problem. Cui et al.[32] propose two measures for computing the data abstract quality: Histogram Difference Measure and Nearest Neighbor Measure. They borrow these two concepts from other disciplines to help analysis be aware of how well the abstracted data represent the original data set. Leskovec and Faloutsos [109] compare existing graph sampling algorithms, and give two different goals of sampling: the back-in-time goal and the scale-down goal. For back-in-time goal, the sampled graph S should be as similar to what the original graph G looks like back in the time, when it had the size of S, as possible. For scale-down goal, the sampling scheme should preserve the properties of original graph as much as possible. In their paper, they perform a systematic evaluation of the published algorithms according to these two goals.

3.2 Interaction and Navigation

Navigation and interaction are essential in computerized information visualization. They can help users reveal the detailed structures in large graphs. Yi et al. [93] give a summarization of popular interaction techniques. Based on the purposes, they are categorized into seven groups:

- **Select:** It helps users highlight certain focus targets, or request computer to process some specific items.
- **Explore:** It is used to change current view point to another part of the data in the same layout representation, such as panning and rotating.
- **Reconfigure:** It is used to switch between different layouts with the same representation scheme, such as replacing nodes in graphs and reordering data items based on a different criteria.
- **Encoding:** It is used to switch different representation schemes, such as changing the layout from node-link representation to treemaps representation.
- **Abstract/Elaborate:** It adjusts the level of abstraction of a data representation to give users different insights into the data, such as zooming and clustering.

- **Filter:** It reduces the amount of data being displayed and makes the remaining items more visible based on users' requested.
- **Connect:** It is used to highlight the connections between items or the items which are relevant to the focus item.

For large graphs, three of them are especially helpful: explore, abstract/elaborate, and filter. In the following parts of this section, we will focus on these three techniques. First, we describe strategies for smooth panning and zooming. Then, we summarize various focus+context methods. After that, we introduce animation, and how it is used to improve the quality of interaction.

3.2.1 Zoom And Pan

Zooming and panning are fundamental tools for exploring large information. Panning means smoothly moving camera across scene. Though zooming, users switch between abstract or detailed insights of data. They are complement to each other in functionality, and quite indispensable when large graph structures are explored.

Due to the simple graphical components of graphs (just nodes and links), zooming is usually don't need a lot of techniques. Unlike zooming into a picture can cause aliasing problems, zooming into graphs only need to adjust the screen transformations, and doesn't any extra cost.

However, the pure geometric zooming sometime can fail to reveal the data pattern, especially when dealing with graphs which are very very dense. Therefore, another form of zooming called semantic zooming is proposed for this situation. Semantic zooming means that the information content changes when users zoom in or out. When users zoom into a particular area, more details are shown. When users zoom out, detailed information is hided, and only abstract information is shown. For semantic zooming, the technical difficulty is not with the zooming operation itself, but rather with assigning an appropriate level of detail to sub-graphs. In the case of graph visualization, the details usually not only refer to the graph structure details, but more important to the underlying data details which users are interested in.

Although zoom and pan are traditional and successful navigation tools, and commonly seen in real world, they can cause some problems in interactive environments. For example, when a computer user is exploring google earth on the screen, and he has zooms into the area around Beijing. The user then wants to change to the view of the area around Hong Kong. The common procedure is first zooming out, panning to Hong Kong and then zooming in again. These zooming out/in steps are not relevant to what he wants but necessary, since without them, it apparently takes much longer time to find Hong Kong. Furthermore, users have to mentally switch between different zoom factors, identify the same item with different resolutions. Furans and Bederson [58] propose space-scale diagrams to alleviate this problem. In their idea, an abstract space is constructed by stacking many copies of the original 2D representations with different different magnifications. Thus, various zoom

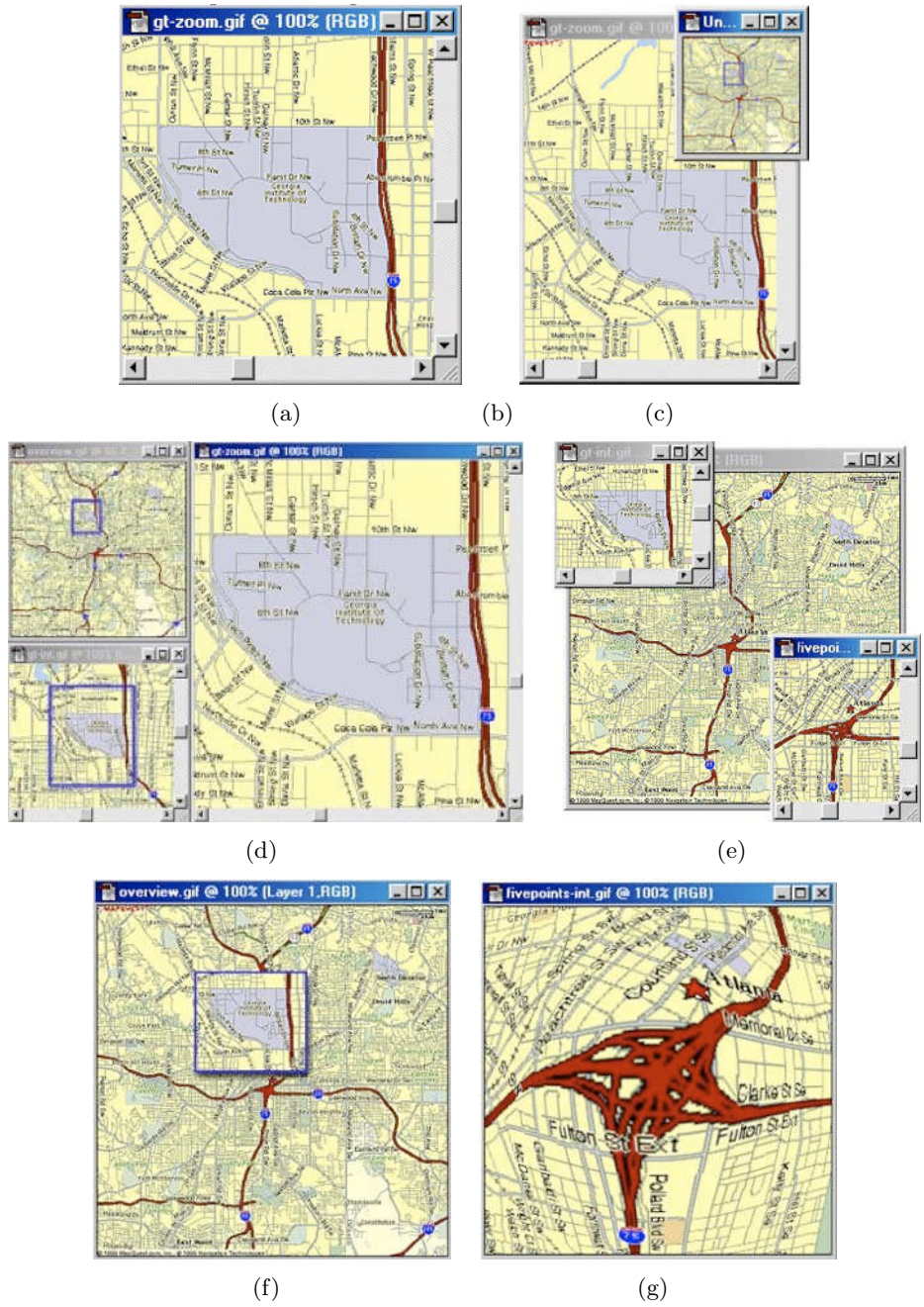


Figure 3.7: Focus+context views: (a) Details only. (b) Single window with zoom and replace. (c) Single coordinated pair. (d) Tiled multilevel browser. (e) Free zoom and multiple overlap. (f) Bifocal magnified. (g) Fisheye view.

and pan actions can be described as a path in this stacked space. For example, a panning interaction refers to a path in the same layer, while a zooming action refers to a path from one layer to another.

3.2.2 Filtering

Filtering refers to hide or de-emphasis items from the view. A straightforward way to filter is based on data's additional attributes, such as removing any data whose attribute values below a threshold. Although the concept of filtering is simple, a useful visual filtering interface should provide various visual browsing tools, such as fast and continuous display of results, progressive refinement of parameters, Ahlberg et al. [5] propose the dynamic query filters for visual information seeking. Their query parameters are rapidly adjusted with sliders, buttons and so on. A key to these principles is to understand the enormous capacity for human visual information processing. The authors utilize the powerful perceptual ability of human beings to help rapidly filter the viewing items. However, Ahlberg et al.'s filtering method has intuitiveness issue, especially when the data is getting more and more complex.

LensBar, an interface tool proposed by Masui [115], is another attempt to simplify the filtering process. Browsing and querying are integrated in LensBar into a simple scroll window slider. The author controls the amount of data to be displayed by ke-word filtering.

Magic lenses [19] also give an idea to further simplify the filtering process. Users can move a magic lens over the display, and wherever it moves to, different information will be shown in the area which it covers. Ellis and Dix [46] use this idea to reduce visual clutter. The lens, which they call sampling lens, is used to show a sampled result of the area it covers (See Figure 3.8). Their sampling lens can automatically reveal the global general structure just by moving the lens around the display.

Figure 3.8: Magic lens: Lines within the lens at 10% sampling rate [46]

Instead of clicking buttons or scrolling slide bars, Adar [3] suggests an interpreted language to facilitate exploratory tasks. Through combining the interpreted language with a graphical front end, user can filter and explore graph structures by typing commands.

3.2.3 Focus+Context

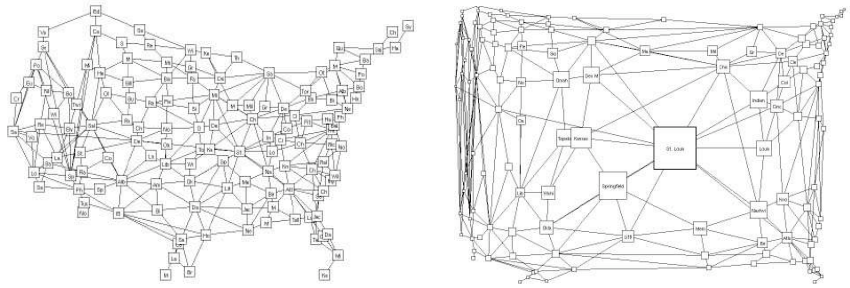
One famous problem with zooming is that if one zooms into some small area, all contextual information is lost. A set of techniques, which can be summarized as focus+context technique, are developed to alleviate this problem. One common characteristic of them is that they all allow users to be able to zoom in one or more certain parts without losing track of where they are in the whole data set.

Different techniques provide focus+context views. Some are quite old. For example, some use separate display regions for different resolutions (See Figure 3.7(c), 3.7(d), 3.7(e)). In these approach, users have to switch between different displays frequently to find out what they are looking at and what the context is. Some techniques improve the usability by avoid frequent view switch. For example, bifocal lens can be adopted to see local details on a large graph. The focus point is directly indicated by the bifocal lens location (See Figure 3.7(f)). Lieberman [111] puts the overview in the same space as the focus. Through making the overview layer and focus layer stacked together translucently, users can see them in the same display at the same time. However, users need to try very hard to differentiate them, because the display becomes very cluttered and the only difference between the two layers is the resolution. In Sunburst, a radial display of tree data [156], the levels of the tree are drawn on concentric circles, with the root of the tree in the center. The farther away a node is from the root (and thus usually the more nodes are on that level), the larger the circle it is located on. Additional space for details and context is made by shrinking the depiction of the whole tree, and only showing the wedge of the circular structure that is currently of interest. The location of the interested part on the whole circle is indicated by the overview image.

In particular, besides the non-distortion techniques mentioned above, there is another group of distortion-based techniques, which provide focus+context views by using geographical distortions. Distortion-based approach imitates the well-known fisheye lens, which enlarge an area of interest, and show other portions of the image with successively less details. Therefore, distortion-based focus+context technique is also called fisheye view.

The fisheye technique is independent of the layout algorithm and is defined as a separate processing step on the graphical layout of the graph. Interacting with fisheye means changing the position of the focus point and/or modifying the distortion value. This independence has positive and negative aspects. On the positive side, it allows for a modular organization of software in which fisheye is a separate step in the graph rendering pipeline somewhere between the layout module and the actual display. Fisheye can also be significantly faster than the layout algorithm, which is an important issue for interaction. However, the fisheye distortion may destroy the aesthetics governing the layout algorithm. For example it can add new and unwanted edge crossings.

Sarkar and Brown [151] presented one of the highly cited works in fisheye view related visualization. They apply different fisheye schemes on a graph of US cities. Through comparing distorted maps with normal US city map, they successfully demonstrate the advantage of fisheye view over normal representation and how they are understandable (See



(a) (b)

Figure 3.9: A graph of 134 US cities and 338 inter-city paths. The a priori importance value assigned to each vertex is proportional to the logarithm of population of the corresponding city. (a) Regular view. (b) Fisheye view.

Figure 3.9).

Fisheye views have been extended and generalized [99]. The extension not only provides means to do the distortion on a different level than just geometrically (for example, by replacing images with icons with they are in the context), and to compose the image of different versions of the undistorted images (for example, to provide color discrimination in addition to the distortion). Gansner et al. [60] propose another method involving a topological fisheye for large graphs. Their fisheye shows a focus point with full detailed network around it, and the nodes farther away from the focus point at increasingly coarse resolutions.

Due to the efficiency and effectiveness of fisheye view, it has become more and more common way to visualize many kinds of data, from graphs [151] to file hierarchies [107] to maps [142] to documents [147] to web sites [146] and so on. However, fisheye and related distortions have two major drawbacks. They distort the whole image, even the region of interest, and the distortion is not well supported by graphics hardware. Since interactive fisheye view usually involve moving the area of focus around the display frequently, these two drawbacks can cause the usability problem. In particular, the distortion of whole image can make the focus-targeting process very difficult and distractive. Every little movement of focus point can cause the whole shape of image changed. Gutwin [68, 69] gives a evaluation about how distortion affects the usability. He present speed-coupled flattening, which can automatically adjust the distortion level based on the user's activity, such as mouse velocity, acceleration, to improve usability without extra interaction. A comparison between zoom and fisheye view can be found in [24].

Though various focus+context techniques are keeping coming out from time to time. It is almost impossible to find a perfect solution for all tasks, because different schemes have their own advantages and disadvantages. Some are good at view-size reduction, some are good at shape preservation, some are good at continuity and so on. For example,

geometric distortion keeps topological continuity, while changing aspect ratios in regions outside the focus, and altering shapes of large patterns in general. Therefore, users should keep in mind that there are distortions in shape and position during their exploration, and be able mentally undo them if needed. For a network engineer ,who mainly cares more about the topological continuity than geometric accuracy, it is a good choice. However, for some art designers who want to identify the exact shape of the models, it would not be a good interface. Multiple simultaneous views at different scales (for example, all the non-distortion approach mentioned above), present the information simultaneously, without geometric distortion, but with topological discontinuity at the edge of the views. Users must find correspondences between features within difference displays.

3.2.4 Animation

Animation is an unique advantage of computerized information visualization technique over other paper-based visualization techniques. It has become a very important feature in helping users understand the data sets, because it implicitly employs time as an extra dimension to facilitate data exploration. Robertson [148] argue that animation can help users understand without thinking, since it transfers parts of the cognition tasks to the human perceptual system. Compared with directly flipping between different views, animated transition can give better clues about what the data relations are and help users relate two states of the system. Some researchers have demonstrated that animation can improve users' subject satisfaction. Thus it becoming more and more popular in a lot of works to help users maintain orientation in data visualization. Kadaba et al. [127] conduct a comparison between different explorations using static representation and animated representation. They apply Michotte's ampliation rule, which suggests it is easier to perceive when a moving object strikes another and set the latter into motion, to visualizing causality relationships. They suggest that the animation can generally facilitate comprehension of complex data.

However, overusing it would cause problems. For example, motion can help indicate the points of interest, while it is also a powerful force for distraction. In fact, animation should only be used when it has to be. The results of study conducted by Tversky et al. [162] show that as long as the static representations are intuitive and clear, using static graphs is a good choice, even for the data which contains temporal dimension.

Animation is not a standalone techniques. In fact, all the techniques described above can be combined with animation to improve their abilities. Baecker and Small [11] summarized different ways in which objects can be animated in. For example, Panning and Zooming can be animated by moving a static object within a scene. Heer and Robertson [76] categorize popular animatic transitions into seven groups, such as view transformation, filtering, ordering, and data schema change and so on. Some early systems use animation include Information Visualizer [25], Cone Trees [148],Eades and Huang's force-directed layout, Yee et al.'s radial layout [177] and Vizster [75].

To avoid the disorientations caused by animation, some interactive graph browsers also preserve invariants to help keep the user oriented. For example, in H3 [120], when a node is selected, an animated transition moves it to the center of a sphere. The transition includes

a rotational component so that when the node reaches the center it ancestors are on its left while its descendants appear on the right. The Hyperbolic browser [106] places nodes and links within a hyperbolic space; changing the focus node in effect changes which portion of the space is currently centered.

Gonzales [67] conducted one of the first user studies looking at how animation helps users make decisions. Her empirical study showed that the effect of animation is closely related to its properties. This includes image realism, transition smoothness, and interaction style. The task domain and the user's experience also affect performance. Smooth animation was shown to have a greater positive effect on task accuracy than more abrupt animations. The use of realistic images was also shown to have a greater positive effect on task performance than more abstract imagery.

Although animation is aesthetically good from a lot of points of view, time probably is the weakness of this technique. Animation consumes time, so there is clearly a trade-off in how long the animation should take. Fast animation may confuse users and makes it hard to notice the connections. On the other hand, if the transition takes too long, the users' time will be wasted. Donskoy and Kaptelinin [38] compared three different navigational techniques (scrollbars, zooming, and fish eye), with and without animation. Animation was accomplished by inserting a single additional frame between the initial and final display states. The results did not show any significant improvement in favor of animated transitions. The authors concede that only one intermediary frame might have been inadequate. To achieve smoothness of movement, 10 frames per second are generally considered the minimum required frame rate [25].

Chapter 4

Tasks and Applications

4.1 Graph Visualization Tasks

A useful graph layout means that users can get what they want easily. A number of common tasks are categorized in [154] and [175], including:

- For the whole graph, count the number of nodes. It is one of the simplest tasks for graph visualization, which can give users a visual impression about the data scale they are dealing with.
- For a given node, count the number of its incoming or outgoing links. Since flows can represent many real world activities, such as migration flows, and movements of funds among different accounts, this task is also commonly seen in graph visualization.
- For a given node, find its adjacent nodes. For example, in social networks, it is very important to find the node who have the most interpersonal connections.
- For a given node, find the nodes that can be reached by a certain number of steps, or a good path, such as shortest path, lowest cost path, or even just a visually pleasing path, between it and another given node. In Internet networks, this task is usually studied heavily because of optimization requirements. In social networks, it usually involves the "small world" phenomenon, which indicates that everyone is at most six steps away from each person on Earth.
- For the whole graph, find the middleman nodes. Middleman nodes isolate nodes which connect two or more subgraphs together. For example, in work flow networks, an analyst usually wants to identify and suppress the weakest link which can disrupt collaboration between different subgroups or to stop this problem from happening.
- For the whole graph, find strongly connected clusters. A lot of real world relation networks potentially have cluster patterns. For example, social networks and and paper citation networks are both typically globally sparse and locally dense networks. Identifying the clusters can simplify the exploration into different subtasks.

- For the whole graph, find all nodes/links which share some specific attribute or a given label. It is very common when users need to explore multi-dimensional data, such as to see the relation pattern between two group of elements.

Common tasks are certainly not limited to those mentioned. Users can always define their own tasks. On the other hand, different tasks require different aspects of graphs. Graphs can include different visual elements, from basic networks, to labels, to attributes [154]. Graphs can also be represented in different forms, such as node-link diagrams and treemaps. For different tasks, different layouts perform dramatically differently. However, generally speaking, good layouts should be kept as simple as possible, because any additional information does not help the tasks but becomes a distraction. For example, basic networks are good enough for the first three tasks mentioned above. Force-directed layouts are suitable for finding clusters or locating the middleman nodes. Regular layouts help users to perform path-related tasks more than any other layouts. Matrix layouts are particularly good at revealing different proportions of links from a node that go to different categories.

Henry [78] classifies current popular network visualization systems into two categories: menu-based systems (including [130, 159, 20, 3]) and visual exploration systems (including [133, 75, 108, 96, 170]). In the following parts of this chapter, based on the data types, we introduce three applications. Of course there are more applications related to graph visualizations such as software visualization, and biological graph analysis and so on, but these three data are most commonly seen topics in graph visualization fields, because of their interesting data features.

4.2 Social Networks

Social networks are a type of small-world network [171] for their high clustering coefficient, power-law degree distribution and small diameter. In addition, Social networks are also famous for their "six degrees" property. Several studies such as Milgram's small world experiment [117], have empirically proved that, if a person is one "step" away from a person he/she knows, then everyone is at most six "steps" away from each person on Earth. In another words, social networks usually present many local dense clusters, and global sparse structure with a small number of hub nodes connecting different clusters.

Henry et al. [122] suggest three major tasks particularly suitable for these kind of graphs: identifying clusters, identifying middleman actors, and analyzing roles and positions. They combine two traditional layouts: node-link diagrams to show the global sparse structures, and matrix layout for exploration in the local dense clusters.

Vister [75] focuses visualizing online interpersonal relations, such as e-mail, msn, and blogging. It uses node-link graph, and provides customized techniques for visually searching in large graphs and visualizing community structures. Interactive highlighting is used to emphasize the hidden connections in the large structure. Panning, zooming and distortion are also integrated to help users navigate.

Social networks are also typical multivariate graphs, which means exploring how the

attributes of nodes affect the linking patterns is very important. OntoVis [152] and PivotGraph [170] are two recently published applications targeting at exploring their multi-dimension nodes. Both of them use abstraction methods. By grouping nodes who share a similar attribute, they try to answer questions like "how race affects patterns of communication between genders", or "what some actor's favorite movie genres are". In particular, OntoVis applies both semantic and structural abstractions, and uses force-directed layouts to show patterns by observing big clusters. PivotGaph focuses on structural abstraction, and use regular grid layouts to show patterns by examining edges properties.

4.3 Communication Networks

Communication networks, such as telecommunication networks and Internet, are one of the first graph visualization applications. Communication networks are typical geographical networks, which means the positions of graph nodes should be strictly kept. Visualizing the links clearly is usually the major task for this kinds of networks. When the network structure grows more and more complex, Donna Cox et al. [30, 31, 29] vividly presented various traffics on the NSFNET by powerful geographic visualizations (See Figure 2.4(b), 2.4(d)).

Figure 2.4(c) shows the link visualization system developed at the AT&T research lab, which shows the linking relation among different websites [31]. The size color, and thickness of the nodes and the links are used to encode the corresponding data.

Becker et al. proposed one of the oldest systems called SeeNet [18] to visualize network data. Their system visualizes the associated data on the network instead of simply visualizing the structure of the network itself. In order to increase the interactivity and decrease clutter during visualization, the authors introduced several interactive controls, such as focusing, filtering and animation. The data set to be visualized are the telecommunication traffic among the 110 switches in the AT&T network on Oct. 17, 1989, the day of the San Francisco earthquake. Figure 2.4(a) shows the network-wide overload at that time into and out of the Oakland node, in which segment thickness and color are used to encode the traffic data amount and the bisected segments show the direction. SeeNet3D [31] (See Figure 2.4(d)) expands SeeNet to 3D space by using 3D graphics techniques.

A visualization of the global topology of the Internet MBone is presented by Munzer et al. [121]. They mapped the latitude and longitude of MBone routers to 3D geographical information (See Figure 2.4(e)). their geographical visualization of the MBone is presented as an interactive 3D map using VRML.

4.4 Reference Networks

Citation networks consist of published scientific articles linked together through their citations. Generally speaking, citation networks are representations of directed acyclic graphs. Usually, they also have a high cluster coefficient. The citation network analysis is another old topic of graph analysis, which started with Garfield et al.'s paper [63] in 1964. In

their paper, on the example of Asimov' history of DNA [9], They show that the analysis "demonstrated a high degree of conincidence between a historian's account of events and the citational relationship between these events". Garner makes an overview of possible applications of graph theory in citation network analysis was made in his thesis [64].

For citation network analysis, finding the most important part of the citation network is one major tasks. In 1989, Hummon and Doreian [89, 90, 91] propose three fundamental algorithms to calculate the link weights to provide us with visual identification the main critical paths in citation networks. Recently, Batagelj enhances their algorithms, and successfully apply their algorithms to very large citation networks with several thousands nodes.

NetLens [96] proposes a "content-actor" model to visualize the citation networks. Netlens uses traditional histogram to overview the whole graph, and multiple simple coordinated views of ordered lists to provide more details about the focus actor.

Lu et al. [112] apply node similarity to citation networks. They consider the similarity based on connectivity information only. Since the citation papers are hand-picked by the the authors as being related to their research, Lu et al. argue that the reference information can be explored to judge relatedness. They propose several algorithms to quantify the relatedness, and compare their results with text-based algorithms.

Shneiderman and Aris [154] propose a network visualization strategy based on semantic substrates. They experiment their strategy on the citation network about 2780 federal judicial cases, which is also a directed acyclic graph. By providing user-defined semantic substrates with interactive filters, their system produces a clear interface for users to perform their exploration tasks.

4.5 Evaluation

Evaluating complex systems is a challenge, especially in the field of information visualization [135]. When a new heuristic or intuitive technique is introduced to people, some evaluation should always be provided to prove that the intuition is correct and the cool results are also correct and useful.

In Moore's research [118], he describes the process of new technology adoption. He divides the end users into two groups: early adopters and early majority. It is always easy to sell novel technologies to early adopters, because they are visionaries and eager to try new tools. On the other hand, early majority are pragmatists who want something that is reliable, proven, and solves real problems. Unfortunately, early majority is a much larger group than early adopters. Therefore, providing convincing evaluations of new techniques is one of the major tasks of information visualization researchers.

Evaluations range from understanding users' needs to formal controlled experiments to get statistically persuasive results. Kang et al. [96] and Plaisant [135] all propose some tasks for evaluation, including:

- **Usability:** Evaluation should measure how usable the proposed technique is, by using

classical usability measures such as speed of performance or error rates on simple imposed representative tasks.

- **Improvement:** Evaluation should show solid evidence that the proposed technique really has an advantage over previous solutions. Controlled comparison experiments are the most common way to do this evaluation.

- **Ability:** Evaluation should find the range and complexity of questions that can be answered, and characterize the questions that cannot be answered.

- **Generality:** Evaluation should assert how specialized the technique is, how highly it depends on the specific problems, and how easy it is to create new applications for new domain problems.

User studies are the backbone of an evaluation, because they can demonstrate the usability and ability. They appear in many papers such as [96, 35, 170, 76, 108, 176, 167, 84, 75]. For example, Wong et al. [175] evaluate the semantics information in the graph by conducting four experiments on common graph tasks, such as finding isolated nodes and finding adjacent nodes to a particular node. They choose 16 participants with different backgrounds. For each experiment, the participants are divided into two groups, and offered different tools and information. Then the researchers question them, measure their responses, for accuracy and time, and ask for their comments.

Sometimes, user studies could be very informal [167, 84]. In informal user studies, researchers just give the participants their systems, after a few hours or days, ask their comments and suggestions, and summarize their feedback and conclusions. In some rare cases, there are no user studies for some systems which should have evaluations done on their usability [2, 79, 13]. However, in these cases, researchers usually have strong arguments when they introduce their techniques, or convincing results to demonstrate their claims.

A formal user study consists of a number of components, such as participants, platforms, hypotheses, procedures, result discussions, and subject evaluation.

- **participants:** Based on the testing data set, user study usually consists of 6 to 30 persons, which could be either experts of the corresponding domain [170] or amateurs [75, 176, 108, 76, 35]. However, the types of subjects usually decide how the methods of the user studies are conducted. For example, evaluations based on experts usually highly depend on users' subjective comments, because they are all experienced users, and have deep insights into the data. Their judgments usually hit the nail on the head. For the evaluations based on amateur users, the design of controlled experiments are more important.

- **Platforms:** Materials are not the determinant components of the user study. Not every user survey has to contain it. However, when the evaluation involves a lot of display interface activities, it is better to mentions the monitor models and their resolutions, and so are CPU models when time is critical.

- **Hypotheses:** Hypotheses refer to the effects researchers anticipate in their experimental data. No matter researchers record them in their papers or not, hypotheses are always needed and important, because they influence the design of test sets.

- **Procedures:** The procedures are related to the types of participants. Controlled experiments are how user studies proceed. For each hypothesis researchers have got, they design a number of (for example, six to ten) tasks [76, 108, 176]. The tasks should be designed carefully, because the results should be qualitative and objective. There are a few popular tasks for graph visualization, including:

  - Locating a specific element.
  - Tracking a path or an element.
  - Counting a number, such as the number of elements with same attribute value, the number of elements connecting with a common elements, etc.
  - Finding a special element, such as the biggest/smallest one, the cluster containing most items, etc.

  In controlled experiments, tasks should be performed in different ways, so that researchers can compare the results to support their claims.

- **Result discussions:** Testing results should be as objective as possible. Time, accuracy rate, and average error are favorable to comparisons. Besides using charts or tables to demonstrate the advantages and improvements of new techniques, analysis should also be made at this stage, including explaining figures, arguing strengths and weaknesses.

- **Subject evaluations:** Subject evaluation usually acts as a complement in the user studies. For example, subject evaluation could be summarized as user feedback [84, 167], or integrated in the result discussion parts, or conducted as a survey [76, 3].

Besides user studies, usability can also be proven by conducting case studies [152, 80, 96, 3, 154, 105, 109, 87]. In case studies, no subjects are involved, researchers just need to apply their techniques or system on one or a few real-world data sets, and demonstrate the usability by showing their findings. In some cases, case studies are preferable to user studies, such as the complex exploration techniques/systems which are very hard to operationalize the exploratory process, or objectively compare subjects' findings. In addition, case studies can show the usefulness of a visualization system in the real world environment. However, comparing with user studies, case studies are less common. The limitation is that the results of the case study are not duplicable. It is also hard to prove the generality by case studies, since different cases are relatively isolated and unique.

Comparison is a third way to do an evaluation. It is the least common method for evaluating graph visualization. It is especially suitable for the new techniques highly related to algorithms and less involving user interaction [35, 8, 87]. However, in the graph visualization field, most cases do not fall into this category.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Graph visualization is a sub-field of information visualization. It focuses on visually representing abstract data elements and the relationships between and reduce the cognitive load to understand the global and local structure.

In this survey, we review two graph visualization techniques: graph layout and visualization techniques.

Graph layout has been studied for years. Some old layout algorithms were developed early on and have become relatively mature now, such as node-link tree layout. However, when the real world data grows exponentially, common layouts are becoming more and more inadequate or even unusable, because of their poor scalability. Thus, different variations, such as Treemaps layout and forced-directed layout, have been developed with the aim of large graph visualization. Though a lot of efforts have been made to improve the layout scalability, each one of them has their own merits and drawbacks. No layout algorithm alone can provide a satisfactory solution for different tasks. For example, some may be very time consuming, while others may not be visually pleasing from the aesthetic point of view.

Hachul and Jünger [71] compare six graph drawing algorithms for twenty-nine real or artificial graph examples. In conclusion, they find out that HDE [73] and FM^3 [70] have best scalability. However, if we just use graph layout to tackle the scalability, there are basically no differences between computer-based and paper-based graph visualization solutions. As an inherent feature of computer system, interactivity is heavily involved in visualization techniques which can help simplifying the exploration tasks. Shneiderman gives a general guide to visual information-seeking mantra [153]:"Overview first, zoom and filter, then details-on-demand". In the case of graph visualization, "overview first" usually indicates visual complexity reduction. Different aspects of visual complexity have been studied, such as edge crossing and node cluster, and different reduction strategies are developed accordingly. Still, the selection of reduction strategies highly depends on the priority of the specific task. After users get the overview of the global structures, different navigation techniques can help users locate and explore their areas of interest, and furthermore, find

patterns in information.

5.2 Future Work

Many graph visualization techniques have been proposed over the past 20 years. They provide fruitful domain-independent graph algorithms which can be applied to various practical applications. However, pure graph is such a primitive model that it will easily encounter extra constraints when facing real problems. The additional information, which real graph data usually has, can divide visualization tasks into different categories.

Generally, there are some specialized data graph which are common in the real world but not in graph visualization. Of them, we are particularly interested in two: graph data with multivariate attributes and graph data with geographical information.

Multivariate dimension data is very common in practice. Just like a person in a social network, a node in a graph can contain much additional information. Intuitively, the most common strategy is drawing nodes with different colors or shapes to indicate different dimensions[152, 170]. Although it is a natural extensions of familiar displays, it has two drawbacks. Color and size both have limitations in quantitative comparisons. Furthermore, complex encoding schemes usually make a visual representation too chaotic to read. Since in many applications, viewing the correlations among different attributes is crucial.

For applications where the data has geographical information, it is important to place the nodes to reflect their relative geographical relations. This type of problems is also very common in practice, such as road maps [86], migration flows [134] and Internet traffics [31]. However, it is also very difficult in graph visualization, because the preassigned node positions make common visualization techniques totally unusable.

Besides dealing with these problems which are common in the real world but not in graph visualization, there are also a lot of unclear general problems that exist in graph visualization. For example, how do we evaluate the usability? Many papers describe intuitive techniques to solve problems, but there not many evaluation is provided to prove the intuition is correct or the pretty resulting picture is useful and correct. More examples include how to give a quantitative comparison between two similar layout results, how to derive highly sensitive and selective algorithms to find causality or make visual inferences and so on.

Bibliography

[1] J. Abello and F. van Ham. Matrix Zoom: A Visual Interface to Semi-External Graphs. *Proceedings of IEEE Symposium on Information Visualization*, 2004.

[2] J. Abello, F. van Ham, and N. Krishnan. ASK-GraphView: A large scale graph visualization system. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, 12(5), 2006.

[3] E. Adar. GUESS: a language and interface for graph exploration. *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 791–800, 2006.

[4] E. Adar and M. Kim. SoftGUESS: Visualization and Exploration of Code Clones in Context. *Software Engineering, 2007. ICSE 2007. 29th International Conference on*, pages 762–766, 2007.

[5] C. Ahlberg and B. Shneiderman. Visual information seeking: tight coupling of dynamic query filters with starfield displays. *Conference on Human Factors in Computing Systems*, 1994.

[6] K. Andrews and H. Heidegger. Information slices: Visualising and exploring large hierarchies using cascading, semi-circular discs. *Proc of IEEE Infovis' 98 late breaking Hot Topics*, pages 9–11, 1998.

[7] D. Archambault and D. Auber. Smashing Peacocks Further: Drawing Quasi-Trees from Biconnected Components. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):813–820, 2006.

[8] D. Archambault, T. Munzner, and D. Auber. TopoLayout: Multilevel Graph Layout by Topological Features. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):305–317, 2007.

[9] I. ASIMOV. THE GENETIC CODE. *NY State J Med*, 65:1646–51, 1965.

[10] D. Auber, Y. Chiricota, F. Jourdan, and G. Melançon. Multiscale visualization of small world networks. *Proc. IEEE Symposium on Information Visualization*, pages 75–81, 2003.

[11] R. Baecker and I. Small. Animation at the interface. *The Art of Human-Computer Interface Design*, pages 251–267, 1990.

[12] W. W. R. Ball. *Mathematical Recreations and Essays*. The MacMillan Company, 11th edition, first published in 1892 edition, 1939.

[13] M. Balzer, O. Deussen, and C. Lewerentz. Voronoi treemaps for the visualization of software metrics. *Proceedings of the 2005 ACM symposium on Software visualization*, pages 165–172, 2005.

[14] S.T. Barnard. PMRSB: parallel multilevel recursive spectral bisection. *Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM)*, 1995.

[15] W. Barth, M. Junger, and P. Mutzel. Simple and Efficient Bilayer Cross Counting. *Proceedings of Graph Drawing*, 2528:130–141, 2002.

[16] V. Batagelj and A. Mrvar. Pajek–program for analysis and visualization of large networks. *Ljubljana, Slovenia*, 2006.

[17] L. Beaudoin, M.A. Parent, and L.C. Vroomen. Cheops: A compact explorer for complex hierarchies. *Proceedings of the IEEE Conference on Visualization*, 1996.

[18] R.A. Becker, S.G. Eick, and A.R. Wilks. Visualizing network Data. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):16–28, 1995.

[19] E.A. Bier, M.C. Stone, K. Pier, W. Buxton, and T.D. DeRose. Toolglass and magic lenses: the see-through interface. *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 73–80, 1993.

[20] S.P. Borgatti, M.G. Everett, and L.C. Freeman. Ucinet for Windows: Software for Social Network Analysis. *Harvard: Analytic Technologies*, 2002.

[21] F. Boutin, J. Thievre, and M. Hascoeet. Focus-based filtering+ clustering technique for power-law networks with small world phenomenon. *Proceedings of SPIE*, 6060:236–247, 2006.

[22] F. Boutin, J. Thièvre, and M. Hascoët. Multilevel Compound Tree-Construction Visualization and Interaction. *Human-Computer Interaction Interact*, 5, 2005.

[23] M. Bruls, K. Huizing, and J.J. van Wijk. Squarified Treemaps. *Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization*, pages 33–42, 2000.

[24] T. Buering, J. Gerken, and H. Reiterer. User Interaction with Scatterplots on Small Screens-A Comparative Evaluation of Geometric-Semantic Zoom and Fisheye Distortion. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):829–836, 2006.

[25] S.K. Card, G.G. Robertson, and J.D. Mackinlay. The information visualizer, an information workspace. *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, pages 181–186, 1991.

[26] J. Carriere and R. Kazman. Interacting with huge hierarchies: beyond cone trees. *The 1995 Information Visualization Conference*, pages 74–81, 1995.

[27] G. Chartrand and O.R. Oellermann. *Applied and algorithmic graph theory*. McGraw-Hill New York, 1993.

[28] C. Chen. *Information Visualization: Beyond the Horizon*. Springer, 2004.

[29] D. Cox and R. Patterson. Visualization Study of the NSFNET. *Retrieved June*, 2003.

[30] KC Cox and SG Eick. Case study: 3D displays of Internet traffic. *Information Visualization, 1995. Proceedings.*, pages 129–131, 1995.

[31] K.C. Cox, S.G. Eick, and T. He. 3D geographic network displays. *ACM SIGMOD Record*, 25(4):50–54, 1996.

[32] Q. Cui, M. Ward, E. Rundensteiner, and J. Yang. Measuring Data Abstraction Quality in Multiresolution Visualizations. *IEEE TVCG*, 12(5):709–716, 2006.

[33] R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics (TOG)*, 15(4):301–331, 1996.

[34] M. Delest, F. Bordeaux, J.M. Fedou, N.S. Antipolis, F. Jean-Marc, G. Melançon, and F. Montpellier. A Quality Measure for Multi-Level Community Structure. *SYNASC 8th International Conference*, 2006.

[35] E. Di Giacomo, W. Didimo, L. Grilli, and G. Liotta. Graph Visualization Techniques for Web Clustering Engines. *Visualization and Computer Graphics, IEEE Transactions on*, 13(2):294–304, 2007.

[36] P. Doemel. WebMap-A Graphical Hypertext Navigation Tool. *Proceedings of the Second International World Wide Web Conference*, 1994.

[37] U. Dogrusoz, E. Giral, A. Cetintas, A. Civril, and E. Demir. A compound graph layout algorithm for biological pathways. *Pach and Shahrokhi [179]. To appear*, 2004.

[38] M. Donskoy and V. Kaptelinin. Window navigation with and without animation: a comparison of scroll bars, zoom, and fisheye view. *Conference on Human Factors in Computing Systems*, pages 279–280, 1997.

[39] C.A. Duncan, M.T. Goodrich, and S.G. Kobourov. Balanced aspect ratio trees and their use for drawing very large graphs. *Lecture Notes in Computer Science*, 1547:111–124, 1998.

[40] T. Dwyer, K. Marriott, and M. Wybrow. Integrating Edge Routing into Force-Directed Layout. 2007.

[41] P. Eades and Q.W. Feng. Multilevel visualization of clustered graphs. *Graph Drawing, Proc. 4th Int. Symp. GD*, 96:101–112.

[42] P. Eades, Q.W. Feng, and X. Lin. Straight-Line Drawing Algorithms for Hierarchical Graphs and Clustered Graphs. *Proceedings of the Symposium on Graph Drawing*, pages 113–128, 1996.

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

[44] P. Eades, W. Lai, K. Misue, and K. Sugiyama. Layout Adjustment and the Mental Map. *Journal of Visual Languages and Computing*, 6(2):183–210, 1995.

[45] P. A. Eades. A heuristic for graph drawing. In *Congressus Numerantium*, volume 42, pages 149–160, 1984.

[46] G. Ellis and A. Dix. Enabling Automatic Clutter Reduction in Parallel Coordinate Plots. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):717–724, 2006.

[47] Geoffrey Ellis and Alen Dix. A Taxonomy of Clutter Reduction for Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1216, 1223 2007.

[48] D. Eppstein, M.T. Goodrich, and J.Y. Meng. Delta-confluent Drawings. *Arxiv preprint cs.CG/0510024*, 2005.

[49] D. Eppstein, M.T. Goodrich, and J.Y. Meng. Confluent Layered Drawings. *Algorithmica*, 47(4):439–452, 2007.

[50] T. Eschbach, W. Gunther, R. Drechsler, and B. Becker. Crossing reduction by windows optimization. *Graph Drawing (Proc. GD'02), LNCS*, pages 285–294.

[51] L. EULER. Solutio problematis ad geometriam situs pertinentis. *Comment. Academiae Sci. I. Petropolitanae*, 8:128–140, 1736.

[52] J.D. Fekete, D. Wang, N. Dang, A. Aris, and C. Plaisant. Overlaying Graph Links on Treemaps. *Proceedings of IEEE Information Visualization Symposium Posters Compendium*, 2003.

[53] Q.W. Feng, R.F. Cohen, and P. Eades. How to Draw a Planar Clustered Graph. *Proceedings of the First Annual International Conference on Computing and Combinatorics*, pages 21–30, 1995.

[54] B. Finkel and R. Tamassia. Curvilinear Graph Drawing Using the Force-Directed Method. *Proc. Graph Drawing*, 3383:448–453, 2004.

[55] M. Forster. Applying crossing reduction strategies to layered compound graphs. *Proc. Graph Drawing, GD*, 2528:276–284, 2002.

[56] A. Frick, A. Ludwig, and H. Mehldau. A Fast Adaptive Layout Algorithm for Undirected Graphs. *Proceedings of the DIMACS International Workshop on Graph Drawing*, pages 388–403, 1994.

[57] T.M.J. Fruchterman and E.M. Reingold. Graph Drawing by Force-directed Placement. *Software- Practice and Experience*, 21(11):1129–1164, 1991.

[58] G.W. Furnas and B.B. Bederson. Space-scale diagrams: understanding multiscale interfaces. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 234–241, 1995.

[59] P. Gajer, M.T. Goodrich, and S.G. Kobourov. A multi-dimensional approach to force-directed layouts of large graphs. *Computational Geometry: Theory and Applications*, 29(1):3–18, 2004.

[60] E.R. Gansner, Y. Koren, and SC North. Topological fisheye views for visualizing large graphs. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):457–468, 2005.

[61] E.R. Gansner and S.C. North. Improved force-directed layouts. *Proceedings of the Graph Drawing Symposium 1998*, pages 364–373, 1998.

[62] M. R. Garey and D. S. Johnson. Crossing number is np-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4(3):312–316, 1983.

[63] E. Garfield, I.H. Sher, and R.J. Torpie. The use of citation Data in writing the history of science. *Philadelphia: Institute for Scientific Information*, 1964.

[64] R. Garner. Computer-oriented graph theoretic analysis of citation index structures. 2004.

[65] M. Ghoniem, J.D. Fekete, and P. Castagliola. On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Information Visualization*, 4(2):114–135, 2005.

[66] G.H. Golub and C.F. Van Loan. *Matrix computations*. Johns Hopkins University Press Baltimore, MD, USA, 1996.

[67] C. Gonzalez. Does animation in user interfaces improve decision making? *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*, pages 27–34, 1996.

[68] C. Gutwin. Improving focus targeting in interactive fisheye views. *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, pages 267–274, 2002.

[69] C. Gutwin and A. Skopik. Fisheyes are good for large steering tasks. *Proceedings of the conference on Human factors in computing systems*, pages 201–208, 2003.

[70] S. Hachul and M. Junger. Drawing large graphs with a potential-field-based multilevel algorithm. *Graph Drawing*, 3383, 2004.

[71] S. Hachul and M. Jünger. An Experimental Comparison of Fast Algorithms for Drawing General Large Graphs. 2006.

[72] M.C. Hao, M. Hsu, U. Dayal, and Hewlett-Packard Laboratories. Web-based Visualisation of Large Hierarchical Graphs Using Invisible Links in a Hyperbolic Space. 2000.

[73] D. Harel and Y. Koren. Graph Drawing by High-Dimensional Embedding. *Proc. Graph Drawing 2002*, pages 207–219, 2002.

[74] K. Hayashi, M. Inoue, T. Masuzawa, and H. Fujiwara. A layout adjustment problem for disjoint rectangles preserving orthogonal order. *Systems and Computers in Japan*, 33(2):31–42, 2002.

[75] J. Heer and D. Boyd. Vizster: Visualizing Online Social Networks. *InfoVis 2005 IEEE Symposium on Information Visualization*, 2005.

[76] Jeffrey Heer and George G. Robertson. Animated transitions in statistical data graphs. *Information Visualization, 2007. INFOVIS 2007. IEEE Symposium on*, 2007.

[77] B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. *Proc. Supercomputing*, 95, 1995.

[78] N. Henry. Visually Exploring Large Social Networks. *LECTURE NOTES IN COMPUTER SCIENCE*, 4663:604, 2007.

[79] N. Henry and J.D. Fekete. MatrixExplorer: a Dual-Representation System to Explore Social Networks. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):677–684, 2006.

[80] N. Henry, J.D. Fekete, and M. Mcguffin. NodeTrix: Hybrid Representation for Analyzing Social Networks. *eprint arXiv: 0705.0599*, 2007.

[81] I. Herman, M. Delest, and G. Melancon. Tree Visualisation and Navigation Clues for Information Visualisation. *Computer Graphics Forum*, 17(2):153–165, 1998.

[82] I. Herman, G. Melancon, MM deRuiter, and M. Delest. Latour: A Tree Visualisation System. *1999.*, 1999.

[83] J. Ho and S.H. Hong. Drawing Clustered Graphs in Three Dimensions. 2006.

[84] D. Holten. Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[85] S. Hong and T. Murtagh. Visualization of large and complex networks using polyplane. *Proceedings of Graph Drawing*, pages 471–482, 2004.

[86] S.H. Hong, D. Merrick, and H.A.D. do Nascimento. The metro map layout problem. *Proc. of the 12th International Symposium on Graph Drawing*, pages 482–491, 2005.

[87] M.L. Huang and Q.V. Nguyen. A Fast Algorithm for Balanced Graph Clustering. *Information Visualization, 2007. IV'07. 11th International Conference*, pages 46–52, 2007.

[88] P. Hui, M.J. Pelsmajer, M. Schaefer, and D. Stefankovic. Train Tracks and Confluent Drawings. *Graph Drawing: 12th International Symposium, Gd 2004, New York, Ny, Usa*, 47(4):465–479, 2004.

[89] N.P. Hummon and P. Doreian. Connectivity in a citation network: The development of DNA theory. *Social Networks*, 11(1):39–63, 1989.

[90] NP Hummon and P. Doreian. Computational Methods for Social Network Analysis. *Social Networks*, 12:273–288, 1990.

[91] N.P. Hummon, P. Doreian, and L.C. Freeman. Analyzing the Structure of the Centrality-Productivity Literature Created Between 1948 and 1979. *Science Communication*, 11(4):459, 1990.

[92] TJ Jankun-Kelly and K.L. Ma. MoireGraphs: Radial Focus+ Context Visualization and Interaction for Graphs with Visual Nodes. *Proceedings of the IEEE Information Visualization 2003 Conference, Seatle, USA*, 2003.

[93] John T. Stasko Ji Soo Yi, Youn ah Kang and Julie A. Jacko. Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, 2007.

[94] B. Johnson and B. Shneiderman. Tree-Maps: a space-filling approach to the visualization of hierarchical information structures. *Readings in information visualization: using vision to think table of contents*, pages 152–159, 1991.

[95] T. Kamada and S. Kawai. Automatic Display of Network Structures for Human Understanding. Technical report, University of Tokyo, Faculty of Science, Dept. of Information Science, 1988.

[96] H. Kang, C. Plaisant, B. Lee, and B.B. Bederson. NetLens: iterative exploration of content-actor network data. *Information Visualization*, 6(1):18–31, 2007.

[97] G. Karypis and V. Kumar. MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 2.0. *University of Minnesota, June*, 1995.

[98] G. Karypis and V. Kumar. Multilevel Algorithms for Multi-Constraint Graph Partitioning. *Supercomputing, 1998. SC98. IEEE/ACM Conference on*, pages 28–28, 1998.

[99] TA Keahey. The generalized detail in-context problem. *Information Visualization, 1998. Proceedings. IEEE Symposium on*, pages 44–51, 1998.

[100] D.A. Keim. Visual data Mining. *Eurographics 2002 State of the Art Reports*, 2002.

[101] B.W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307, 1970.

[102] E. Kleiberg, H. van de Wetering, and JJ van Wijk. Botanical visualization of huge hierarchies. *Information Visualization, 2001. INFOVIS 2001. IEEE Symposium on*, pages 87–94, 2001.

[103] S.G. Kobourov and K. Wampler. Non-Euclidean Spring Embedders. *Proceeding of The IEEE Symposium On Information Visualization*, 2004.

[104] V. Krishnamurthy, M. Faloutsos, M. Chrobak, L. Lao, JH Cui, and AG Percus. Reducing Large Internet Topologies for Faster Simulations. *IFIP Networking*, 2005.

[105] G.V. Kumar. Visual Exploration of Complex Time-varying Graphs. 2006.

[106] J. Lamping and R. Rao. The Hyperbolic Browser: A Focus+ Context Technique for Visualizing Large Hierarchies. *Journal of Visual Languages and Computing*, 7(1):33–55, 1996.

[107] J. Lamping, R. Rao, and P. Pirolli. A focus+ context technique based on hyperbolic geometry for visualizing large hierarchies. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, 1995.

[108] B. Lee, C.S. Parr, C. Plaisant, B.B. Bederson, V.D. Veksler, W.D. Gray, and C. Kotfila. TreePlus: Interactive Exploration of Networks with Enhanced Tree Layouts. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1414–1426, 2006.

[109] J. Leskovec and C. Faloutsos. Sampling from large graphs. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and Data mining*, pages 631–636, 2006.

[110] W. Li, P. Eades, and N. Nikolov. Using spring algorithms to remove node overlapping. *Conferences in Research and Practice in Information Technology Series; Vol. 109*, pages 131–140, 2005.

[111] H. Lieberman. Powers of ten thousand: navigating in large information spaces. *Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 15–16, 1994.

[112] W. Lu, J. Janssen, E. Milios, N. Japkowicz, and Y. Zhang. Node similarity in the citation graph. *Knowledge and Information Systems*, 11(1):105–129, 2007.

[113] S. Mancoridis, BS Mitchell, Y. Chen, and ER Gansner. Bunch: a clustering tool for the recovery and maintenance ofsoftware system structures. *Software Maintenance, 1999.(ICSM'99) Proceedings. IEEE International Conference on*, pages 50–59, 1999.

[114] K. Marriott and P. Sbarski. Compact layout of layered trees. *Proceedings of the thirtieth Australasian conference on Computer science-Volume 62*, pages 7–14, 2007.

[115] T. Masui. LensBar-Visualization for Browsing and Filtering Large Lists of Data. *Proceedings of InfoVis*, 98:113–120, 1998.

[116] M.J. McGuffin and R. Balakrishnan. Interactive Visualization of Genealogical Graphs. *Information Visualization, 2005. INFO VIS 05. Proceedings of the 2005 IEEE Symposium on*, pages 3–3, 2005.

[117] S. Milgram. The small world problem. *Psychology Today*, 2(1):60–67, 1967.

[118] G.A. Moore. *Crossing the Chasm*. HarpersCollins Publishers, New York, 1991.

[119] S. Mukherjea, J.D. Foley, and S.E. Hudson. Visualizing Complex Hypermedia Networks through Multiple Hierarchical Views. 1995.

[120] T. Munzner. H3: laying out large directed graphs in 3D hyperbolic space. *Information Visualization, 1997. Proceedings., IEEE Symposium on*, pages 2–10, 1997.

[121] T. Munzner, E. Hoffman, K. Claffy, and B. Fenner. Visualizing the global topology of the MBone. *Proceedings of IEEE Symposium on Information Visualization, San Francisco, California, USA*, 1996.

[122] Jean-Daniel Fekete Nathalie Henry and Michael J. McGuffin. Nodetrix: A hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics*, 2007.

[123] FJ Newbery. Edge concentration: a method for clustering directed graphs. *Proceedings of the 2nd International Workshop on Software configuration management*, pages 76–85, 1989.

[124] MEJ Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):66133, 2004.

[125] M. Newton, O. Sykora, and I. Vrto. Two new heuristics for two-sided bipartite graph drawings. *Proceedings of the 10th International Symposium on Graph Drawing (GD'02)*, 2528.

[126] Q.V. Nguyen and M.L. Huang. A space-optimized tree visualization. *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on*, pages 85–92, 2002.

[127] Pourang P. Irani Nivedita R. Kadaba and Jason Leboe. Visualizing causal semantics using animations. *IEEE Transactions on Visualization and Computer Graphics*, 2007.

[128] A. Noack and C. Lewerentz. A space of layout styles for hierarchical graph models of software systems. *Proceedings of the 2005 ACM symposium on Software visualization*, pages 155–164, 2005.

[129] M. Nollenburg and A. Wolff. A mixed-integer program for drawing high-quality metro maps. *Proc. of the 13th International Symposium on Graph Drawing*, 2005.

[130] W. Nooy, A. Mrvar, and V. Batagelj. Exploratory Social Network Analysis with Pajek. 2005.

[131] S. Palmer and I. Rock. Rethinking perceptual organization: The role of uniform connectedness. *Psychonomic Bulletin & Review*, 1(1):29–55, 1994.

[132] H. Parker. *An Account of the Aborigines and of Part of the Early Civilization.* London: Luzac and Co., 1909.

[133] A. Perer and B. Shneiderman. Balancing Systematic and Flexible Exploration of Social Networks. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):693–700, 2006.

[134] D. Phan, L. Xiao, R. Yeh, P. Hanrahan, and T. Winograd. Flow Map Layout. *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005*, pages 219–24, 2005.

[135] C. Plaisant. The challenge of information visualization evaluation. *Proceedings of the working conference on Advanced visual interfaces*, pages 109–116, 2004.

[136] C. Plaisant, J. Grosjean, and BB Bederson. SpaceTree: supporting exploration in large node link tree, design evolution and empirical evaluation. *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on*, pages 57–64, 2002.

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

[138] H.C. Purchase. Which Aesthetic Has the Greatest Effect on Human Understanding. 1998.

[139] H.C. Purchase, R.F. Cohen, and M. James. Validating Graph Drawing Aesthetics. 1996.

[140] D. Rafiei and S. Curial. Effectively Visualizing Large Networks Through Sampling. *Visualization, IEEE 2005*, pages 48–48, 2005.

[141] M.S. Rahman, S. Nakano, and T. Nishizeki. Rectangular drawings of plane graphs without designated corners. *Computational Geometry: Theory and Applications*, 21(3):121–138, 2002.

[142] U. Rauschenbach, T. Weinkauf, and H. Schumann. Interactive focus and context display of large raster images. *International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media*, 2000.

[143] EM Reingold and JS Tilford. Tidier Drawings of Trees. *IEEE Transactions on Software Engineering*, 7(2):223–228, 1981.

[144] J. Rekimoto and M. Green. The information cube: Using transparency in 3d information visualization, 1993.

[145] JS Risch, DB Rex, ST Dowson, TB Walters, RA May, and BD Moon. The STARLIGHT Information Visualization System. *Proceedings of the IEEE Conference on Information Visualization*, pages 42–49, 1997.

[146] K. Risden, M.P. Czerwinski, T. Munzner, and D.B. Cook. An Initial Examination of Ease of Use for 2D and 3D Information Visualizations of Web Content. *International Journal of Human-Computer Studies*, 53(5):695–714, 2000.

[147] G.G. Robertson and J.D. Mackinlay. The document lens. *Proceedings of the 6th annual ACM symposium on User interface software and technology*, pages 101–108, 1993.

[148] G.G. Robertson, J.D. Mackinlay, and S.K. Card. Cone Trees: animated 3D visualizations of hierarchical information. *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, pages 189–194, 1991.

[149] R. Sablowski and A. Frick. Automatic graph clustering. *Proc. of 4th Symposium on Graph Drawing (GD'96)*, pages 395–400.

[150] E.S. Sandvad, K. Grønbæk, L. Sloth, and J.L. Knudsen. A metro map metaphor for guided tours on the Web: the Webvise guided tour system. *Proceedings of the 10th international conference on World Wide Web*, pages 326–333, 2001.

[151] M. Sarkar, M.H. Brown, P. Bauersfeld, J. Bennett, and G. Lynch. Graphical Fisheye Views of Graphs. *Human Factors*, pages 83–91, 1992.

[152] Z. Shen, K.L. Ma, and T. Eliassi-Rad. Visual Analysis of Large Heterogeneous Social Networks by Semantic and Structural Abstraction. *Visualization and Computer Graphics, IEEE Transactions on*, 12(6):1427–1439, 2006.

[153] B. Shneiderman. The eyes have it: a task by Data type taxonomy for information visualizations. *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343, 1996.

[154] B. Shneiderman and A. Aris. Network Visualization by Semantic Substrates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5), 2006.

[155] B. Shneiderman and M. Wattenberg. Ordered treemap layouts. *Information Visualization, 2001. INFOVIS 2001. IEEE Symposium on*, pages 73–78, 2001.

[156] J. Stasko and E. Zhang. Focus+Context Display and Navigation Techniques for Enhancing Radial, Space-Filling Hierarchy Visualizations. *Proceedings of the IEEE Symposium on Information Vizualization*, page 57, 2000.

[157] K. Sugiyama and K. Misue. Visualization of structural information: automatic drawing ofcompound digraphs. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(4):876–892, 1991.

[158] Y. Tanaka, Y. Okada, and K. Niijima. Treecube: visualization tool for browsing 3D multimedia Data. *Proceedings. Seventh International Conference on Information Visualization, 2003. IV 2003.*, pages 427–432, 2003.

[159] R.C. Team. R: A language and environment for statistical computing. *R Foundation for Statistical Computing*, 2004.

[160] A. Telea and J.J. van Wijk. Visualization of Generalized Voronoi Diagrams. *Proceedings Data Visualization 2001*, 2001.

[161] C. Tominski, J. Abello, and H. Schumann. Axes-based visualizations with radial layouts. *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1242–1247, 2004.

[162] B. Tversky, J.B. Morrison, and M. Betrancourt. Animation: can it facilitate. *International Journal of Human-Computer Studies*, 57(4):247–262, 2002.

[163] F. van Ham and JJ van Wijk. Beamtrees: compact visualization of large hierarchies. *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on*, pages 93–100, 2002.

[164] F. van Ham, JJ van Wijk, and T.U. Eindhoven. Interactive Visualization of Small World Graphs. *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 199–206, 2004.

[165] JJ Van Wijk and H. Van de Wetering. Cushion treemaps: visualization of hierarchical information. *Information Visualization, 1999.(Info Vis' 99) Proceedings. 1999 IEEE Symposium on*, pages 73–78, 1999.

[166] F. Vernier and L. Nigay. Modifiable Treemaps Containing Variable Shaped Units. *IEEE Information Visualization*, 2000.

[167] W. Wang, H. Wang, G. Dai, and H. Wang. Visualization of large hierarchical Data by circle packing. *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 517–520, 2006.

[168] C. Ware. The Visual Representation of Information Structures. *Proceedings of the 8th International Symposium on Graph Drawing*, pages 1–4, 2000.

[169] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, 2004.

[170] M. Wattenberg. Visual exploration of multivariate graphs. *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 811–819, 2006.

[171] DJ Watts and SH Strogatz. Collective dynamics of'small-world'networks. *Nature*, 393(6684):409–10, 1998.

[172] G.J. Wills. NicheWorks: Interactive Visualization of Very Large Graphs. *Journal of Computational and Graphical Statistics*, 8(2):190–212, 1999.

[173] N. Wong and S. Carpendale. Interactive Poster: Using Edge Plucking for Interactive Graph Exploration. *Poster in the IEEE Symposium on Information Visualization*, 2005.

[174] N. Wong, S. Carpendale, and S. Greenberg. Edgelens: an interactive method for managing edge congestion in graphs. *Information Visualization, 2003. INFOVIS 2003. IEEE Symposium on*, pages 51–58, 2003.

[175] P.C. Wong, H. Foote, G. Chin Jr, P. Mackey, and K. Perrine. Graph Signatures for Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1399–1413, 2006.

[176] P.C. Wong, H. Foote, P. Mackey, K. Perrine, and G. Chin Jr. Generating Graphs for Visual Analytics through Interactive Sketching. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1386–98, 2006.

[177] K.P. Yee, D. Fisher, R. Dhamija, and M. Hearst. Animated exploration of dynamic graphs with radial layout. *Information Visualization, 2001. INFOVIS 2001. IEEE Symposium on*, pages 43–50, 2001.

[178] P. Young. Three Dimensional Information Visualisation. *Computer Science Technical Report*, 12:96, 1996.

[179] J. Zhang. The interaction of internal and external representations in a problem solving task. *Proc. Thirteenth Annual Conference of Cognitive Science Society*, 1991.