Visual Analysis of Large Graphs

T. von Landesberger[1,2], A. Kuijper[1,2,3], T. Schreck[1], J. Kohlhammer[1,2], J.J. van Wijk[4], J.-D. Fekete[5], and D.W. Fellner[1,2,3]

[1] TU Darmstadt, Germany
[2] Fraunhofer IGD, Darmstadt, Germany
[3] TU Graz, Austria
[4] TU Eindhoven, the Netherlands
[5] INRIA AZIZ, Paris, France

Abstract

The analysis of large graphs plays a prominent role in various fields of research and is relevant in many important application areas. Effective visual analysis of graphs requires appropriate visual presentations in combination with respective user interaction facilities and algorithmic graph analysis methods. How to design appropriate graph analysis systems depends on many factors, including the type of graph describing the data, the analytical task at hand, and the applicability of graph analysis methods. The most recent surveys of graph visualization and navigation techniques were presented by Herman et al. [HMM00] and Diaz [DPS02]. The first work surveyed the main techniques for visualization of hierarchies and graphs in general that had been introduced until 2000. The second work concentrated on graph layouts introduced until 2002. Recently, new techniques have been developed covering a broader range of graph types, such as time-varying graphs. Also, in accordance with ever growing amounts of graph-structured data becoming available, the inclusion of algorithmic graph analysis and interaction techniques becomes increasingly important.
In this State-of-the-Art Report, we survey available techniques for the visual analysis of large graphs. Our review firstly considers graph visualization techniques according to the type of graphs supported. The visualization techniques form the basis for the presentation of interaction approaches suitable for visual graph exploration. As an important component of visual graph analysis, we discuss various graph algorithmic aspects useful for the different stages of the visual graph analysis process.

Categories and Subject Descriptors (according to ACM CCS): Data Structures [E.1]: Graphs and Networks; Trees—Mathematics of Computing [G.2.2]: Discrete Mathematics—Graph Theory Information Systems [H.4]: Applications— Information Systems [H.5.2]: Interfaces and Presentation—User Interfaces

1. Introduction

The analysis of large graphs is important in many application areas including finance, biology, sociology, transportation, and software engineering. The proper understanding of global and local graph structures is an essential aspect in many analysis tasks in such areas.

Analysis of graphs leads to a variety of different tasks. The analytical tasks often consist of a series of low level tasks [LPS*06]. The main aspect is often the understanding of global and local structure of the graph, the connections between entities, their connectivity, the clusters of highly connected entities, etc. These tasks get very complicated when dealing with large and complex graphs. Obviously, the visualization itself becomes more problematic, but also search and analysis tasks become time demanding, even to a critical extent. One can think of tasks like finding and selecting relevant adjacent nodes, or determining if nodes are accessible, locating clusters of nodes, determining the shortest path between nodes. Furthermore, finding nodes or links satisfying a certain property becomes more and more time demanding.

The analysis of graphs is often supported by visual presentations of the graph. Graph visualization research concentrates on the development of efficient graph layouts and visual mappings supported by interaction and analysis tech-

© The Eurographics Association 200x.

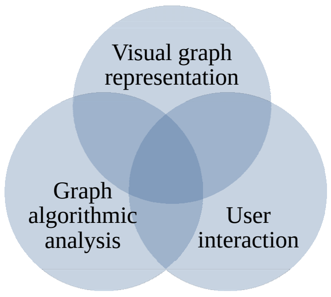


Figure 1: *The main components of visual graph analysis considered in this report.*

niques that enable efficient understanding of the data. The exploration of large graphs is supported by effective interaction techniques, in particular, in cases when the whole graph is too complex or large to be visualized in one static view. The interaction alone may not be sufficient accomplish certain analytical tasks. Therefore, also algorithmic support, such as machine learning, or graph analysis algorithms need to be combined in interactive visualization systems. Such integrated visual analysis of large data sets is the main focus of the research field Visual Analytics, which evolved from Information Visualization and Scientific Visualization [KMS*08]. It has effectively started to grow after the publication of the seminal book by Thomas and Cook in 2005 [TC05]. Therein, Visual Analytics is defined as *the science of analytical reasoning facilitated by interactive visual interfaces*. Recently, Visual Analytics has been a major driving force for the research and development of interactive visualization techniques for large amounts of data including graphs.

Our motivation for this report is two-fold. First, we recognize that by now most recent graph visualization surveys [HMM00, DPS02] date back several years. Therefore, we aim to provide an update by adding more recent publications to the body of work presented in these surveys. Second, we aim to take a Visual Analytics perspective on the field of visual graph analysis by explicitly considering in a unified way the aspects of *visual representation*, *algorithmic analysis*, and *user interaction* (see Figure 1). These three elements form the basis for effective visual graph analysis systems, and are closely interrelated. For example, algorithmic graph analysis may be applied as a preprocessing step before a specific graph layout is determined for visual representation. Interactive direct object manipulation approaches are often useful for exploring large and complex visual graphs. Also, by means of user interaction, further graph analytic processing steps, or updates to the presented views, can be requested. The algorithmic analysis thereby helps to reveal interesting aspects of the data. The user involvement in this analytic process can vary from an automatic analysis, where the calculation is done without user involvement, over a user-driven analysis, where the user triggers the algorithmic processing of the data, up to a user-steered processes where the user has full control of the analytical process including setting the algorithm parameters.

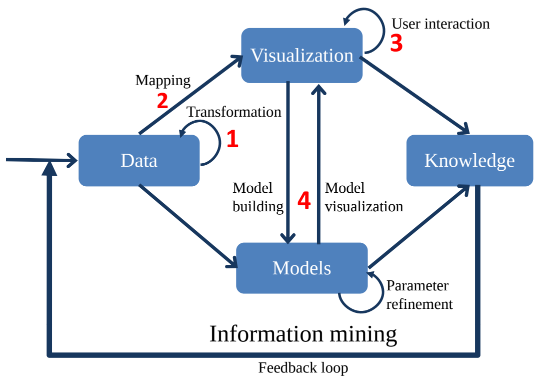


Figure 2: *The Visual Analytics process by Keim et al. [KAF*08] with the four key steps 1: data pre-processing, 2: mapping/layout 3: visual user interaction 4: model-based analysis. ©2008 Springer-Verlag Berlin Heidelberg.*

The structure of the report mirrors the steps of the visual analytics process by Keim et al. [KAF*08] (see Figure 2). Section 2 details definitions and a classification of graphs by types and introduces main pre-processing methods for visual graph analysis. This section is the basis for a discussion of visual graph representations given in Section 3. Sections 4 and 5 survey key approaches for interaction with and algorithmic analysis of graphs, respectively, as these three components are tightly interwoven. Finally, Section 6 concludes and outlines future challenges in this research domain.

2. Basic Graph Definition and Preprocessing Techniques

In this section, we recall fundamental graph definitions as well as approaches for graph preprocessing useful for subsequent graph visualization.

2.1. Graph Types

Graphs are a prominent data structure within Visual Analytics and related research fields. Often, graphs are applied for describing relationships between entities. A **graph** refers to a set of vertices (nodes) and a set of edges that connect pairs of vertices. It is a pair $G = (V,E); E \subseteq [V^2]; V \cap E = 0$, where elements of V are vertices and elements of E are edges [Die05]. A **tree** is a graph without cycles. Cycles are closed paths in the graph, i.e., sequences of nodes following the graph edges, where the first node equals the last node. Trees are called *rooted* when one leaf node (one node with only one incident edge) is distinguished as a so called root node. Such trees are often treated as *hierarchies*, where the length of the path to the root denotes the level of nodes in the hierarchy.

Connected graphs can be transformed to trees by removing edges in the cycles while the graph stays connected (i.e., there is an undirected path between all pairs of nodes) and includes all vertices of the original graph. This process can be reversed by adding back the removed edges. For weighted graphs (graphs with weight-attributes assigned to edges), algorithms for calculating minimum spanning trees (e.g., Kruskal's Algorithm [Kru56]) can be used for this task. Furthermore, attributes can be attached to vertices and nodes, e.g., to denote their type, size, or some other application related information.

Graphs are often classified according to the direction of edges into undirected and directed [HMM00]. In graph theory literature directed graphs with weighted edges are also called *networks*. In information visualization, the term *network* is often used in a broader sense also including graphs with cycles.
The classification in directed and undirected graphs, however, is not sufficient if hierarchical and generic relationships exist within one graph at the same time. E.g., in social networks persons in an organization can be in a subordination (hierarchic) relationship and at the same time in a friendship (generic) relationship. This type of graph in the following is referred to as a *compound graph*. Compound graphs can also be created by successive aggregation (or clustering) of graph vertices in a bottom-up approach. In this case, nodes (and implicitly, also edges) of the original graph are aggregated (i.e., merged), thereby creating constructed *meta-nodes* or *super-nodes*. The attributes of the meta-nodes are calculated from the attributes of the merged nodes. Similarly, edges between meta-nodes are aggregated into *meta-edges* and their attributes are calculated from the original edges. Compound graphs which are constructed in this way are also referred to as *aggregated graphs*. The type of calculation used is dependent on the particular application and graph type.

Graphs may also evolve over time, implying changes in the graph structure and/or in the attributes of vertices and nodes. If such a development is considered, we consider **dynamic graphs** (i.e., time-dependent graphs) in contrast to **static graphs**. Time-dependent changes may affect the node/edge attributes, the graph structure, or both. Figure 3 summarizes the graph classification presented above.

From the Information Visualization point of view, a specific group of graphs are **graphs with geographic reference**, such as transportation graphs. In this case, the nodes and possibly also edges of the graph have an inherent geographic location, which needs to be taken into consideration in their graphic presentation. For example, a specific graph layout algorithm is not needed for determining the position of each node on the screen. However, the fixed node position

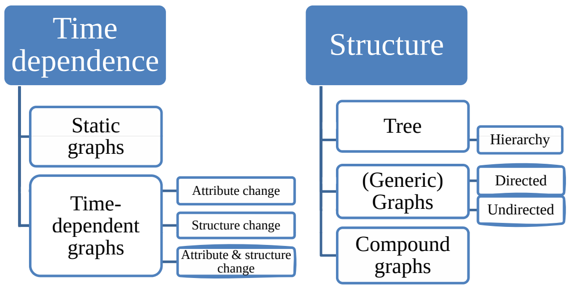


Figure 3: *Classification of graphs according to their time dependence and graph structure.*

exacerbates graph readability problems, such as crossings and long edges. These problems need to be solved appropriately. Visualization of geographic data is a special research field, which we do not address here in detail.

Furthermore, graphs may be distinguished according to their **topological properties**. There exists a variety of literature on graph theory (e.g., [Die05]) which focuses on graph terminology, classification, and algorithmic graph analysis. In the following, we mention only the most relevant terminology used later in this report. Basic graph properties include the number of nodes, graph density, and connectivity. Properties are often taken into account (or are a prerequisite) for certain visualization techniques. *The number of nodes* (i.e., graph order) often heavily influences which methods can be used or fall short, with respect to readability and performance. Another important attribute is the *graph density*, the number of edges relative to the maximum potential number of edges. Sparse graphs have around $O(|V|) < |E| < O(|V^2|)$ edges, while dense graphs show density values close to one. Graphs with the maximum number of edges are called complete graphs. A clique is a subset of a graph that is fully connected. Large and/or dense graphs pose a scalability problem in visualization owing to limited display space and human perception capabilities.

Several special graph structures appear often in real-world cases, and dedicated visualization methods have been developed for these [ACJM03, vHW08, JHGH08, MJW*09]. For example, in the so called *small world graphs* often found in social networks, most nodes are connected to each other with short paths. *Scale-free networks*, e.g., protein networks or certain types of social networks have degree distributions following approximately the power law. *Bipartite graphs* are graphs whose nodes form two disjoint sets V_1 and V_2, $V_1 \cup V_2 = V$, such that every edge $e = v_1, v_2 \in E$ connects vertex $v1 \in V_1$ with one vertex $v_2 \in V_2$.

2.2. Algorithmic Graph Preprocessing

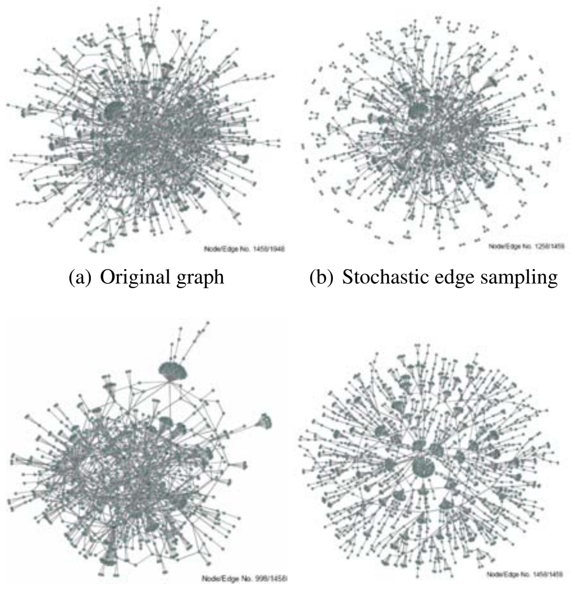
In graph visualization, algorithmic graph preprocessing often includes graph simplification to reduce the size, while maintaining the main graph structure. Also pre-processing of graph properties can be used for graph visualization (in algorithms for positioning of nodes and edges) or highlighting of interesting parts of the graph. This modified graph is used then for an easier visual inspection as large and complex graphs are difficult to understand even using advanced node and edge positioning algorithms (layouts). Such preprocessing steps can usually be performed automatically without user interaction. There are two main approaches to graph reduction: graph filtering and graph aggregation.

Graph filtering There are two types of filtering: stochastic and deterministic. Stochastic filtering is mainly based on random selection of nodes and edges from the original graph. These methods are compared in [LF06]. Deterministic filtering uses, as its name suggests, a deterministic algorithm for selecting of the nodes/edges to be removed. This filtering can be based on node/edge attributes, on topologic values such as betweenness centrality, or other graph properties. For example, filtering based on edge-betweenness-centrality can be used for removal of less important edges while keeping the underlying structure (connectedness and other features such as cliques) of the graph [JHGH08] (see Figure 4).

Graph aggregation Here, nodes and edges are merged to single nodes and edges, thereby reducing the size of the graph and revealing relationships between groups of nodes. Graph aggregation can be repeated multiple times, creating a hierarchical graph. There are various ways of graph aggregation, including using predefined node hierarchies, or aggregation according to node attributes, to name a few [EDG*08] (see Figure 5).

3. Visual Representations of Graphs

For an efficient representation of graphs, aesthetic criteria need to be followed. Beck et al. [BBD09] recently presented aesthetic criteria for drawing graphs. They consider three groups of criteria, which are irrespective of the type of graphic representation: *general, dynamic and aesthetic scalability*.
The *general criteria* include reduction of visual clutter, reduction of spatial misunderstanding resulting from spatial closeness, maximization of spatial matching of items for following paths and maximization of space efficiency.
For *dynamic* graphs, the following criteria are desired: maximization of display stability between time points, reduction of cognitive load when analyzing time dynamics, minimization of temporal aliases mainly owing to positioning of different nodes in the same place in two time periods.
Aesthetic scalability criteria refer to graph readability for larger graphs, i.e., scalability in number of vertices (i.e, increasing graph order), scalability in number of edges (i.e., increasing graph density), and scalability in number of graphs,


(a) Original graph (b) Stochastic edge sampling

(c) Geodesic clustering (d) Structure-based filtering

Figure 4: *Example of various graph reduction techniques. The graphs are visualized using GEM layout [JHGH08]. ©2008 IEEE.*

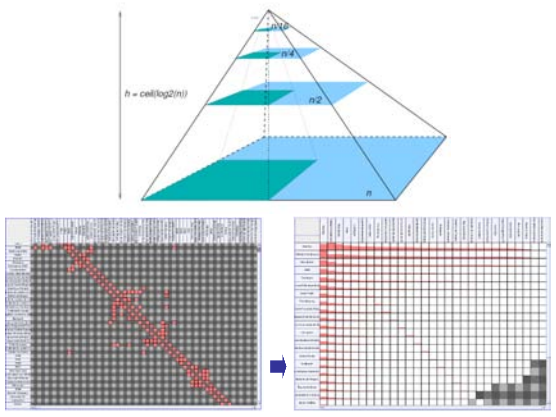


Figure 5: *Graph aggregation for multi-scale graph visualization [EDG*08]. ©2008 IEEE. Top: Graph aggregation schema. Bottom: Example of graph aggregation using a matrix visualization.*

in particular with increasing number of time steps for which graph data is given.

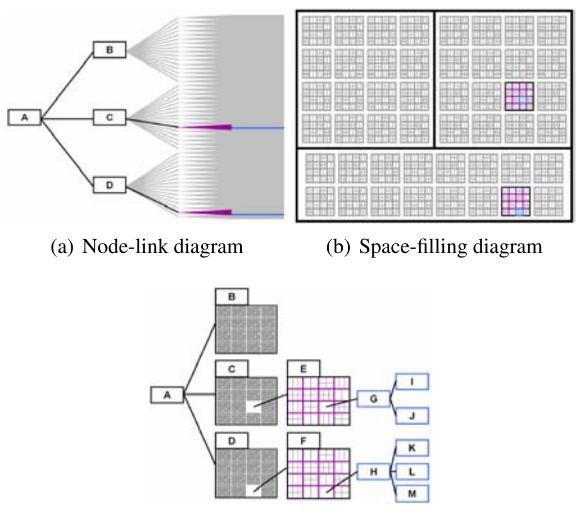
In this section, we describe main graph visualization techniques following the graph classification from Section 2. We introduce techniques for static and time-varying graphs. In each part, techniques for hierarchies, generic directed and undirected graphs, and compound graphs are presented.

3.1. Visual Representations of Static Graphs

The visualization of static graphs has received much attention in the Information Visualization community. Often, static graph visualization serves as a basis for time-dependent graph visualization as described in Section 3.2.

3.1.1. Trees Including Hierarchies

Techniques displaying trees can be divided into three main groups: *Space filling, node-link based, and combined* (see Figure 6). There have been several studies comparing the different ways of tree visualization, in particular hierarchy visualization [BN01,AK07,Kob04,Sta00,vHvW02]. In general, it is difficult to unify these results as they differ significantly. Recently, it has been found that the effectivity of the respective technique largely depends not only on the task to be solved, but also on the formulation of the task assignment, i.e., if it reflects a containment or a levels metaphor [ZK08].



(a) Node-link diagram (b) Space-filling diagram

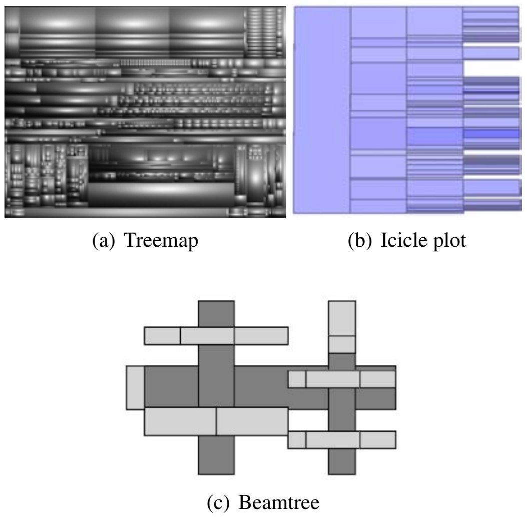
(c) Combined representation

Figure 6: *Three types of hierarchy visualization techniques. a) Node-Link, b) space-filling, c) combined. [ZMC05], ©2005 IEEE.*

Space filling techniques These are mainly applied to rooted trees. They use the spatial position of the nodes (such as closeness or enclosure) to represent the hierarchic structure of the graph. Moreover, they try to use the full area of the display to present the graph. They are mainly used to visualize the hierarchic partitioning of the set of all data items into partitions, e.g., when considering the set of files in a standard file system. The size of the nodes is encoded by the area size of the displayed items. Additionally, color and height can represent additional data attributes. In case more complex additional information needs to be displayed, specialized data presentations can be placed in the child nodes such as icons, parallel coordinate diagrams, etc. Space-filling techniques can be categorized by the placement strategy employed into enclosure, adjacency and crossing (see Figure 7).

- *Enclosures* These techniques recursively layout child nodes within the area of their parent nodes. The most prominent examples are treemaps – rectangular shapes recursively subdividing rectangular display space according to the underlying hierarchy, introduced by Shneiderman [Shn92] (so called slice-and-dice algorithm). They can be displayed both in 2D [BSW02] and 3D [SLS07]. Variants include using Voronoi tessellations [BDL05] or bubble layouts [Bed01]. Further types, such as elliptic [OCNF09] or circular shapes have been proposed but by definiton cannot fully use rectangular input display area as the child nodes do not fully cover the parent nodes.
  The main advantage of enclosures is the very good usage of the available space, as the child node do not need extra space owing to the overlap with the parent nodes. The disadvantage is that the overlapping of the parent nodes may also lead to a more difficult distinction of the hierarchy structure by the user, as it is rather implicitly encoded. For treemaps, several layout techniques have been developed including ordered (i.e., pivot-based) [BSW02], squarified [BHvW99], and spiral [TS07] treemap layouts. For example, squarified treemaps aim at generating subrectangles of square-like aspect ratios, supporting easier comparison of sizes and presentation of additional diagrams or other elements within the rectangles. According to Tu and Shen [TS07], the slice-and-dice algorithm leads to high aspect ratios with high readability. Strip, pivot-based and spiral techniques have medium aspect ratios with medium readability. Squarified treemap has very good (low) aspect ratios but low readability. In order to better distinguish the hierarchical structure, cushion treemaps [vWvdW99] apply shading of the shapes. Treemaps that reflect the geographic distribution of the hierarchical data were presented in [WD08].
- *Adjacency* In contrast to treemaps, adjacency-based techniques do not overlap the parent nodes by child nodes and instead, represent the node relationships by placing the child nodes next to their parent nodes. The placement can be in circular layers such as in the SunBurst method (2D [SZ00] or 3D [SKW*07] variants), or on linear layers, yielding so-called "icicle plots". The advantage of this visualization is that the parent nodes are not overlapped by their child nodes and therefore, their attributes can be more easily displayed and analyzed. However, this visualization consumes more space.
- *Crossings* The crossing method places child nodes across the parent node, thereby only partially overlapping the parent. The "Beamtree" method [vHvW02] improves over the classic Treemap problem where the hierarchic structure may be difficult to visually assess, while still being more space efficient than the adjacency techniques. The main drawback of this technique is that if users are un-

familiar with this approach. It is often less readable than other methods.


(a) Treemap (b) Icicle plot

(c) Beamtree

Figure 7: *Three types of space filling hierarchy visualization techniques. a) Enclosure – Cushion treemap [vWvdW99], ©1999 IEEE. b) Adjacency — Icicle plot [TS08a], ©2008 IEEE. c) Crossing – Beamtrees [vHvW02], ©2002 IEEE.*

Node-link techniques These approaches use links between items to depict their relationship. Layout algorithms controlled by optimization criteria or layout heuristics calculate a layout for the positions of the nodes. The method by design typically leaves significant background space empty and thereby may encounter scalability problems when applied to larger graphs. Many layout algorithms have been proposes to date in the Graph Drawing community. They include radial or balloon layouts in 2D [HMM00], Cone trees [RMC91] in 3D, point based trees [SSH09], nature inspired Phyllo trees [NCA06], or Hyperbolic layouts [Mun97, AH98] (see Figure 8). For the visualization of node attributes, specialized techniques for multi-dimensional data visualization such as glyphs, radial or parallel plots can be used.

Combined approaches These approaches combine node-link diagrams with treemaps. In these, a part of the hierarchy is displayed in an enclosing (treemap) mode, and the rest as a node-link diagram (see Figure 6c). They present the data in a flexible space-efficient way while still clearly presenting the data structure and emphasizing the content. The most prominent representative are "elastic hierarchies" [ZMC05]. In connection to interactive determination of the type of visual metaphor used for each part of the hierarchy, this technique allows for flexible analysis of the data using advantages of both representations.

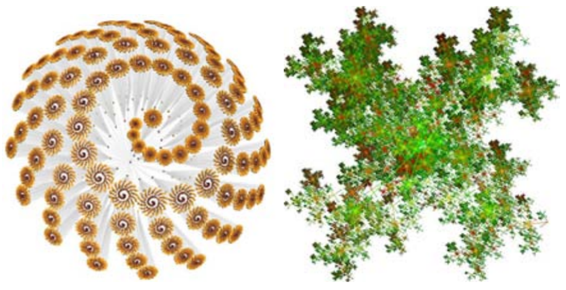

(a) Phyllotrees (b) Point-based tree

Figure 8: *Examples of node-link tree visualizations. a) Phyllotrees [NCA06], ©2006 IEEE. b) Point-based tree [SSH09], ©2009 IEEE.*

3.1.2. Directed and Undirected Graphs

Graph visualization techniques can be classified according to the visual metaphor used into node-link, matrix or combined representation (see Figure 9 for an illustration). A comparison of node-link and matrix techniques is presented by Ghoniem et al. [GFC04]. According to the study, the advantage of node-link diagrams is their intuitiveness, compactness, and better suitability for path following tasks. They are more effective for smaller and sparse graphs. Matrix representations inherently do not have edge crossings and node overlapping problems, and are thereby suitable also for dense graphs. When using appropriate node ordering, they can easily reveal dense substructures in the graph. However, they also suffer from scalability in limited display spaces, especially for very large graphs. In visual graph analysis, graph layout and matrix ordering influence the effectiveness of these representations. These issues are therefore in the core of graph visualization research.

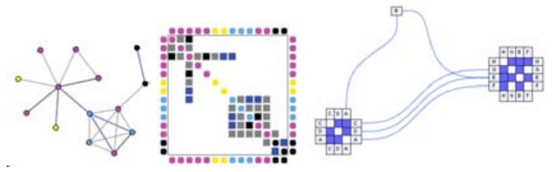
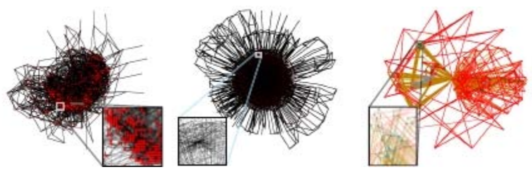

(a) Node-link diagram (b) Adjacency matrix diagram (c) Combination

Figure 9: *Three types of general graph visualization techniques: a) Node-link diagram, b) adjacency matrix, c) combination. From [HFM07], ©2007 IEEE.*

Node-link representations The main challenge is the placement of the nodes so that graph readability and certain notions of graph aesthetics are supported (see Figure 10 for an illustration). Typical requirements include that the nodes do not overlap, the number of edge crossings is minimized,

edge length is homogeneous, and in general, graph substructures are easily recognizable. This problem is intensively studied in the *graph drawing community*. Given these aesthetic goals and constraints, the aim is to find algorithms that efficiently provide good solutions. An overview of graph drawing algorithms is given by Battista et al. [DBETT99].
The graph layout field is very large, and an extensive survey of proposed techniques is beyond the scope of this report. There has been a dedicated state-of-the art report by Diaz [DPS02] summarizing techniques up to 2002. Moreover, the related work part in [AAM07, MM08] as well as the comparison in [HJ07] nicely summarize many current techniques. In our report, we classify the techniques according to the type of node placement.

- *Force-based layouts.* These techniques are based on a simulation of mechanical laws by assigning forces among nodes and edges. Basically, the forces between edges correspond to springs and the forces between nodes to electric forces between charged particles. The classic techniques lead to pleasing results for small graphs up to a hundred nodes (examples are the Fruchterman-Reingold [FR91] and the Kamada-Kawai [KK89] layouts). They, however, do not scale well to graphs of thousands of nodes or more. For larger graphs, other approaches have been introduced (see below). For example, the GEM algorithm [FLM95] uses heuristics for faster calculation of forces.
- *Constraint-based layouts.* This family of layouts extends the force-directed approach with constraints on node position. These constraints include horizontal and vertical alignment of nodes, non-overlapping nodes, edge direction or closeness of grouped nodes [DMW09a]. An example are orthogonal layouts, where the edges are only composed of straight vertical and horizontal lines. These layouts can be supported also by user interaction (see also Section 4). Example works from this category include [DMS*08, DMW09b, DMW09a].
- *Multi-scale approaches.* These techniques first lay out a coarser graph (a subgraph of the original graph) and then include more nodes in a level-by-level fashion. Exemplary works include [GK01, FT07, KCH02, HJ05, MM08] (see Figure 10a for an illustration). These methods are typically much faster than traditional force-directed methods. They can be differentiated according to the technique used for creating the node hierarchy, and the layout of the resulting layers. For example, [MM08] employs node clustering and subsequent positioning of the nodes along space filling curves.
- *Layered layouts.* These approaches, also called "hierarchic layouts", place nodes of the graph on parallel horizontal layers, e.g., [Bab02]. They are mainly used for directed graphs and are based on the Sugiyama approach [STT81]. It works in four phases: (1) cycle removal, (2) assignment of nodes to levels, (3) reduction of edge crossings and (4) assignment of coordinates to nodes. Improvements to these layouts, specifically for cyclic graphs, position all nodes of a cycle within one level; examples include the Dig-Cola layout [DK05] and Cyclic Leveling [BBBL09] (see Figure 10b).
- *Further approaches.* Other approaches exist that combine the previous techniques, or use completely alternative approaches to graph layouts. Projection of a node layout from high-dimensional to two-dimensional space has been proposed in [HK02]. LGL [ADWM04] uses a layout of the minimum spanning tree as a basis for the drawing of the whole graph. TopoLayout [AAM07] uses topologic properties of the graph parts, to choose the best graph layout. A layout revealing specific graph substructures (motifs) was presented in [KSS06]. The ISOM method [Mey98] applies the Self-Organizing Map algorithm [Koh01] for finding a suitable graph layout. A graph layout visualization based on the semantics of the graph (on node labels) was presented in [SA06]. Semantically identical nodes (e.g., with the same labels) are placed in boxes using standard layout algorithms (e.g., force-directed) (see Figure 11).


(a) Multi-level graph layouts

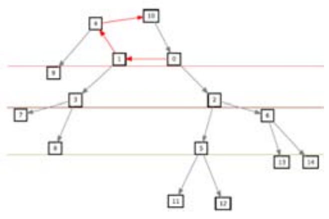

(b) Dig-cola layout

Figure 10: *Graph layout examples. a) A comparison of multi-level graph layouts GRIP, FM^3 and Topolayout [AAM07]. ©2007 IEEE. b) Layered layout of cyclic directed graph [DK05]. ©2005 IEEE.*

Comparison of graph layouts A recent comparison of the readability of graph layouts using eye-tracking [Hua07, PSD09] has shown that force directed layouts outperform orthogonal and layered layouts on various user tasks. Another comparison of advantages and disadvantages of numerous current layouts was published by Hachul and Jünger [HJ07]. They compare the graph drawing outputs according to various criteria finding that the HDE layout [HK02] is very fast but frequently produces layouts with many overlapping edges. In contrast, FM^3 [HJ05] creates pleasing layouts in reasonable time. Both algorithms together

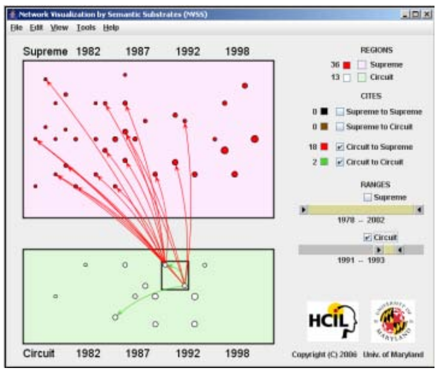
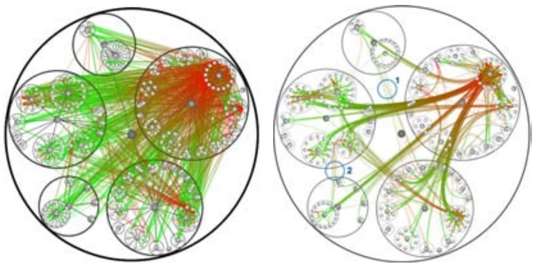

Network Visualization by Semantic Substrates (NVSS)
File Edit View Tools Help
Supreme 1982 1987 1992 1998
Circuit 1982 1987 1992 1998
REGIONS
36 Supreme
13 Circuit
CITES
0 Supreme to Supreme
0 Supreme to Circuit
18 Circuit to Supreme
2 Circuit to Circuit
RANGES
Supreme
1978 – 2002
Circuit
1991 – 1993
HCIL
Copyright (C) 2006 Univ. of Maryland

Figure 11: *Graph visualization using data semantics [SA06]. ©2006 IEEE.*



(a) Original graph (b) Edge bundling

Figure 12: *The use of edge bundling for improving graph readability. a) original graph b) graph with edge bundling. [Hol06], ©2006 IEEE.*

with GRIP [GK01] scale well with graph size. A comparison of user-produced vs. automatically generated layouts [vHR08, DLF*09] found also that the results of physics-based algorithms, such as force-directed layouts, were preferred by the users.

Design of graph drawing The above mentioned techniques cover graph layout. In addition to specific layouts, occlusion and readability of the display can be improved by edge-bundling [CZQ*08, Hol06] (see Figure 12) and the removal of node overlap [GH09, IAG*09]. Drawing of node-link diagrams also includes a suitable design of edge and node drawing primitives. For directed graphs, the representation of edge directions is of importance. There are multiple design possibilities including usage of arrows, color transitions (from color A to color B), thickness transitions (from thick to narrow), curves, and animated textures [HvW09, TK08, BBG*09]. These options may also be combined. A comparison of graph drawing different ways to represent edges was presented in [HvW09]. It shows that arrows, although popular and widely used, do not perform as well as color and thickness transitions. Graph nodes and edges often have associated attributes that are included in the analysis. This study did not concentrate on attributed edges. For such edge attributes, in particular edge weight, coloring of edges or edge thickness can be employed. For the visualization of node attributes, a visualization of multivariate data items (e.g., glyphs or radial plots) is employed.

Visualization of multiple graph connected components For the visualization of multiple components, first layout for each individual connected component is calculated and then a specific placement of these components on the screen is performed. The mostly used placement method is called *packing*. It lays out the components so that they do not overlap and are space efficient. Dogrusoz [Dog02] compares several two-dimensional packing algorithms for graphs which use representation of graphs by their bounding rectangles. They include strip packing, tiling and alternate-bisection. The polyomino algorithm of Freivalds et al. [FDK02] uses polyomino representation of the graph objects, which substantially reduces the unused display space in comparison to rectangular shapes. Goehlsdorf et al. [GKS07] introduce new quality measures to evaluate a two-dimensional placement which yields more compact layouts than the previously mentioned approaches.

Matrix These techniques visualize the adjacency matrix of a given graph, where edge attributes are encoded in the matrix cells. It can display both directed and undirected graphs, where the latter leads to a symmetric matrix. The advantage of this representation with respect to node-link representation is the non-overlapping display of graph edges, and the easy readability of the graph especially for larger and more dense graphs. The disadvantage is an increased difficulty for users to follow paths, and a possible unfamiliarity of matrices to the users. In a matrix visualization, the ordering of rows/columns plays an important role. Different strategies to sort the matrix prior to visualization can be employed (see Figure 13 for an illustration). A proper reordering can reveal clusters in the graph and other patterns. For a discussion of these, we refer to [EDG*08, HF06]. Although matrices are suitable for larger graphs, they also suffer from scalability issues as they use linear order of nodes along the matrix rows/columns. Therefore, interaction techniques and aggregated displays have been proposed [EDG*08, HF06, AvH04, vHSD09, vH03] (see also Sections 4 and 5).

Combination of matrix and node-link approach Techniques using a combination of the two previous approaches aim at overcoming their limitations by focusing on their strengths. Three main approaches exist (see Figure 14).

- *Multiple synchronized views.* These techniques link the matrix and node-link representation [HF06]. Both views show the same data and are synchronized during explo-

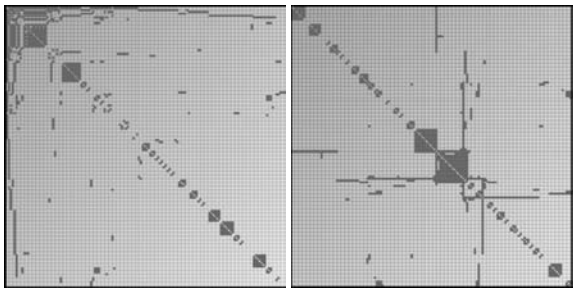

(a) HDE matrix ordering (b) NNTSP matrix ordering

Figure 13: *Examples of matrix reordering on graph presentation. a) Using HDE algorithm. b) Using NNTSP reordering. From [EDG*08], ©2008 IEEE.*

ration. Thereby, the user can concentrate on whatever view is more suitable for the current task.

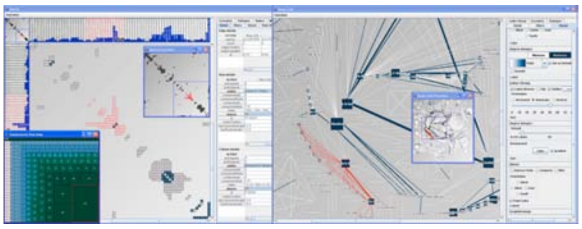
- *Matlink.* [HF07] This approach enhances matrix visualization with links at the border of the matrix (connecting the nodes). Using link highlighting, the paths can be easily spotted in the Matlink view and at the same time, the advantages of the matrix representation are retained.
- *NodeTrix.* [HFM07] It combines both representations in one view, where node-link diagrams display the overall graph structure of the network, and adjacency matrices show communities. The work also discusses three ways of link display for this setting: aggregated links, underlying links, and underlying links with full size (see Figure 15). These forms can be also used for attributed links.

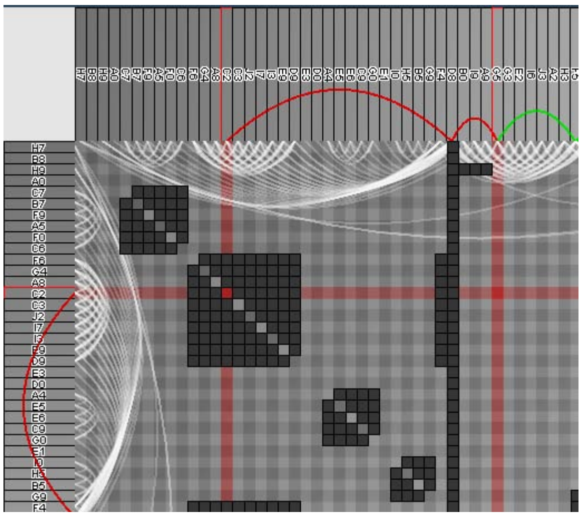
3.1.3. Compound Graphs

Literature on visualization of graphs with hierarchic structure is relatively rare. We identify three main approaches.

Node-link graph visualization techniques These use node-link diagrams for the lowest hierarchy level and then use "bubbles" (enclosures) for various hierarchy levels. Examples include TugGraph [AMA09] and GrouseFlocks [AMA08]. The advantage of this method is its intuitiveness. However, for large graphs with many links, this view gets easily overcrowded (see Figure 16 a). Edge overplotting problem can be partially solved by edge bundling [Hol06] (see Figure 12). Alternatively, only links between merged nodes can be drawn (see Figure 16 c).

Treemap-based A Treemap visualization of the node hierarchy uses overlaid links between nodes [FWD*03] (see Figure 16b). This approach may suffer from strong overplotting in case of many links between nodes of the hierarchy. Therefore, edge bundling is advised to improve the readability of the display [Hol06](see Figure 12). Similarly, also one-dimensional Treemaps with links between nodes, so called


(a) Multiple linked views


(b) Links connected to the matrix view

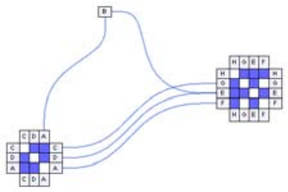

(c) Node-link and matrix combined

Figure 14: *Combined matrix and node link graph visualization techniques. a) Multiple linked views [HF06]. ©2006 IEEE. b) Links connected to the matrix view [HF07]. ©2007 Springer-Verlag Berlin Heidelberg. c) Node-link and matrix combined - part and part [HFM07]. ©2007 IEEE.*

ArcTrees [BDJ05] can be employed (see Figure 16d), but these do not scale well for large hierarchies.

Matrix view with links These visualizations combine the generic node relationship visualization with a tree-based visualization of the hierarchic node relationships. This is an analogy to MatLink [HF07]. This view is very clear, however, it may be difficult to understand the compound relationships between nodes (see Figure 16e).

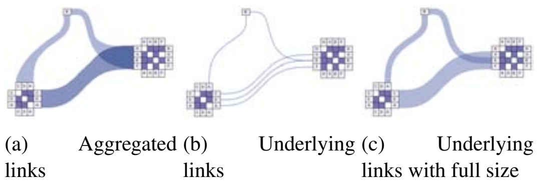


Figure 15: *Three ways of link visualization in a combined node-link and matrix data representation using the NodeTrix approach [HFM07]. ©2007 IEEE.*

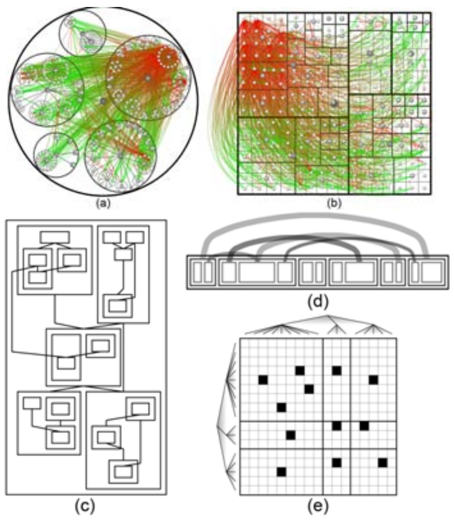


Figure 16: *Visualization techniques for compound graphs [Hol06], ©2006 IEEE. (a) Node-link visualization with grouped nodes in "bubbles". (b) Links overlaying a Treemap visualization. (c) Compound drawing using enclosures and links between merged nodes. (d) ArcTrees - links overlaying a 1D Treemap [BDJ05]. (e) A matrix view for showing relations between entities linked with tree view of the nodes as in MatLink approach [HF07].*

3.2. Visual Representation of Dynamic Graphs

In this section, we discuss two categories of visual display of the time changes on graph elements: Using animation, and using static displays. Animated displays usually employ or enhance static visualization techniques such as presented in Section 3.1. Animation is a natural way of conveying the change of the data over time. However, its effectiveness is limited by human perception capabilities. Usually, users are able to recognize and remember larger changes in the data. The static view is preferred for more detailed analysis of data changes. Static views which also incorporate the time-dimension of the data are more complex. In the following, we categorize the visualization techniques according to the type of data changes captured into those that affect only *data attributes*, and those that affect also *data relationships*.

3.2.1. Trees Including Hierarchies

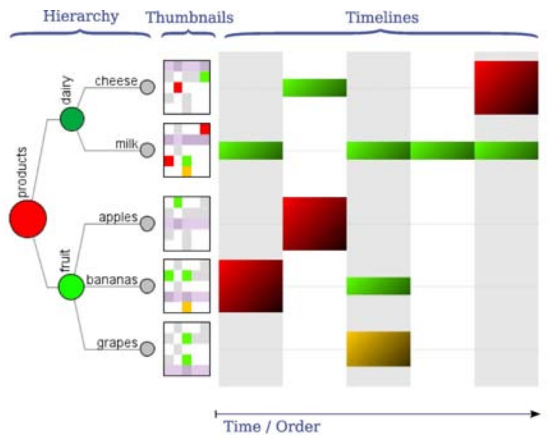
For the visualization of dynamic trees with only data attribute changes, either Treemaps with time series in the leaf nodes [SKM06, DHKS05] or the so called Timeline Trees [BBD08] can be used (see Figure 17 a and b). Timeline trees show the hierarchy on one side and the time sequences on the other side of the view. The Treemap representation directly shows the hierarchic structure and time-variation in one combined view. This allows for an easy comparison of the time-developments across the hierarchy. However, the comparison is affected by different node sizes and difficult for small nodes. Therefore, a specific Treemap layout preserving the aspect ratio has been developed [SKM06, DHKS05]. Timeline Trees assign the same space to all nodes. The vertical positioning of time lines allows for very good comparison of the values at the same time points. The separation of the time dimension from the hierarchic structure, however, complicates the comparison of tree branches.

For visualization of dynamic data with structural changes, animated views are used. In this respect animated graphs (see Section 3.2.2) can be employed in general. In particular, the layouts based on the Sugiyama approach [GBPD04] are suitable. Alternatively, animated treemaps [GF01, TS07] or icicle/circular plots [TS08a] can be used (see Figure 17 c). When choosing the graph layout, the layout stability needs to be taken into consideration. E.g., in the treemap representations, the spiral layout [TS07] achieves a high continuity with high stability of the layout. Strip and pivot-by-middle layouts have also been shown to have higher layout suitability. All these layouts are preferable in spite of their higher aspect ratios in comparison to the Squarified Treemap. Tu and Shen [TS07] propose also static comparison of two time points in a Treemap visualization (called contrast Treemap).

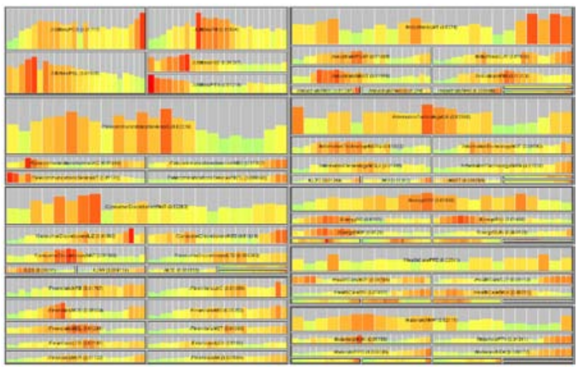
3.2.2. Directed and Undirected Graphs

For attribute changes only, techniques for visualization of static graphs can be combined with visualizations of individual time dependent data items (e.g., color charts [SLN05]) (see Figure 18a). The advantage of this approach is the large number of the available graph layouts.

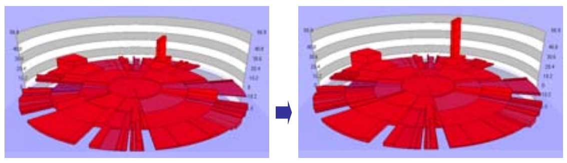
In case of structural changes, time-dependent graph layouts (animated graphs) need to be employed [KG06, DGK01]. In animated graph visualization (in analogy to animated tree visualization), a stable graph layout, which changes minimally, is of essence. A stable graph layout preserves the mental map of the user and therefore, facilitates the analysis of graph changes. In laying out dynamic graphs, there is a large difference between strategies for drawing graphs with known histories and those that need to be adjusted in real-time depending on new data streams. A paper of Frishman and Tal [FT08] addresses this particular issue by proposing an online algorithm for dynamic layout implemented on the GPU, thereby accelerating the layout computation (see Figure 18b).



(a) Time line tree



(b) Treemap with time series



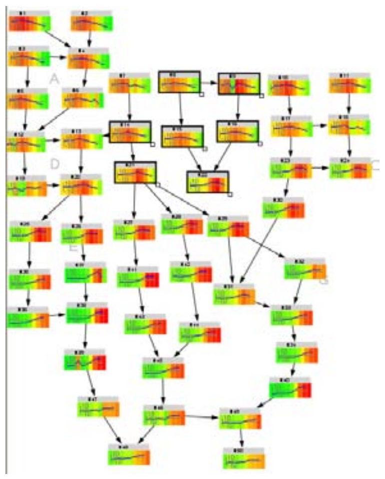
(c) Animated circular icicle plot

Figure 17: *Visualization of time-dependent trees. (a) Time line tree [BBD08], ©2008 ACM. (b) Time series in the Treemap nodes [DHKS05], ©2005 IEEE. (c) Animated hierarchic circlular plots [TS08a], ©2008 IEEE.*

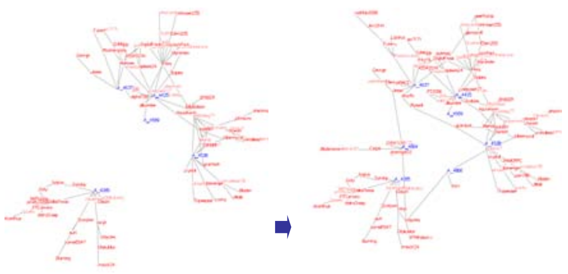
3.2.3. Compound Graphs

There are only few techniques that visualize time-varying compound graphs. They employ either animation or static data representations.

Kumar et al. [KG06] present a specific layout for *animation* of a node-link diagram with transparent "bubbles" for the hierarchic grouping of nodes (see Figure 19a). Frishman and Tal [FT04] present a layout which focuses on maintaining the clustered structure during the animation. The groups of nodes are displayed using bounding boxes around the



(a) Node-link diagram with time series in nodes



(b) Animated node-link diagram

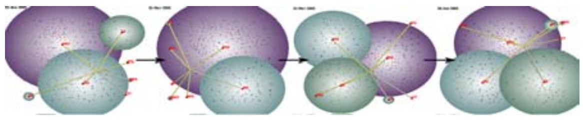
Figure 18: *Visualization of time dependent graphs. (a) Time series in nodes [SLN05], ©2005 IEEE. (b) Animated graphs [FT08], ©2008 IEEE.*

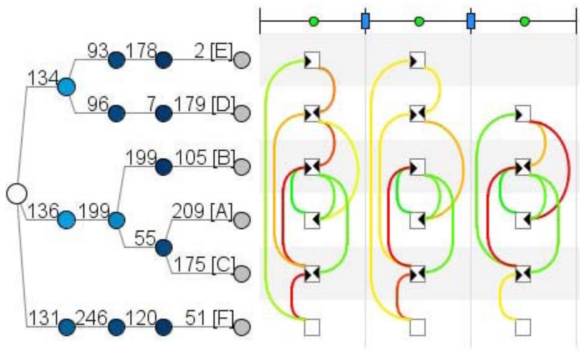
groups. Reitz et al. [RPD09] use dynamic graph layouts for showing areas of interest in dynamic compound graphs.

A *static approach* to visualization of dynamic compound Digraphs using TimeArcTrees was presented by Greilich et al. [GBD09] (see Figure 19b). They show a sequence of node-link diagrams with horizontal node alignment in a single view, thereby supporting their direct comparison. TimeRadarTrees [BD08] use radial tree layouts for the hierarchy and a sequence of circle segments for representation of the temporal change of the structure (edges) of the Digraph (see Figure 19c). This view easily gets complex for larger graphs.

4. User Interaction in Graph Visualization

An overview of interaction techniques in Information Visualization is presented in [KHG03]. Standard interaction techniques such as zooming, panning, brushing and linking [CMS99, War00] can also be applied in graph visualization. However, additional specialized interaction techniques


(a) Animated compound graphs


(b) TimeArcTrees

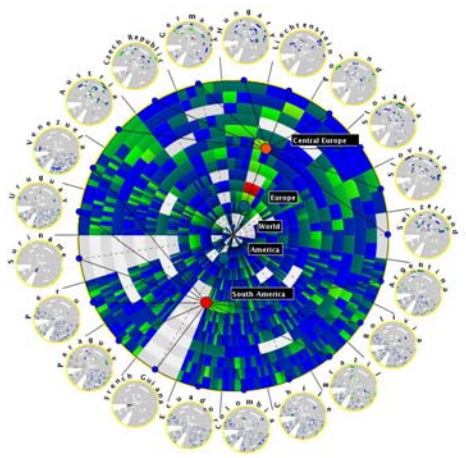

(c) TimeRadarTrees

Figure 19: *Visualization of time varying compound graphs. (a) Animated graphs by Kumar et al. [KG06], ©2006 IEEE. (b) TimeArcTrees [GBD09], ©2009 held by the authors. (c) TimeRadarTrees [BD08], ©2008 held by the authors.*

have been developed for interactive visual graph navigation and exploration.

Recently, Yi et al. [YKSJ07] presented a general *taxonomy of interaction techniques*. This taxonomy is based on a broad literature survey of available taxonomies. It categorizes interaction according to *user intention* into seven categories:

1. Select: mark something as interesting,
2. Explore: show something else,
3. Reconfigure: show a different arrangement,
4. Encode: show a different representation,
5. Abstract/Elaborate: show more or less detail,
6. Filter: show something conditionally,
7. Connect: show related items.

Alternatively, user interaction can be categorized according to the *action* that is taken by the user. This categorization is more suitable for dividing interaction techniques into categories, as each action is supported by the employed technique. The two categorization approaches are interrelated. A user intention can be achieved by several user actions or, vice versa, an action can suit several intentions.

We categorize interaction techniques according to whether the action of the user affects the data (the selection of the displayed data or the data values) or the visual display of the data itself (visual parameters or visual representation). Data and view manipulation can be used for interactive data exploration and navigation. This categorization follows the idea of Elmqvist and Fekete [EF09] and Bertini and Lalanne [BL09]. It is in line with the Information Visualization reference model of Card et al. [CMS99]. Please note that these two types of interaction are often closely connected. For example, data manipulation may automatically lead to changes of visual parameters (e.g., data filtering can influence the graph layout, or zooming can be combined with data filtering forming a type of semantic zooming). Such techniques that combine both types of techniques are assigned to one of the categories and marked "(*)".

4.1. Data Manipulation

Data manipulation affects the selection of the data to be displayed, or may change the data values.

4.1.1. Data selection

These interaction techniques influence which parts of the data set are displayed. The data selection may follow three paths.

A top down approach This approach starts from the whole graph and then constrains the part of the data set to be visualized by filtering according to criteria or by manual data selection. The disadvantage of this approach is the need to show the whole graph at the beginning, which may require higher computational time for the layout and may lead to occlusions owing to the limited screen size. The advantage is gaining an overview of the graph structure first and then concentrating on interesting parts.

A bottom up approach This approach starts from one selected node [vHP09, Fur86, AF07] and successively shows more nodes/connections on demand. There are two main methods of choosing the additional nodes/edges to be displayed: based on graph structure, or based on a degree-of-interest function. The advantage of this approach is that only the most interesting part of the data set is visualized, however it is difficult to determine the starting point for the exploration and to define the degree-of-interest function. Therefore, we consider these methods in more detail.

- *Navigation based on graph structure.* These techniques reveal/hide that part of the graph that is determined by the connections between nodes. In graphs, *neighborhood traversal* shows neighbor nodes of a focus node up to a certain level [HB05]. For hierarchies, several traversal methods for have been described in [EF09]. The *hierarchy traversal* methods include: (1) above traversal, where nodes up to a certain level are shown; (2) below traversal, where nodes starting from a selected level are displayed; (3) level traversal, where nodes at a certain level are displayed; (4) range traversal, where nodes in a range of levels are shown; and (5) unbalanced traversal, where certain branches of a tree are visible (see Figure 20).

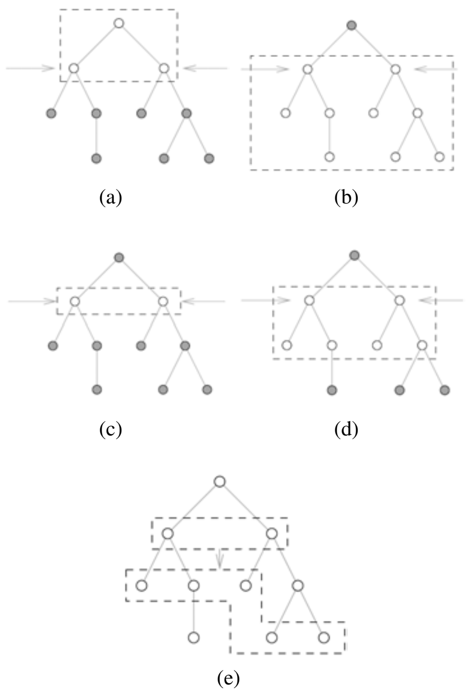


Figure 20: *Hierarchy traversal strategies [EF09], ©2009 IEEE. (a) above traversal, (b) below traversal, (c) level traversal, (d) range traversal. (e) unbalanced traversal.*

- *Navigation based on a degree of interest function.* These methods start from a selected node, and next the edges and nodes of highest interest are shown [Fur86, vHP09]. For the determination of the interesting nodes, a specific degree of interest (DOI) function is used. Depending on the specification of the DOI function, various graph exploration paths can be followed. These DOI functions were used for building specific views on trees (DOITrees) [CN02, HC04]. In the work of Furnas [Fur86], the DOI of a node depends on the distance to the node in focus and the a priori interest in this node (e.g., according to node importance in the network, or node properties). van Ham and Peer [vHP09] extended this function with user interest (UI), which reflects the current specific exploratory focus of the user.

A middle-out approach This method combines both bottom-up and top-down approaches. It starts with a coarsened graph (*middle*) and then interactively either reduces or increases the graph coarsening level by hiding visible nodes or showing additional nodes [WMC*09]. For determining the middle coarsening level and the next interactive steps, graph algorithms are used (see Section 5).

4.1.2. Changes of data values

In these approaches, the change of the displayed data set result from direct data value manipulation. Specifically, the user can change the data values on one level or create/change graph aggregations.

Graph editing The user can interactively delete or add nodes or edges directly in the visual interface. These graph editing actions trigger adjustment of the layout, while still maintaining the layout style and, where reasonable, the current layout topology. Graph editing affects the structural properties of the graph. In particular, the changes can affect specific types of subgraphs (so-called motifs). Automatic identification and highlighting of such structural changes was presented in [vLGRS09].

Interactive graph aggregation For simplification of graphs, graph aggregation is often used. The graph aggregation can be pre-defined, or determined interactively by the user [AMA08, AMA09, HF06]. E.g., GrouseFlocks [AMA08] allows the user to add and remove aggregated nodes on demand (see Figure 21). This allows for variable views on the graph and its structure.

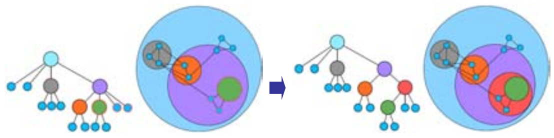
4.2. Changes of Visual Display

In these approaches, the change of the visual presentation of the data concerns adjusting the type of visual presentation and its parameters.

4.2.1. Changes of Visual Parameters

These techniques affect the parameters of the visual presentation. They include highlighting of items, zooming, panning, view distortion, and other techniques.

Highlighting The emphasis of interesting items is a standard interaction technique. Recently, new techniques for highlighting a node and its neighborhood using hotbox and lasso selections were presented in [MJ09].


(a) Creating new aggregation node

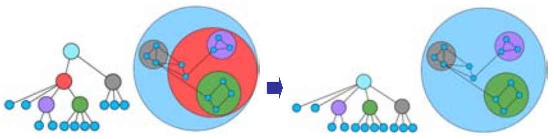

(b) Deleting an aggregation node

Figure 21: *Interactive editing of a graph hierarchy. a) Creating a new aggregation node by merging of nodes. b) Deleting an aggregation node, thereby revealing the underlying merged nodes. From [AMA08], ©2008 IEEE.*

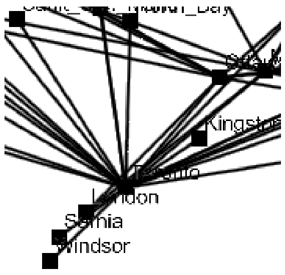
Linking & Brushing Multiple coordinated views are used to show the data from different perspectives. In these views, changes in one visualization (e.g. highlighting) are automatically transfered to the other views. For example, a matrix view coupled to a hierarchical view of the data can be used to reveal important information in the data [AvH04].

Panning Panning allows to navigate in any direction in the view. For graphs, a specific type of panning (*guided panning*) has been proposed. It allows to navigate along edges of a selected node and thereby to explore the structure of the graph. It can be combined with automatic zooming on the edge and distortion of end-node position closer to the currently selected node [MCH*09].

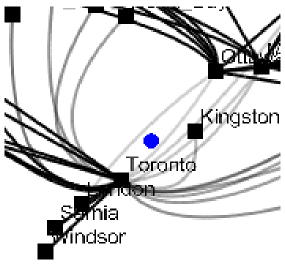
Semantic Zooming(*) Semantic zooming combines zooming with an increasing level of detail. In particular, graph aggregation can be used for gaining a coarser view on a large graph. The semantic zooming increases the level of detail by drilling down to lower levels of aggregation of the original data [EDG*08, AvH04].

Distortion techniques Owing to the limited display space, showing the whole data set leads to strong overplotting or very small (up to, unreadable) data items. Distortion techniques allocate more space to items in focused areas and thereby, improve the readability of the data of interest. They are used both for node-link and space filling graph visualization techniques. The distortion can concentrate either on one area or on multiple areas of the screen. The distorted views are also called *fisheye* views. Interactive selection of the focus area helps to explore different parts of the data in more detail.

- *Single focus.* Graphical fisheye views were introduced in [SB92]. So called *edge lenses* resolve strong overlaps of edges in the view. They displace the edges to a larger area [WCG03] (see Figure 22). This approach is especially useful for geographic-based graphs, where node-edge repositioning is not desired and therefore, cannot help to solve edge overlap. Another approach uses filtering of interesting edges in a specified area (*), or moving neighbor nodes closer to a selected node relying on the graph structure [MCH*09]. This type of node position change can be combined with geometric view distortion [TAvHS06] (see Figure 23). In node-link visualization of hierarchies, a degree-of-interest function can be used for allocating more area to more interesting parts of the tree, e.g., in DOITrees [CN02, HC04].
- *Multiple foci.* Multiple foci distort several view areas at the same time. It is useful for comparing various parts of the display or focusing on several items that are spread across the view. In node-link diagrams either magnification of the areas of interest [TS99, SZG*96] or space folding (shrinking of area out of focus) can be used [ERHF09, MGT*03] (see Figure 28 bottom right). For treemaps, the so-called *balloon focus* can be used for enlarging multiple items in a treemap [TS08b]. This approach keeps the form of other areas keeping relative position of items unchanged (see Figure 24).



(a) Original view



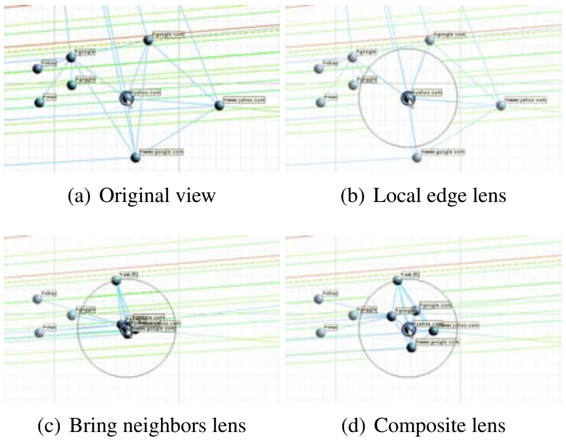
(b) Edge lens

Figure 22: *Example of edge lens interaction. (a) Original view without lens. (b) Using edge lens From [WCG03], ©2003 IEEE.*

4.2.2. Changes of Visual Scheme

Changes of the visual scheme cover changing of the type of data visualization either by changing the layout or by changing the visual mapping.

Layout change In node-link diagrams, layout change (adjustment) affects the positions of the data items on the screen (see Section 3). It can be performed by changing of the layout type with automatic recalculation of the new layout, by manual movement of nodes, or by adjusting the layout parameters including automatic readjustment of the layout. When concentrating on user-defined changes to graph layouts, an approach to easy selection and layout change of nodes and subgraphs was presented in [MJ09]. Furthermore,



(a) Original view (b) Local edge lens

(c) Bring neighbors lens (d) Composite lens

Figure 23: *Examples of different types of edge lens interactions. (a) Original view without lens. (b) Using local edge lens. (c) Using bring neighbors lens. (d) Using composite lens which combines (b), (c) and Fisheye lens. From [TAvHS06], ©2006 IEEE.*

interactive adjustment of the layout constraints was presented in [DMW09a]. For matrix visualizations, user-driven reordering of matrix representation was described in [HF06].

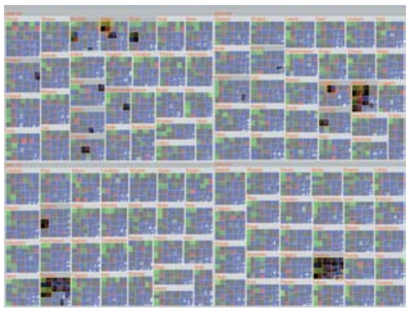
Change of visual representation The change of the type of data presentation, e.g., from a matrix to a node-link diagram was presented in [ZMC05, HFM07]. This change can affect the whole data view [HFM07] (see Figure 25) or only a part of it [ZMC05, HFM07]. By changing of the visual representation, new insights into the data can be reached. In order to be able to follow the changes, smooth animations across transitions should be used.

5. Graph Analysis

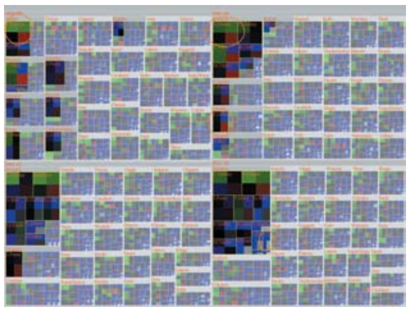
Algorithmic graph analysis is beneficial during all stages of the visual graph analysis process. Relevant techniques allow, e.g., to reduce a large graph to a smaller graph prior to visualization, to search for specific graph structures of interest, or to find similarities and dissimilarities for generating comparative graph views. In this section, we describe important graph analytical approaches.

5.1. Analysis of Graph Structure

In most user tasks, the analysis of the relationships between entities in the graph and the assessment of the global graph structure plays the key role. These tasks may be effectively supported by a combination of algorithmic graph analysis and interactive visualization. The algorithmic methods allow, e.g., to calculate node/edge properties, identify clusters in the graphs, etc., which results are visualized interactively.



(a) Original view



(b) Balloon focus

Figure 24: *Multiple foci in a Treemap. (a) Original view. (b) Using balloon focus. From [TS08b], ©2008 IEEE.*

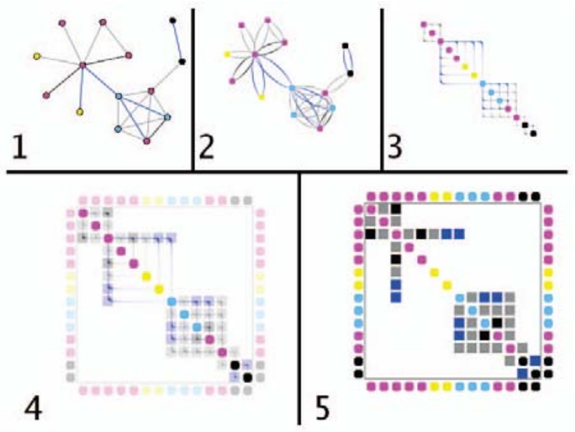


Figure 25: *Transformation of visual representation of a graph from node-link to matrix view. The picture shows five stages of this process [HFM07], ©2007 IEEE.*

In the following, we summarize the methods according to user tasks starting from more simple to more complex tasks.

Identification of important nodes In networks, some nodes play a specific role owing to their position within the network. For example, so called hubs and authorities can be identified and visualized in the network, enabling faster analysis of the graph [OPPROG09]. The importance of nodes and edges is measured by derived quantities such as centrality-based measures [Fre79] and ranking-measures [WS03].

Analysis of connections between two nodes Besides focusing on single nodes, relations between two nodes can be analyzed, typically by calculation and highlighting of shortest paths between the entities. Usually, such analysis is combined with interactive selection of two entities of interest [TK08, HB05, HF07, GBD09] (see Figure 14b).

Analysis of graph substructures In many applications, specific types of substructures (i.e., motifs) play an important role. For example, in social networks, cliques identify highly connected communities, or feed-forward motifs (substructures in form of a triangle where directed edges exist from nodes A to B, A to C and B to C) in biologic networks indicate the functional properties of the network [Sch08]. In order to support the substructure analysis, these motifs can be calculated and visualized in the network [vLGRS09, HFM07, MMO05, MJW*09, SS05] (see Figure 26). The type of structure can be interactively chosen by the user in order to support various analytical tasks.

Analysis of graph structure on several aggregation levels User-defined or data-driven graph aggregation can reveal relationships between groups of entities in a graph. The grouping may be based on categoric node attributes [Wat06], or on a pre-defined node hierarchy [AMA09]. It can also be user-specified [AMA08], or depend on structural properties of the graph [vLGRS09] (see Figures 5 and 21).

Identification of the impact of graph changes on the structural properties In time-dependent graphs, the role of the nodes can change over time, therefore analysis and visualization of topologic properties (e.g., betweenness centrality) of selected nodes has been proposed [PD08, PRB08]. Additionally, when analyzing user-defined changes (in what-if-scenarios) the impact of node or edge deletion/addition on local substructure can be analyzed and highlighted [vLGRS09].

5.2. Graph Comparison

One specifically important analytical task is the examination of the similarities and differences between multiple graphs, especially focusing on structural aspects. Usually, structural differences are in the focus. Such difference may be identified by the identical node labels in both graphs, or by graph matching algorithms. After the matching, visualization is employed to explore the differences [AWW09]. There are various types of analysis which we describe next.

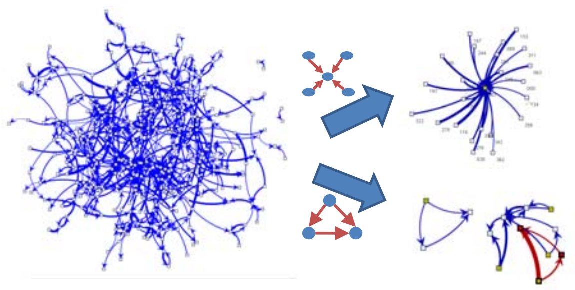


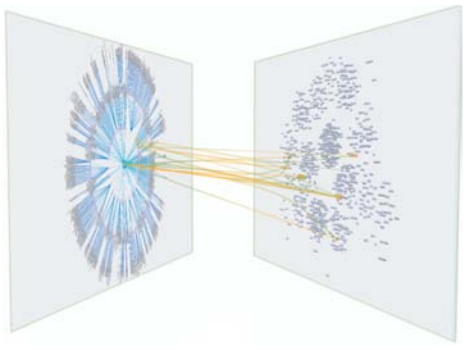
Figure 26: *Interactive graph motif search and visualization. From [vLGRS09], ©2009 held by the authors.*

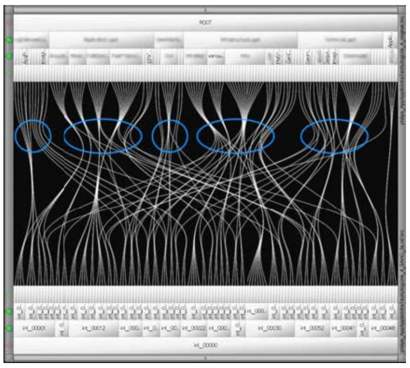
One-to-one node comparison of two graphs Probably the most common task in graph comparison is the matching of individual nodes from one graph to individual nodes of the second graph. The VisLink visualization approach [CC07] was developed to support this task. It shows both graphs on separate planes in 3D, and draws matching links between corresponding nodes (see Figure 27a). For comparison of hierarchies, a similar approach, based on drawing the two hierarchies in opposite parts of the display and linking of their leaf nodes was proposed in [HvW08] (see Figure 27b). In both cases, the visibility of matching links can be increased by edge bundling.

One-to-many nodes comparison of two graphs One-to-many nodes comparison concerns correspondence of one node in one graph to many nodes in another graph. Di Giacomo et al. [GDLP09] developed a system that visualizes these one-to-many connections with low overlapping of links (see Figure 27c).

Structural differences between two graphs When analyzing structural differences between two graphs, analysts are often interested in identifying which links or parts of the graphs correspond to or differ from the other one. For the analysis of trees, the TreeJuxtaposer system supports to analyze and highlight structural differences between two trees [MGT*03] (see Figure 28). For general graphs, Fung et al. [FHK*09] use both multi-level graph views following the VisLink approach [CC07], and overlapping of two networks with highlighting of common structural parts (see Figure 29a). Archambault [Arc09] uses graph aggregation and graph filtering to reveal structural differences between two graphs (see Figure 29b).

Comparison of multiple graphs Clustering of graphs helps gaining overview of types of graphs in large graph databases. The use of Self-Organizing Maps for grouping of graphs according to their structural similarity and visualization of clustering results has been proposed in [vLGS09] (see Figure 30). The proposed system allows for an interactive definition of the graph similarity function, and an exploration of the results.


(a) One-to-one graph matching


(b) One-to-one hierarchy matching

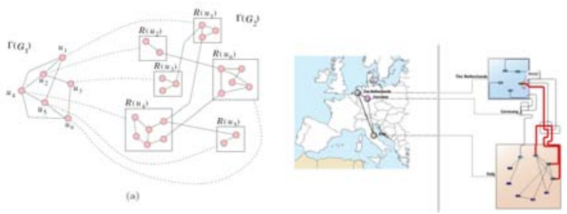
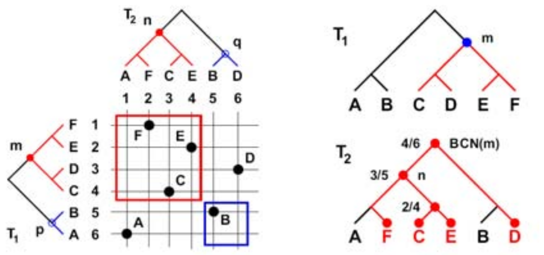

(c) One-to-many graph matching

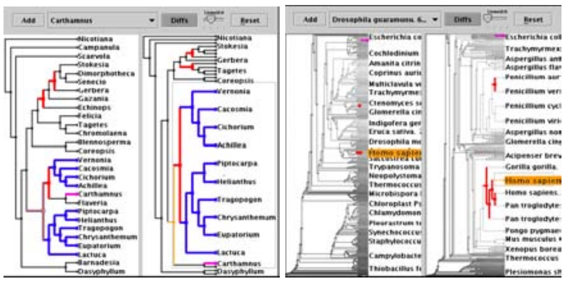
Figure 27: *Visualization of graph comparison. a) One-to-one graph matching [CC07], ©2007 IEEE. b) One-to-one hierarchy matching [HvW08], ©2009 held by the authors. c) One-to-many graph matching [GDLP09], ©2009 Springer-Verlag Berlin Heidelberg.*

6. Concluding Remarks and Future Challenges

Research on visual graph analysis deals with the interrelated issues of graph drawing, graph presentation, human-computer-interaction, and analytics. This state-of-the-art report represents an encompassing overview and systematization of recent developments in this field. Many advances



(a)

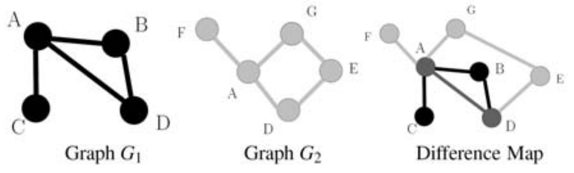


(b)

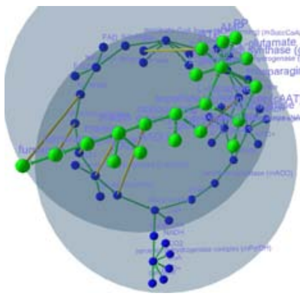
Figure 28: *Tree comparison. a) Schema of the tree comparison. b) Example of tree comparison using highlighting of tree differences. The left view shows the traditional view, the right view is distorted in order to emphasize important parts of the tree [MGT*03], ©2003 ACM*

have been made on individual parts of visual graph analysis. In the following, we attempt an assessment of these, and outline future research challenges in this area.

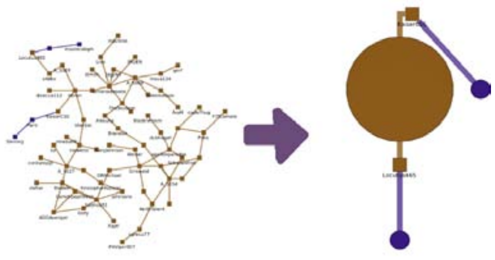
Scalability issues in graph drawing There has been much interest in the development of faster layout algorithms that produce more readable layouts for large graphs, also using parallel computing, as provided e.g., by current CPUs and GPUs. It is recognized that using a combination of automatic graph layout generation and user-oriented, interactive layout steering, better layouts can be obtained. As graphs get larger, graph filtering and aggregation have been the main means of graph simplification allowing to draw them. Alternatively, the limited screen space leading to strong overplotting in large graph visualization can be avoided by drawing graphs on large screens, where specialized layouts can be applied [MGL06]. It can be foreseen that work on more sophisticated graph layouts revealing the main structures in the whole graphs or parts thereof will continue. In particular, user involvement in the graph layout process involving analytical expertise of the user is a promising approach and may lead to easier interpretation of the drawings.
From an analytical perspective, also the understanding of the meaning of the nodes and edges, not only their global structure, is necessary. In particular, the readable/non-overlapping drawing of nodes, edges and their labels is an



(a)



(b)



(c)

Figure 29: *Visualization of structural differences between two graphs. a) A schematic illustration of graph difference. b) Visualization of graph differences using network overlapping [FHK*09], ©2009 IEEE. c) Visualization of graph differences using difference hierarchies [Arc09], ©2009 held by the author.*

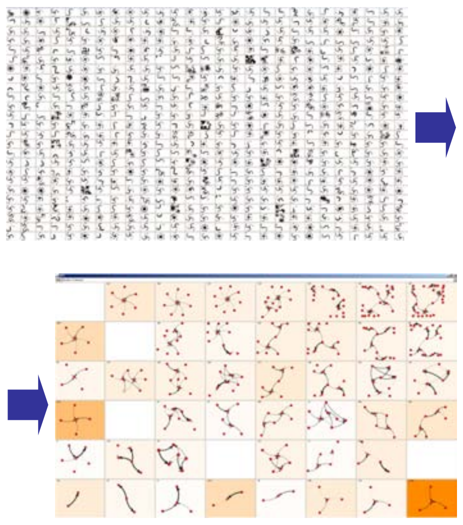


Figure 30: *SOM-based graph clustering for analysis of types of graph data space and similarities between graphs [vLGS09], ©2009 IEEE.*

important issue. When displaying graphs with labels, even smaller graphs can easily lead to overcrowded displays. This topic is gaining more interest in visual analytics research.

Graph types in graph drawing In recent years, the variety of considered graph types has increased substantially. In particular, there has been a large amount of work on drawing dynamic and compound graphs. When drawing dynamic graphs, layout stability and on-line graph drawing are the main points of interest for the future research. In visual analysis, the understanding of the graph changes needs to be supported by stable layouts that preserve the mental map of the analyst. These layouts should be very stable for minor graph changes and, at the same time, be able to effectively show large graph changes. This issue is far from trivial, but it leads to easier spotting of structural changes in the graph and thereby to faster analytical results. On-line graph drawing, where the data stream is unpredictable, poses major challenges in this respect.
Compound graphs as a combined graph type, including aggregated graphs, represent a complex data type. The main analytical problem there is the understanding of both types of connections in a graph, as well as the understanding of the graph structures on multiple abstraction levels. This is a very cumbersome task, which can be supported by graph visualization systems. However, the drawing of such complex graphs is still in its infancy.
In the future, also further graph types such as hypergraphs [KKS09] may become more prominent in visual graph analysis research.

Graph uncertainty Graph visualization by now mainly deals with drawing graphs with given data, largely disregarding graph uncertainty. Visualization of uncertain data is a general challenge in visual analytics. As has been shown in [GS05], the degree of data certainty affects analytical decisions. Therefore, it is an important issue in visual graph analysis. In graph visualization, various types of uncertainty can be regarded. The uncertainty can relate to the graph structure (the existence of nodes and edges between them) and/or on graph attributes (edge and node attributes). For displaying node and edge attribute uncertainty, various methods from multivariate data visualization with uncertainty (see e.g., overviews given in [THM*05,PWL97,GS06]) could be applied. However, their suitability for graph needs to be studied. When dealing with structural uncertainty, it is expected that completely new methods will need to be developed.

Perception issues in graph visualization The understanding of graph structures in visualization strongly depends on human perception capabilities. Studies of human perception for graph drawing have recently focused on comparison of graph understanding using varying graph layouts. In graph design, studies on edge visualization have shown that the edge design has an influence on the graph reading. These various studies have given rise to new problems in graph visualization, which need to be studied in the future.

Taxonomies and benchmarks The field of visual graph analysis would profit from more elaborate taxonomies for tasks, interaction, visualization techniques, measures for quality, and benchmarks for comparing the new techniques. Although several taxonomies and sample data sets exists, a more broader scope of theory and data aspects is needed owing to the large set of problems in visual analysis of graphs.

Graph Interaction Techniques In graph exploration, recently new interaction techniques for various graph types have been developed. These techniques increasingly make use of the structural properties of the graph to interactively navigate in the graph (e.g. in [TS08b, vHP09, TAS09]). This tendency supports the analytical purpose of graph visualization, as analysts can more easily examine the structural relationship between entities in the graph. In the future, this direction can be extended.

Insight Provenance for Visual Graph Analysis In Visual Analytics applications, the analytical processes are often long-running. In order to support the reproducibility, reversibility and automation of these processes, user tracking of the graph interaction steps is necessary. As a basis for tracking, a taxonomy of graph interaction techniques is necessary. The theory of interaction is a general Visual Analytics challenge [TC06]. Although several interaction taxonomies also for insight provenance have been recently introduced [GZ08, HMSA08], their applicability and the need for their adaptation to graph analysis needs to be studied. In return, specific classifications of graph interaction techniques could be developed. In this report, we have aimed to classify them for gaining a concise overview of the current state of the research. This classification, however, may not be directly applicable to user tracking applications.

Visual Analysis Systems In line with Keim's visual analytics process [KAF*08], modern visual graph analysis systems should interactively integrate data pre-processing, interactive data visualization, building and visualizing of data models for gaining knowledge from the data. Many visual analysis techniques already include parts of this process. However, many of them rely on black box computations (e.g., automatic graph pre-processing, automatic calculation of graph similarities, of cliques). In order to support the variable hypothesis-insight-driven analytical process, more user involvement in the process should be aimed at. The user should have full control of the type of the analysis and its parameters. As this process includes multiple loops, interactive feedback possibilities are necessary. Therefore, integrated visual analysis systems should include such features.

Visual Graph Comparison One complex analytical task is the examination of the similarities and differences between graphs. This task builds up on the examination of the structure of one graph as discussed above. Lately, several papers about visual graph comparison for both trees and general graphs have been published (see Section 5). The comparison can concern only two graphs, trying to match nodes and edges between them. It can focus on finding similar graphs for one particular graph from a large set of graphs. It can concern gaining an overview of the types of structures in a large set of graphs. It can concentrate on analyzing the similarities of whole graphs or on matching of parts of one graph to other graphs. Owing to its complexity, and the variety of the problems, it is foreseeable that the research in this area will continue.

Integration of various data types in visual analysis Graphs as data structures capturing relationships between entities are part of a larger set of data types examined in various applications. Usually, the analysis of graphs is undertaken in combination with analysis of related data sets, or other data sets are transformed into graphs for their analysis [CGK*07, BMGK08]. For analysis of the various data sets as a whole, the sole focus on visual graph analysis (in particular graph exploration) without taking other relevant data into account, is not suitable. In the future, larger integrated visual analytics systems combining research results from several areas are needed.

Collaborative visual graph analysis For solving complex analytical tasks concerning multiple large related data sets, a collaboration of several experts is necessary. Recently, the development of collaborative visual analysis systems has received attention [Kee06, Ise07, BMZ*06]. However, collaborative visual graph analysis is not represented prominently. Therefore, the study of collaborative systems including graph data sets would be of advantage. The specifics of graph exploration, in particular, need to be studied.

Applications For analytical purposes, standard graph visualization and analysis methods need to be adapted to the specific needs of the particular application domain. For example, there are specialized systems for visualization of biochemical structures, shareholding structures and many more. Designing graph visualization systems with fast adaptability to various data types, analytical tasks and application-dependent analytical processes is still a challenge. Even within one application, often, the network to be analyzed needs to be constructed from heterogeneous data sources, and the focus of interest (attributes of nodes and edges) varies dynamically. Designing such systems is obviously not trivial.

Acknowledgements

We thank Vyara Ivanova, Anna Mitkova, Melanie Goerner, and Robert Rehner for their useful comments and suggestions. We thank all authors and copyright holders of the original figures for agreeing to their reproduction in this paper. This work was partially supported by the German Federal Ministry of Economics and Technology within the THESEUS project (http://www.theseus-programm.de/). It was also partially supported by the German Research Foundation (DFG) within the project Visual Feature Space Analysis as part of the Priority Program on Scalable Visual Analytics (SPP 1335).

References

[AAM07] Archambault D., Auber D., Munzner T.: Topolayout: Multilevel graph layout by topological features. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (2007), 305–317. 7

[ACJM03] Auber D., Chiricota Y., Jourdan F., Melancon G.: Multiscale visualization of small world networks. *Proceedings of IEEE Symposium on Information Visualization* (2003), 75–81. 3

[ADWM04] Adai A. T., Date S. V., Wieland S., Marcotte E. M.: LGL: creating a map of protein function with an algorithm for visualizing very large biological networks. *Journal of Molecular Biology 340*, 1 (June 2004), 179–190. 7

[AF07] Appert C., Fekete J.-D.: Naviguer dans des grands arbres avec controltree. In *Proceedings of International Conference of the Association Francophone d'Interaction Homme-Machine* (2007), pp. 139–142. 12

[AH98] Andrews K., Heidegger H.: Information slices: Visualising and exploring large hierarchies using cascading, semicircular discs. In *Late Breaking Hot Topic Paper, Proceedings of IEEE Symposium on Information Visualization* (1998). 6

[AK07] Andrews K., Kasanicka J.: A comparative study of four hierarchy browsers using the hierarchical visualisation testing environment (HVTE). In *Proceedings of International Conference Information Visualization* (2007), pp. 81–86. 5

[AMA08] Archambault D., Munzner T., Auber D.: Grouseflocks: Steerable exploration of graph hierarchy space. *IEEE Transactions on Visualization and Computer Graphics 14*, 4 (July/August 2008), 900–913. 9, 13, 14, 16

[AMA09] Archambault D., Munzner T., Auber D.: Tuggraph: Path-preserving hierarchies for browsing proximity and paths in graphs. In *Proceedings of IEEE Pacific Visualization Symposium* (April 2009), pp. 113–120. 9, 13, 16

[Arc09] Archambault D.: Structural differences between two graphs through hierarchies. In *Proceedings of Graphics Interface* (2009), pp. 87–94. 16, 18

[AvH04] Abello J., van Ham F.: Matrix zoom: A visual interface to semi-external graphs. In *Proceedings of IEEE Symposium on Information Visualization* (2004), pp. 183–190. 8, 14

[AWW09] Andrews K., Wohlfahrt M., Wurzinger G.: Visual graph comparison. In *Proceedings of International Conference on Information Visualisation* (2009), pp. 62–67. 16

[Bab02] Baburin D. E.: Some modifications of Sugiyama approach. In *Revised Papers from the 10th International Symposium on Graph Drawing* (2002), pp. 366–367. 7

[BBBL09] Bachmaier C., Brandenburg F. J., Brunner W., Lovász G.: Cyclic leveling of directed graphs. *Lecture Notes in Computer Science 5417* (2009), 348–359. 7

[BBD08] Burch M., Beck F., Diehl S.: Timeline trees: visualizing sequences of transactions in information hierarchies. In *Proceedings of the working conference on Advanced visual interfaces* (2008), pp. 75–82. 10, 11

[BBD09] Beck F., Burch M., Diehl S.: Towards an aesthetic dimensions framework for dynamic graph visualisations. In *Information Visualisation, 2009 13th International Conference* (July 2009), pp. 592–597. 4

[BBG*09] Blaas J., Botha C., Grundy E., Jones M., Laramee R., Post F.: Smooth graphs for visual exploration of higher-order state transitions. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 969–976. 8

[BD08] Burch M., Diehl S.: Timeradartrees: Visualizing dynamic compound digraphs. *Computer Graphics Forum 27*, 3 (2008), 823–830. 11, 12

[BDJ05] Brodlie K. W., Duke D. J., Joy K. I.: Arctrees: Visualizing relations in hierarchical data. In *Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization* (2005), Neumann P., Schlechtweg S., Carpendale S., (Eds.), pp. 53–60. 9, 10

[BDL05] Balzer M., Deussen O., Lewerentz C.: Voronoi treemaps for the visualization of software metrics. In *Proceedings of ACM symposium on Software visualization* (2005), pp. 165–172. 5

[Bed01] Bederson B. B.: Photomesa: a zoomable image browser using quantum treemaps and bubblemaps. In *Proceedings of ACM symposium on User interface software and technology* (2001), pp. 71–80. 5

[BHvW99] Bruls M., Huizing K., van Wijk J.: Squarified treemaps. In *Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization* (1999), pp. 33–42. 5

[BL09] Bertini E., Lalanne D.: Surveying the complementary role of automatic data analysis and visualization in knowledge discovery. In *Proceedings of the ACM SIGKDD Workshop on Visual Analytics and Knowledge Discovery* (2009), pp. 12–20. 12

[BMGK08] Barsky A., Munzner T., Gardy J., Kincaid R.: Cerebral: Visualizing multiple experimental conditions on a graph with biological context. *Visualization and Computer Graphics, IEEE Transactions on 14*, 6 (Nov.-Dec. 2008), 1253–1260. 19

[BMZ*06] Brennan S., Mueller K., Zelinsky G., Ramakrishnan I., Warren D., Kaufman A.: Toward a multi-analyst, collaborative framework for visual analytics. In *Proceedings of IEEE Symposium on Visual Analytics Science and Technology* (31 2006-Nov. 2 2006), pp. 129–136. 19

[BN01] Barlow T., Neville P.: A comparison of 2-D visualizations of hierarchies. In *Proceedings of IEEE Symposium on Information Visualization* (2001), pp. 131–138. 5

[BSW02] Bederson B. B., Shneiderman B., Wattenberg M.: Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies. *ACM Transactions on Graphics 21*, 4 (October 2002), 833–854. 5

[CC07] Collins C., Carpendale S.: VisLink: Revealing relationships amongst visualizations. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1192–1199. 16, 17

[CGK*07] Chang R., Ghoniem M., Kosara R., Ribarsky W., Yang J., Suma E., Ziemkiewicz C., Kern D., Sudjianto A.: WireVis: Visualization of categorical, time-varying data from financial transactions. *Proceedings of IEEE Symposium on Visual Analytics Science and Technology* (2007), 155–162. 19

[CMS99] Card S. C., Mackinlay J., Shneiderman B.: *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers, 1999. 11, 12

[CN02] Card S. K., Nation D.: Degree-of-interest trees: a component of an attention-reactive user interface. In *AVI '02: Proceedings of the Working Conference on Advanced Visual Interfaces* (New York, NY, USA, 2002), ACM, pp. 231–245. 13, 14

[CZQ*08] Cui W., Zhou H., Qu H., Wong P. C., Li X.: Geometry-based edge clustering for graph visualization. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1277–1284. 8

[DBETT99] Di Battista G., Eades P., Tamassia R., Tollis I. G.: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999. 7

[DGK01] Diehl S., Goerg C., Kerren A.: Preserving the mental map using foresighted layout. In *Proceedings of Joint Eurographics, IEEE TCVG Symposium on Visualization* (2001), pp. 175–184. 10

[DHKS05] Dayal U., Hao M., Keim D., Schreck T.: Importance driven visualization layouts for large time-series data. In *Proceedings of IEEE Symposium on Information Visualization* (2005), pp. 203–210. 10, 11

[Die05] Diestel R.: *Graph Theory*. Springer-Verlag, Heidelberg, 2005. 2, 3

[DK05] Dwyer T., Koren Y.: DIG-COLA: Directed graph layout through constrained energy minimization. In *Proceedings of IEEE Symposium on Information Visualization* (2005), p. 9. 7

[DLF*09] Dwyer T., Lee B., Fisher D., Quinn K. I., Isenberg P., Robertson G., North C.: A comparison of user-generated and automatic graph layouts. In *IEEE Symposium on Information Visualization* (2009). 8

[DMS*08] Dwyer T., Marriott K., Schreiber F., Stuckey P., Woodward M., Wybrow M.: Exploration of networks using overview+detail with constraint-based cooperative layout. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (Nov.-Dec. 2008), 1293–1300. 7

[DMW09a] Dwyer T., Marriott K., Wybrow M.: Dunnart: A constraint-based network diagram authoring tool. *Graph Drawing 5417* (2009), 420–431. 7, 15

[DMW09b] Dwyer T., Marriott K., Wybrow M.: Topology preserving constrained graph layout. In *Revised Papers from International Symposium on Graph Drawing* (2009), pp. 230–241. 7

[Dog02] Dogrusoz U.: Two-dimensional packing algorithms for layout of disconnected graphs. *Information Sciencies 143*, 1-4 (2002), 147–158. 8

[DPS02] Díaz J., Petit J., Serna M.: A survey of graph layout problems. *ACM Computing Surveys 34*, 3 (2002), 313–356. 1, 2, 7

[EDG*08] Elmqvist N., Do T.-N., Goodell H., Henry N., Fekete J.-D.: Zame: Interactive large-scale graph visualization. In *Proceedings of IEEE Pacific Visualization Symposium* (2008), pp. 215–222. 4, 8, 9, 14

[EF09] Elmqvist N., Fekete J.-D.: Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics 99* (2009). 12, 13

[ERHF09] Elmqvist N., Riche Y., Henry N., Fekete J.-D.: Mélange: Space folding for visual exploration. *IEEE Transactions on Visualization and Computer Graphics 99*, 1 (2009). 14

[FDK02] Freivalds K., Dogrusöz U., Kikusts P.: Disconnected graph layout and the polyomino packing approach. In *Revised Papers from the 9th Int. Symposium on Graph Drawing* (London, UK, 2002), Springer-Verlag, pp. 378–391. 8

[FHK*09] Fung D. C. Y., Hong S.-H., Koschutzki D., Schreiber F., Xu K.: Visual analysis of overlapping biological networks. In *Proceedings of the 2009 13th International Conference Information Visualisation* (Washington, DC, USA, 2009), IEEE Computer Society, pp. 337–342. 16, 18

[FLM95] Frick A., Ludwig A., Mehldau H.: A fast adaptive layout algorithm for undirected graphs. In *Proceedings of the DIMACS International Workshop on Graph Drawing* (1995), pp. 388–403. 7

[FR91] Fruchterman T. M. J., Reingold E. M.: Graph drawing by force-directed placement. *Softw. Pract. Exper. 21*, 11 (1991), 1129–1164. 7

[Fre79] Freeman L. C.: Centrality in social networks. *Social Networks 1*, 3 (1979), 215–239. 16

[FT04] Frishman Y., Tal A.: Dynamic drawing of clustered graphs. In *Proceedings of the IEEE Symposium on Information Visualization* (2004), pp. 191–198. 11

[FT07] Frishman Y., Tal A.: Multi-level graph layout on the GPU. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1310–1319. 7

[FT08] Frishman Y., Tal A.: Online dynamic graph drawing. *IEEE Transactions on Visualization and Computer Graphics 14*, 4 (2008), 727–740. 10, 11

[Fur86] Furnas G. W.: Generalized fisheye views. *SIGCHI Bull. 17*, 4 (1986), 16–23. 12, 13

[FWD*03] Fekete J.-D., Wang D., Dang N., Aris A., Plaisant C.: Overlaying graph links on treemaps. In *Proceedings of IEEE Information Visualization Symposium Posters Compendium* (2003). 9

[GBD09] Greilich M., Burch M., Diehl S.: Visualizing the evolution of compound digraphs with TimeArcTrees. *Computer Graphics Forum 28*, 3 (2009), 975–982. 11, 12, 16

[GBPD04] Görg C., Birke P., Pohl M., Diehl S.: Dynamic graph drawing of sequences of orthogonal and hierarchical graphs. In *Graph Drawing* (2004), Pach J., (Ed.), vol. 3383 of *Lecture Notes in Computer Science*, Springer, pp. 228–238. 10

[GDLP09] Giacomo E., Didimo W., Liotta G., Palladino P.: Visual analysis of one-to-many matched graphs. In *Revised Papers from International Symposium on Graph Drawing* (2009), pp. 133–144. 16, 17

[GF01] Ghoniem M., Fekete J.-D.: Animating treemaps. In *Proceedings of 18th HCIL Symposium - Workshop on Treemap Implementations and Applications* (2001). 10

[GFC04] Ghoniem M., Fekete J.-D., Castagliola P.: A comparison of the readability of graphs using node-link and matrix-based representations. In *Proceedings of IEEE Symposium on Information Visualization* (2004), pp. 17–24. 6

[GH09] Gansner E. R., Hu Y.: Efficient node overlap removal using a proximity stress model. *Graph Drawing 5417* (2009), 206–217. 8

[GK01] GAJER P., KOBOUROV S. G.: GRIP: Graph drawing with intelligent placement. In *Proceedings of the 8th International Symposium on Graph Drawing* (2001), pp. 222–228. 7, 8

[GKS07] GOEHLSDORF D., KAUFMANN M., SIEBENHALLER M.: Placing connected components of disconnected graphs. In *Proceedings of Asia-Pacific Symposium on Information Visualisation* (Februar 2007), pp. 101–108. 8

[GS05] GRIETHE H., SCHUMANN H.: Visualizing uncertainty for improved decision making. In *Proceedings of 4th International Conference on Perspectives in Business Informatics Research (BIR 2005)* (2005). 18

[GS06] GRIETHE H., SCHUMANN H.: The visualization of uncertain data: Methods and problems. In *Proceedings of Simulation and Visualization* (2006), pp. 143–156. 18

[GZ08] GOTZ D., ZHOU M.: Characterizing users' visual analytic activity for insight provenance. In *Proceedings of IEEE Symposium on Visual Analytics Science and Technology* (2008), pp. 123–130. 19

[HB05] HEER J., BOYD D.: Vizster: visualizing online social networks. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on* (Oct. 2005), pp. 32–39. 13, 16

[HC04] HEER J., CARD S. K.: Doitrees revisited: scalable, space-constrained visualization of hierarchical data. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces* (New York, NY, USA, 2004), ACM, pp. 421–424. 13, 14

[HF06] HENRY N., FEKETE J.-D.: MatrixExplorer: a dual-representation system to explore social networks. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (Sept.-Oct. 2006), 677–684. 8, 9, 13, 15

[HF07] HENRY N., FEKETE J.-D.: Matlink: Enhanced matrix visualization for analyzing social networks. In *Proceedings of the International Conference on Human-Computer-Interaction – Interact* (2007), pp. 288–302. 9, 10, 16

[HFM07] HENRY N., FEKETE J.-D., MCGUFFIN M.: NodeTrix: a hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (Nov.-Dec. 2007), 1302–1309. 6, 9, 10, 15, 16

[HJ05] HACHUL S., JÜNGE M.: Drawing large graphs with a potential-field-based multilevel algorithm. *Lecture notes in Computer Science 4372* (2005), 285–295. 7

[HJ07] HACHUL S., JÜNGER M.: Large-graph layout algorithms at work: An experimental study. *Journal of Graph Algorithms and Applications 11*, 2 (2007), 234–369. 7

[HK02] HAREL D., KOREN Y.: Graph drawing by high-dimensional embedding. In *Revised Papers from the 10th International Symposium on Graph Drawing* (2002), pp. 207–219. 7

[HMM00] HERMAN I., MELANCON G., MARSHALL M. S.: Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics 6*, 1 (/2000), 24–43. 1, 2, 3, 6

[HMSA08] HEER J., MACKINLAY J., STOLTE C., AGRAWALA M.: Graphical histories for visualization: Supporting analysis, communication, and evaluation. *Visualization and Computer Graphics, IEEE Transactions on 14*, 6 (Nov.-Dec. 2008), 1189 –1196. 19

[Hol06] HOLTEN D.: Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 741–748. 8, 9, 10

[Hua07] HUANG W.: Using eye tracking to investigate graph layout effects. In *Proceedings of Asia-Pacific Symposium on Visualization* (2007), pp. 97–100. 7

[HvW08] HOLTEN D., VAN WIJK J. J.: Visual comparison of hierarchically organized data. *Computer Graphics Forum 27*, 3 (2008), 759–766. 16, 17

[HvW09] HOLTEN D., VAN WIJK J. J.: A user study on visualizing directed edges in graphs. In *Proceedings of international conference on Human factors in computing systems* (2009), pp. 2299–2308. 8

[IAG*09] IMAMICHI T., ARAHORI Y., GIM J., HONG S.-H., NAGAMOCHI H.: Removing node overlaps using multi-sphere scheme. In *Graph Drawing* (2009), pp. 296–301. 8

[Ise07] ISENBERG P.: Information visualization in co-located collaborative environments. In *Proceedings of the Grace Hopper Celebration of Women in Computing, PhD Forum* (2007). 19

[JHGH08] JIA Y., HOBEROCK J., GARLAND M., HART J.: On the visualization of social and other scale-free networks. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1285–1292. 3, 4

[KAF*08] KEIM D., ANDRIENKO G., FEKETE J.-D., GÖRG C., KOHLHAMMER J., MELANCON G.: *Information Visualization*, vol. 4950 of *Lecture Notes in Computer Science*. Springer, 2008, ch. Visual Analytics: Definition, Process, and Challenges, pp. 154–175. 2, 19

[KCH02] KOREN Y., CARMEL L., HAREL D.: ACE: A fast multiscale eigenvectors computation for drawing huge graphs. In *Proceedings of the IEEE Symposium on Information Visualization* (2002), p. 137. 7

[Kee06] KEEL P.: Collaborative visual analytics: Inferring from the spatial organization and collaborative use of information. In *Proceedings of IEEE Symposium on Visual Analytics Science and Technology* (31 2006-Nov. 2 2006), pp. 137–144. 19

[KG06] KUMAR G., GARLAND M.: Visual exploration of complex time-varying graphs. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (Sept.-Oct. 2006), 805–812. 10, 11, 12

[KHG03] KOSARA R., HAUSER H., GRESH D. L.: An interaction view on information visualization. In *State-of-the-Art Proceedings of EUROGRAPHICS* (2003), pp. 123–137. 11

[KK89] KAMADA T., KAWAI S.: An algorithm for drawing general undirected graphs. *Information Processing Letters 31*, 1 (1989), 7–15. 7

[KKS09] KAUFMANN M., KREVELD M., SPECKMANN B.: Subdivision drawings of hypergraphs. In *Revised Papers from International Symposium on Graph Drawing* (2009), pp. 396–407. 18

[KMS*08] KEIM D. A., MANSMANN F., SCHNEIDEWIND J., THOMAS J., ZIEGLER H.: Visual analytics: Scope and challenges, visual data mining: Theory, techniques and tools for visual analytics. *Lecture Notes In Computer Science* (LNCS*) 4404* (2008), 76–90. 2

[Kob04] KOBSA A.: User experiments with tree visualization systems. In *Proceedings of the IEEE Symposium on Information Visualization* (2004), pp. 9–16. 5

[Koh01] KOHONEN T.: *Self-Organizing Maps*, 3rd ed. Springer, Berlin, 2001. 7

[Kru56] KRUSKAL J. B.: On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society 7*, 1 (1956), 48–50. 3

[KSS06] KLUKAS C., SCHREIBER F., SCHWÖBBERMEYER H.: Coordinated perspectives and enhanced force-directed layout for the analysis of network motifs. In *Proceedings of Asia-Pacific Symposium on Information Visualisation* (2006), pp. 39–48. 7

[LF06] LESKOVEC J., FALOUTSOS C.: Sampling from large graphs. In *Proceedings of 12th ACM SIGKDD Int. Conference on Knowledge Discovery and data mining* (2006), pp. 631–636. 4

[LPS*06] LEE B., PLAISANT C., SIMS PARR C., FEKETE J.-D., HENRY N.: Task taxonomy for graph visualization. In *Proceedings of AVI workshop on BEyond time and errors* (2006), pp. 1–5. 1

[MCH*09] MOSCOVICH T., CHEVALIER F., HENRY N., PIETRIGA E., FEKETE J.-D.: Topology-aware navigation in large networks. In *Proceedings of international conference on Human factors in computing systems* (2009), pp. 2319–2328. 14

[Mey98] MEYER B.: Competitive learning of network diagram layout. In *Proceedings of IEEE Symposium on Visual Languages* (September 1998), pp. 56–63. 7

[MGL06] MUELLER C., GREGOR D., LUMSDAINE A.: Distributed force-directed graph layout and visualization. In *Eurographics Symposium on Parallel Graphics and Visualization* (2006). 17

[MGT*03] MUNZNER T., GUIMBRETIÈRE F., TASIRAN S., ZHANG L., ZHOU Y.: TreeJuxtaposer: scalable tree comparison using focus+context with guaranteed visibility. In *In Proceedings of ACM SIGGRAPH* (2003), pp. 453–462. 14, 16, 17

[MJ09] MCGUFFIN M. J., JURISICA I.: Interaction techniques for selecting and manipulating subgraphs in network visualizations. *IEEE Transactions on Visualization and Computer Graphics 15* (2009), 937–944. 13, 14

[MJW*09] MA'AYAN A., JENKINS S., WEBB R., BERGER S., PURUSHOTHAMAN S., HUSN N. A., POSNER J., FLORES T., IYENGAR R.: SNAVI: Desktop application for analysis and visualization of large-scale signaling networks. *BMC Systems Biology 3*, 1 (2009). 3, 16

[MM08] MUELDER C., MA K.-L.: Rapid graph layout using space filling curves. *Visualization and Computer Graphics, IEEE Transactions on 14*, 6 (Nov.-Dec. 2008), 1301–1308. 7

[MMO05] MCPHERSON J., MA K.-L., OGAWA M.: Discovering parametric clusters in social small-world graphs. In *Proceedings of the 2005 ACM symposium on Applied computing* (2005), pp. 1231–1238. 16

[Mun97] MUNZNER T.: H3: laying out large directed graphs in 3D hyperbolic space. In *Proceedings of IEEE Symposium on Information Visualization* (1997), pp. 2–10, 114. 6

[NCA06] NEUMANN P., CARPENDALE M. S. T., AGARAWALA A.: Phyllotrees: Phyllotactic patterns for tree layout. In *Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization* (2006), Eurographics, pp. 59–66. 6

[OCNF09] OTJACQUES B., CORNIL M., NOIRHOMME M., FELTZ F.: CGD — a new algorithm to optimize space occupation in ellimaps. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction* (2009), pp. 805–818. 5

[OPPROG09] OVALLE-PERANDONES M. A., PERIANES-RODRIGUEZ A., OLMEDA-GOMEZ C.: Hubs and authorities in a spanish co-authorship network. In *Proceedings of International Conference Information Visualisation* (2009), pp. 514–518. 16

[PD08] POHL M., DIEHL S.: What dynamic network metrics can tell us about developer roles. In *Proceedings of the international workshop on Cooperative and human aspects of software engineering* (New York, NY, USA, 2008), ACM, pp. 81–84. 16

[PRB08] POHL M., REITZ F., BIRKE P.: As time goes by: integrated visualization and analysis of dynamic networks. In *Proceedings of the working conference on Advanced visual interfaces* (2008), pp. 372–375. 16

[PSD09] POHL M., SCHMITT M., DIEHL S.: Comparing readability of graph layouts using eyetracking and task-oriented analysis. In *Proceedings of Computer Graphics International* (2009). 7

[PWL97] PANG A., WITTENBRINK C., LODHA. S.: Approaches to uncertainty visualization. *The Visual Computer 13*, 8 (Nov. 1997), 370–390. 18

[RMC91] ROBERTSON G. G., MACKINLAY J. D., CARD S. K.: Cone Trees: animated 3D visualizations of hierarchical information. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (1991), pp. 189–194. 6

[RPD09] REITZ F., POHL M., DIEHL S.: Focused animation of dynamic compound graphs. In *Proceedings of International Conference Information Visualisation* (2009), pp. 679–684. 11

[SA06] SHNEIDERMAN B., ARIS A.: Network visualization by semantic substrates. *Visualization and Computer Graphics, IEEE Transactions on 12*, 5 (Sept.-Oct. 2006), 733–740. 7, 8

[SB92] SARKAR M., BROWN M. H.: Graphical fisheye views of graphs. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (1992), pp. 83 – 91. 14

[Sch08] SCHWÖBBERMEYER H.: *Analysis of Biological Networks*. Wiley Series on Bioinformatics, Computational Techniques and Engineering. Wiley, 2008, ch. 5, pp. 85 – 112. 16

[Shn92] SHNEIDERMAN B.: Tree visualization with tree-maps: 2-D space-filling approach. *ACM Transactions on Graphics 11*, 1 (1992), 92–99. 5

[SKM06] SCHRECK T., KEIM D. A., MANSMANN S.: Regular treemap layouts for visual analysis of hierarchical data. In *Proceedings of Spring Conference on Computer Graphics* (2006). 10

[SKW*07] SCHEDL M., KNEES P., WIDMER G., SEYERLEHNER K., POHLE T.: Browsing the web using stacked three-dimensional sunbursts to visualize term co-occurrences and multimedia content. In *Poster Compendium of Visualization Conference* (2007). 5

[SLN05] SARAIYA P., LEE P., NORTH C.: Visualization of graphs with associated timeseries data. In *Proceedings of IEEE Symposium on Information Visualization* (Oct. 2005), pp. 225–232. 10, 11

[SLS07] SCHULZ H.-J., LUBOSCHIK M., SCHUMANN H.: Exploration of the 3D treemap design space. In *Poster Compendium of IEEE Information Visualization Conference* (2007). 5

[SS05] SCHREIBER F., SCHWÖBBERMEYER H.: MAVisto: a tool for the exploration of network motifs. *Bioinformatics 21*, 17 (July 2005), 3572–3574. 16

[SSH09] SCHULZ H.-J., SCHUMANN H., HADLAK S.: Point-based tree representation - a new approach for large hierarchies. In *Proceedings of IEEE Pacific Visualization Symposium* (2009), pp. 81–88. 6

[Sta00] STASKO J.: An evaluation of space-filling information visualizations for depicting hierarchical structures. *International Journal of Human-Computer Studies 53*, 5 (2000), 663–694. 5

[STT81] SUGIYAMA K., TAGAWA S., TODA M.: Methods for visual understanding of hierarchical system structures. *Systems, Man and Cybernetics, IEEE Transactions on 11*, 2 (Feb. 1981), 109–125. 7

[SZ00] STASKO J., ZHANG E.: Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Proceedings of the IEEE Symposium on Information Vizualization* (2000), p. 57. 5

[SZG*96] SCHAFFER D., ZUO Z., GREENBERG S., BARTRAM L., DILL J., DUBS S., ROSEMAN M.: Navigating hierarchically clustered networks through fisheye and full-zoom methods. *ACM TOCHI 3*, 2 (1996), 162–188. 14

[TAS09] TOMINSKI C., ABELLO J., SCHUMANN H.: CGV–an interactive graph visualization system. *Computers & Graphics 33*, 6 (2009), 660 – 678. 19

[TAvHS06] TOMINSKI C., ABELLO J., VAN HAM F., SCHUMANN H.: Fisheye tree views and lenses for graph visualization. In *Proceedings of IEEE Information Visualization* (2006), pp. 17–24. 14, 15

[TC05] THOMAS J. J., COOK K. A.: *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Center, 2005. 2

[TC06] THOMAS J., COOK K.: A visual analytics agenda. *Computer Graphics and Applications, IEEE 26*, 1 (Jan.-Feb. 2006), 10– 13. 19

[THM*05] THOMSON J., HETZLER B., MACEACHREN A., GAHEGAN M., PAVEL M.: A typology for visualizing uncertainty. In *Proceedings of SPIE Conference on Visualization and Data Analysis 2005* (2005), vol. 5669, pp. 146–157. 18

[TK08] TEKUŠOVÁ T., KOHLHAMMER J.: Visual analysis and exploration of complex corporate shareholder networks. In *Proceedings of the IS&T/SPIE Visualization and Data Analysis* (jan 2008), vol. 6809, p. 68090F. 8, 16

[TS99] TOYODA M., SHIBAYAMA E.: Hyper mochi sheet: a predictive focusing interface for navigating and editing nested networks through a multi-focus distortion-oriented view. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (1999), pp. 504–511. 14

[TS07] TU Y., SHEN H.-W.: Visualizing changes of hierarchical data using treemaps. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1286–1293. 5, 10

[TS08a] TEKUŠOVÁ T., SCHRECK T.: Visualizing time-dependent data in multivariate hierarchic plots - design and evaluation of an economic application. In *Proceedings of International Conference on Information Visualisation* (2008), pp. 143–150. 6, 10, 11

[TS08b] TU Y., SHEN H.-W.: Balloon focus: a seamless multi-focus+context method for treemaps. *Visualization and Computer Graphics, IEEE Transactions on 14*, 6 (Nov.-Dec. 2008), 1157–1164. 14, 15, 19

[vH03] VAN HAM F.: Using multilevel call matrices in large software projects. pp. 227 –232. 8

[vHP09] VAN HAM F., PERER A.: Search, show context, expand on demand: Supporting large graph exploration with degree-of-interest. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 953–960. 12, 13, 19

[vHR08] VAN HAM F., ROGOWITZ B.: Perceptual organization in user-generated graph layouts. *Visualization and Computer Graphics, IEEE Transactions on 14*, 6 (Nov.-Dec. 2008), 1333–1339. 8

[vHSD09] VAN HAM F., SCHULZ H.-J., DIMICCO J. M.: Honeycomb: Visual analysis of large scale social networks. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part II* (2009), pp. 429–442. 8

[vHvW02] VAN HAM F., VAN WIJK J.: Beamtrees: compact visualization of large hierarchies. In *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on* (2002), pp. 93–100. 5, 6

[vHW08] VAN HAM F., WATTENBERG M.: Centrality based visualization of small world graphs. *Computer Graphics Forum 27*, 3 (2008), 975–982. 3

[vLGRS09] VON LANDESBERGER T., GÖRNER M., REHNER R., SCHRECK T.: A system for interactive visual analysis of large graphs using motifs in graph editing and aggregation. In *Proceedings of Vision Modeling Visualization Workshop* (2009), pp. 331–339. 13, 16

[vLGS09] VON LANDESBERGER T., GÖRNER M., SCHRECK T.: Visual analysis of graphs with multiple connected components. In *Proceedings of IEEE Symposium on Visual Analytics Science and Technology* (2009), pp. 155–162. 16, 18

[vWvdW99] VAN WIJK J., VAN DE WETERING H.: Cushion treemaps: visualization of hierarchical information. In *Proceedings of IEEE Symposium on Information Visualization* (1999), pp. 73–78. 5, 6

[War00] WARE C.: *Information visualization: Perception for Design*. Morgan Kaufmann, 2000. 11

[Wat06] WATTENBERG M.: Visual exploration of multivariate graphs. In *Proceedings of the SIGCHI conference on Human Factors in computing systems* (2006), pp. 811–819. 16

[WCG03] WONG N., CARPENDALE S., GREENBERG S.: Edgelens: an interactive method for managing edge congestion in graphs. In *Proceedings of IEEE Symposium on Information Visualization* (2003), pp. 51–58. 14

[WD08] WOOD J., DYKES J.: Spatially ordered treemaps. *Visualization and Computer Graphics, IEEE Transactions on 14*, 6 (Nov.-Dec. 2008), 1348–1355. 5

[WMC*09] WONG P. C., MACKEY P., COOK K., ROHRER R., FOOTE H., WHITING M.: A multi-level middle-out cross-zooming approach for large graph analytics. In *Proceedings of IEEE Symposium on Visual Analytics Science and Technology* (Oct. 2009), pp. 147–154. 13

[WS03] WHITE S., SMYTH P.: Algorithms for estimating relative importance in networks. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2003), ACM, pp. 266–275. 16

[YKSJ07] YI J. S., KANG Y. A., STASKO J., JACKO J.: Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1224–1231. 12

[ZK08] ZIEMKIEWICZ C., KOSARA R.: The shaping of information by visual metaphors. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (Nov.-Dec. 2008), 1269–1276. 5

[ZMC05] ZHAO S., MCGUFFIN M., CHIGNELL M.: Elastic hierarchies: combining treemaps and node-link diagrams. In *Proceedings of IEEE Symposium on Information Visualization* (Oct. 2005), pp. 57–64. 5, 6, 15