FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Visualization Of Multivariate Networks

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Stephan Pajer
Matrikelnummer 0325816

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao. Univ.-Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Dipl.-Ing. Dr.techn. Harald Piringer

Wien, October 24, 2012

(Unterschrift Verfasserin) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Visualization Of Multivariate Networks

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Stephan Pajer
Registration Number 0325816

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao. Univ.-Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Assistance: Dipl.-Ing. Dr.techn. Harald Piringer

Vienna, October 24, 2012 _____ _____
(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Stephan Pajer
Björnsongasse 29/Haus 15, 1130 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasserin)

Abstract

Multivariate networks are graphs, which consist of a set of nodes and a set of connections between these nodes, that have additional data dimensions for each node and/or connection. Such multivariate graphs are prevalent in a number of different fields, including biological systems and social sciences.

In this thesis, an existing linked-view system for analyzing multivariate data has been extended for the analysis of networks. A node-link view has been implemented to give an overview of the graph. To leverage existing visualizations, additional data about the structure of the network can be added to the nodes and analyzed with views designed for unconnected multivariate data. Another contribution is a novel visualization that supports the study of queries concerning relationships between different groups of nodes in a network.

Kurzfassung

Multivariate Netzwerke sind Graphen, welche aus einer Menge von Knoten und einer Menge von Verbindungen zwischen diesen Knoten bestehen und zusätzliche Werte für jeden Knoten und/oder jede Verbindung zwischen Knoten enthalten. Solche multivariaten Netzwerke werden in unterschiedlichen Gebieten eingesetzt, unter anderem um biologische Systeme oder soziale Kontakte zu modellieren.

Diese Diplomarbeit basiert auf einem bestehenden System zur Analyse multivariater Daten, welches erweitert wurde um Graphen zu unterstützen. Die Auswahl an Ansichten wurde um ein Node-Link Diagramm erweitert, welches die Verbindungen zwischen Knoten darstellen kann. Bereits bestehende Ansichten können zur Analyse des Graphen verwendet werden, indem zusätzliche Werte über die Struktur des Graphen zu den Daten hinzugefügt werden. Um Suchen nach speziell zusammenhängenden unterschiedlichen Knoten zu ermöglichen, wurde eine neuartige Visualisierung implementiert.

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Goals

Many problems can be modeled in the form of a multivariate network. For human interactions, social networks model people and their relationships with each other. Such a model of society can, for example, be used to study how diseases spread across the population [OS04]. In terrorism prevention, social networks can be used to develop strategies for disrupting the formation of terror networks [PS06].Other areas where multivariate networks occur are, for example, in a biological context with applications like helping biologists study cellular reactions to various stimuli [BMGK08] or the analysis of state transition graphs [PV06].

This thesis deals with the visual analysis of multivariate networks. This chapter contains definitions of graph and information visualization terminology (Section 1.2), an overview of the main layouting approaches for graphs (Section 1.3) and describes the main tasks a system for the analysis of graphs needs to perform (Section 1.4). Finally, Section 1.5 gives an overview of the entire thesis.

1.2 Definitions

1.2.1 Definition of Graph Terminology

1.2.1.1 Graph

A graph G can be defined as a tuple containing a set N of nodes and a set E of edges:

$$G = (N, E)$$

Each edge E defines a binary relation on the set of nodes:

$$E \subseteq N \times N$$

Graphs can be either directed or undirected. The order of nodes in an edge does not matter in undirected graphs. In directed graphs, an edge $n_1 \times n_2$ is said to go from n_1 to n_2.

1.2.1.2 Adjacency

Two nodes are adjacent when an edge that connects them exists. An edge is incident to a node if the node is in the relation defined by the edge.

1.2.1.3 Node Degree

The degree of a node is defined as the number of edges that are connected to the node. For directed graphs, this can be split into *in*degree and *out*degree. Indegree describes the number of nodes a node can be directly reached from while outdegree shows how many nodes can be reached directly from a node.

1.2.1.4 Paths

A path consists of a sequence of nodes for which an edge connecting each pair of consecutive nodes is also in the graph. The length of a path is defined as the number of edges it utilizes, equaling its number of nodes minus one.

A loop is a path that starts and ends at the same node. A circle is a path in which the nodes are pairwise different with the exception of the start and end nodes.

1.2.1.5 Distance

The distance between two nodes is defined as the length of the shortest path that connects these nodes. If there is no path that connects two nodes their distance from each other is infinite. If two nodes are directly connected, they are adjacent.

1.2.1.6 Edge Weight

Edges in a weighted graph contain an additional weight value. This value is generally a real number, changing the definition of the edge data to:

$$E \subseteq N \times N \times \mathbb{R}$$

The weight of a path is the sum of the weights of the edges that connect the nodes in the path. In weighted graphs, the distance metric is in many cases modified and is given as the minimal sum of weights of a path that connects two nodes.

1.2.1.7 Betweenness Centrality

Betweenness centrality is a measure for the centrality of a node. Let g_{ij} be the number of paths with minimal distance between n_i and n_j and $g_{ij}(n_k)$ be the number of paths g_{ij} that pass through n_k for $i \neq j \neq k$, then the betweenness centrality C_B of n_k can be calculated as

$$C_B(n_k) = \sum_{i<j} \frac{g_{ij}(n_k)}{g_{ij}}$$

A useful variant of the betweenness centrality is the *relative* betweenness centrality C'_B, which is obtained by dividing by the maximal obtainable betweenness centrality of the graph:

$$C'_B(n_k) = \frac{2C_B(n_k)}{(|N|-1)(|N|-2)}$$

A maximal betweenness centrality means that a node is the central node of a star-formed graph, while a betweenness centrality of zero means that the removal of the node would not affect the shortest distance between any other two nodes.

1.2.1.8 Subgraph

A subgraph is a subset N_s of the nodes and a subset E_s of the edges of G. The connections supplied by the edges E_s may only cover the nodes in N_s.

1.2.1.9 Cliques

A complete graph is a graph which contains an edge from every node to every other node. If only a subgraph is completely connected, this subset of the graph is called a clique. Searching for cliques within a large graph is one of the basic tasks required in graph analysis.

1.2.1.10 Connected Components

A connected component is a subgraph in which there is a path from every node in the subgraph to every other node in the subgraph while no node has a connection to a node outside the subgraph. If a connected component can be split into multiple parts with the removal of a single node, this node is called an articulation point or cut point.

1.2.2 Special Graph Types

1.2.2.1 Planar Graph

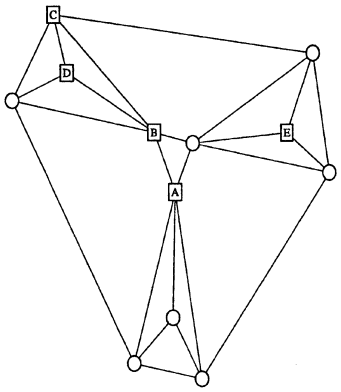


Figure 1.1: A planar drawing of a graph. (Image courtesy of McGrath et al. [MBK97]

Planar graphs (see Figure 1.1 for an example) are graphs that can be embedded in a two-dimensional drawing without any edge crossings. Graph planarity is important when designing electronic circuits, in which no connectors are allowed to cross.

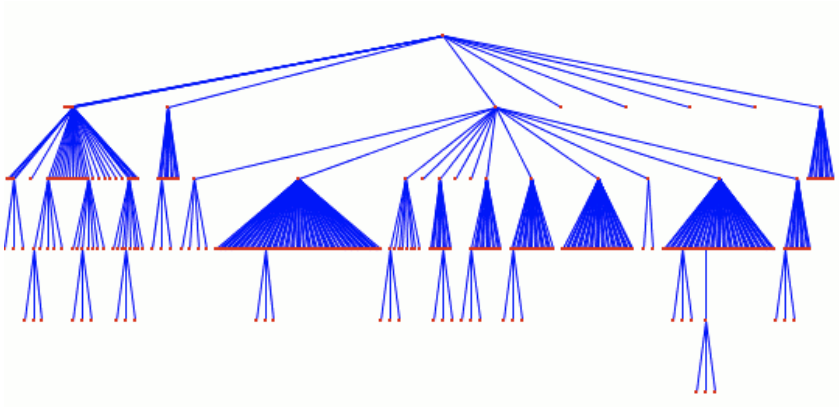


Figure 1.2: A tree. (Image courtesy of Herman et al. [HMM00]

1.2.2.2 Tree

A tree (see Figure 1.2 for an example) is a graph that consists of a single connected component without any circles. Any graph that is a tree is also planar.

1.2.2.3 Scale-Free Network

Scale-free networks (see Figure 1.3 for an example), described by Jia et al. [JHGH08], are graphs in which the part of nodes that has k connections follows a power law distribution:

$$P(k) \sim k^{-\gamma}$$

The maximal distance occurring in a scale-free network is generally very small compared to its size, as described by Watts [Wat04]. Scale-free networks occur in various fields and include power grids, the topology of web pages, social networks or biological systems.

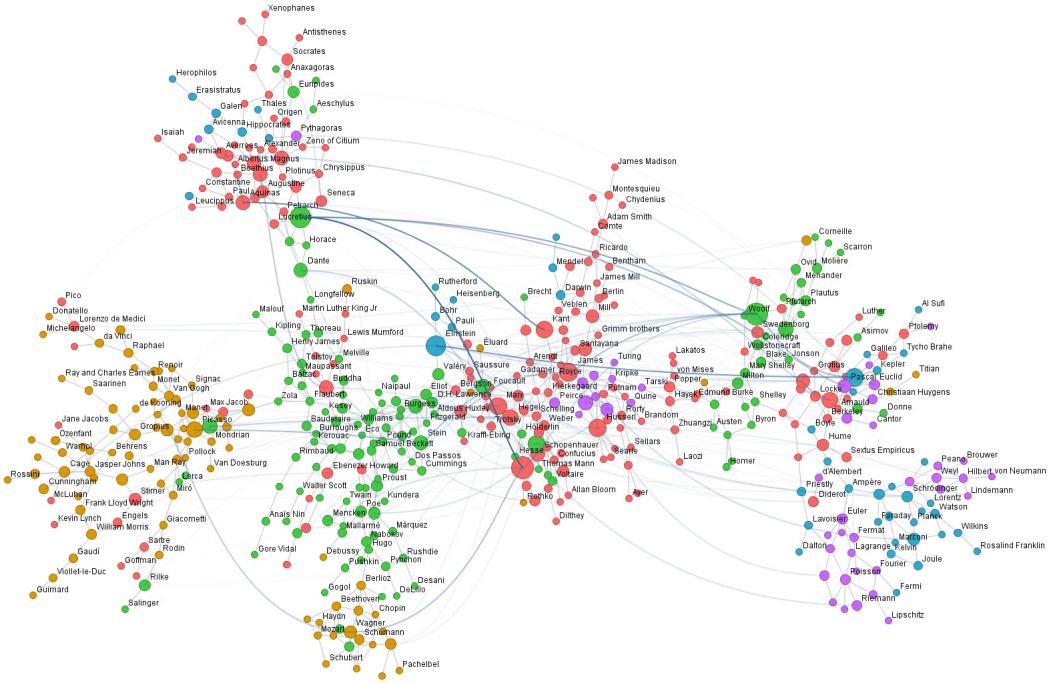


Figure 1.3: A scale-free network. (Image courtesy of Ham and Wattenberg [vHW08]

1.2.3 Visualization Types

1.2.3.1 Node-Link Diagram

Node-link diagrams show an embedding of the nodes as points in a plane. The edges are displayed as lines that connect the nodes. Figure 1.4 shows an example of a node-link diagram.

The readability of this visualization is highly dependent on the layout of the nodes in the plane. McGrath et al. [MBK97] have shown that the arrangement of nodes can also influence the judgment of those that view the image.

The following criteria which a good layout needs to optimize have been identified by Battista et al. [BETT94]:

- Distribute nodes uniformly
- Avoid edge crossings
- Reflect inherent symmetry
- Keep edge lengths uniform
- Avoid bends in edges

Numerous algorithms for node placement that strive for an informative and aesthetically pleasing layout have already been developed. An overview of the main approaches is given in Section 1.3.

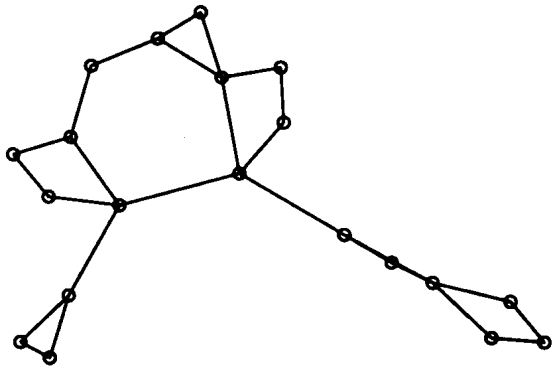


Figure 1.4: Network visualized using a node-link diagram. (Image courtesy of Fruchterman and Reingold [FR91])

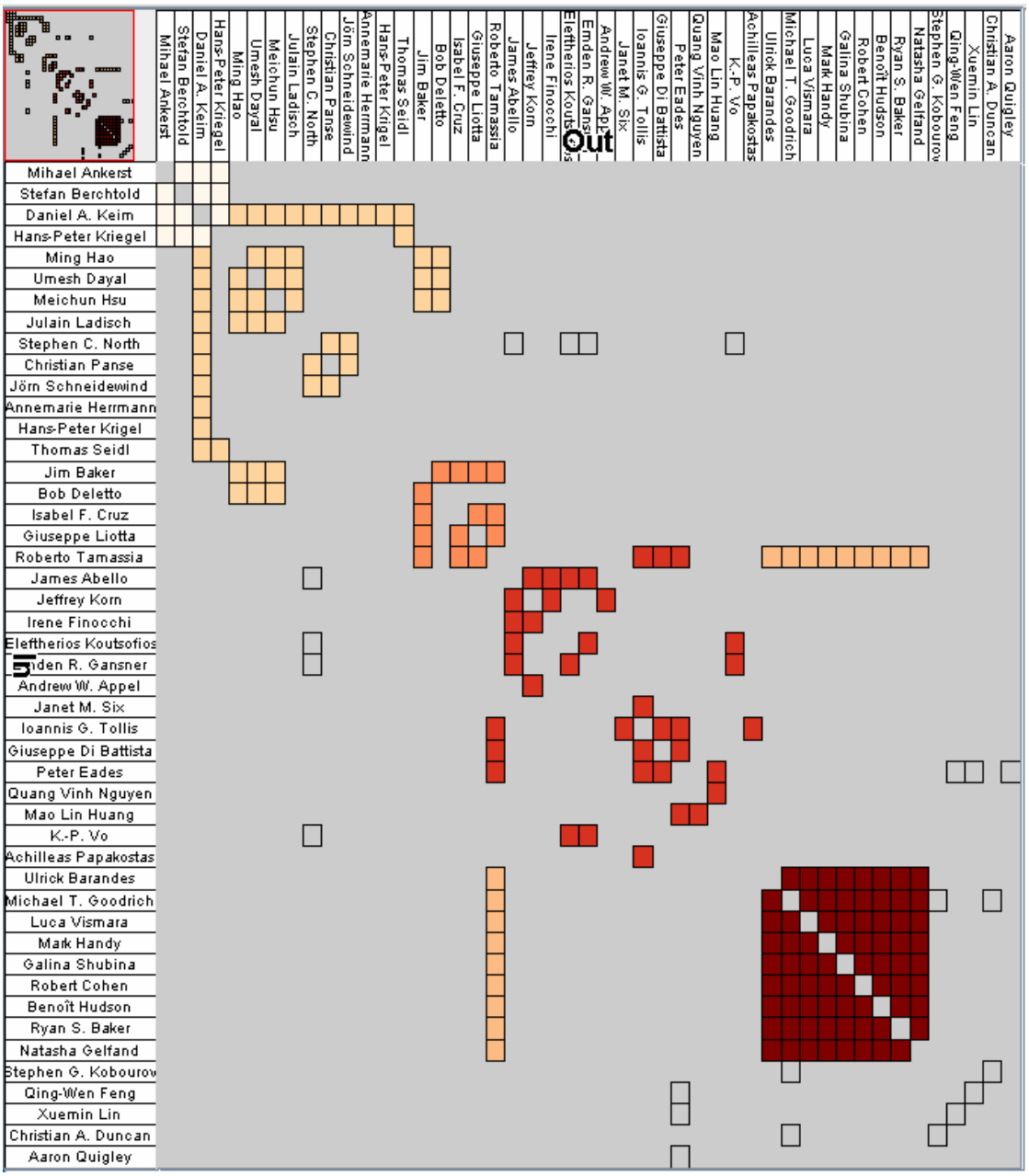


Figure 1.5: Network visualized using a matrix diagram. (Image courtesy of Henry and Fekete [HF06])

1.2.3.2 Matrix Diagram

A matrix diagram (see Figure 1.5) shows the adjacency matrix of the graph, in which both rows and columns represent the nodes and each entry M_{ij} shows if an edge from N_i to N_j exists in the graph. The adjacency matrix of an undirected graph is symmetric.

Compared to node-link diagrams, matrix diagrams scale better with the number of nodes N and edges E, since node-link views become cluttered when these values increase. On the other hand, it is easier to search for and follow paths that contain more than one edge with node-link diagrams, as shown by Ghoniem et al. [GFC05]. Furthermore, users in many fields are already used to seeing node-link diagrams, and are therefore less inclined to learn how to work with a different visualization if their performances are similar.

1.2.3.3 Arc Diagram

Arc diagrams use circular arcs to represent the connections between nodes placed in a one-dimensional layout. An example can be seen in Figure 1.6.

As with node-link diagrams, the readability of the resulting image is highly dependent on the layout of the nodes. Generally it es not quite as good as a node-link diagram could be due to the restricted placement of the nodes. The main advantage of this visualization is that it can be added to existing visualizations that already order and display data using a one-dimensional layout.

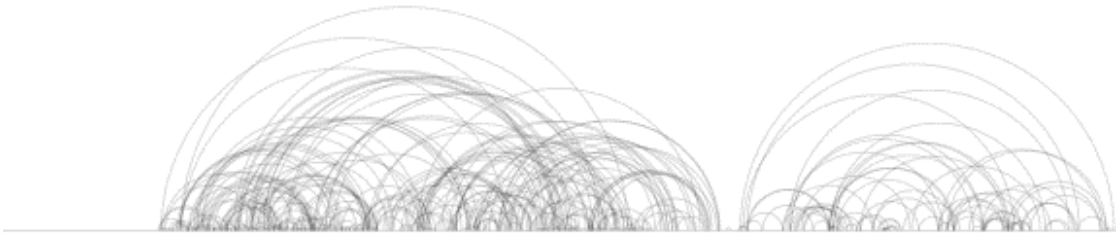


Figure 1.6: Network visualized using an arc diagram. (Image courtesy of Wattenberg [Wat02])

1.2.4 Definition of Information Visualization Terminology

1.2.4.1 Linked Views

Linked data views, as proposed by Shneiderman [Shn97], forgo utilizing a single monolithic view in favor of multiple smaller, connected views of the data. An example of this is shown in Figure 1.7. The views in such a system are generally less complex compared to monolithic systems, but their major advantage is that they can be linked together. If a user interacts with a single one of these views, all views are updated with the result of the interaction. This way, it is possible to combine the distinct advantages that different visualizations offer.

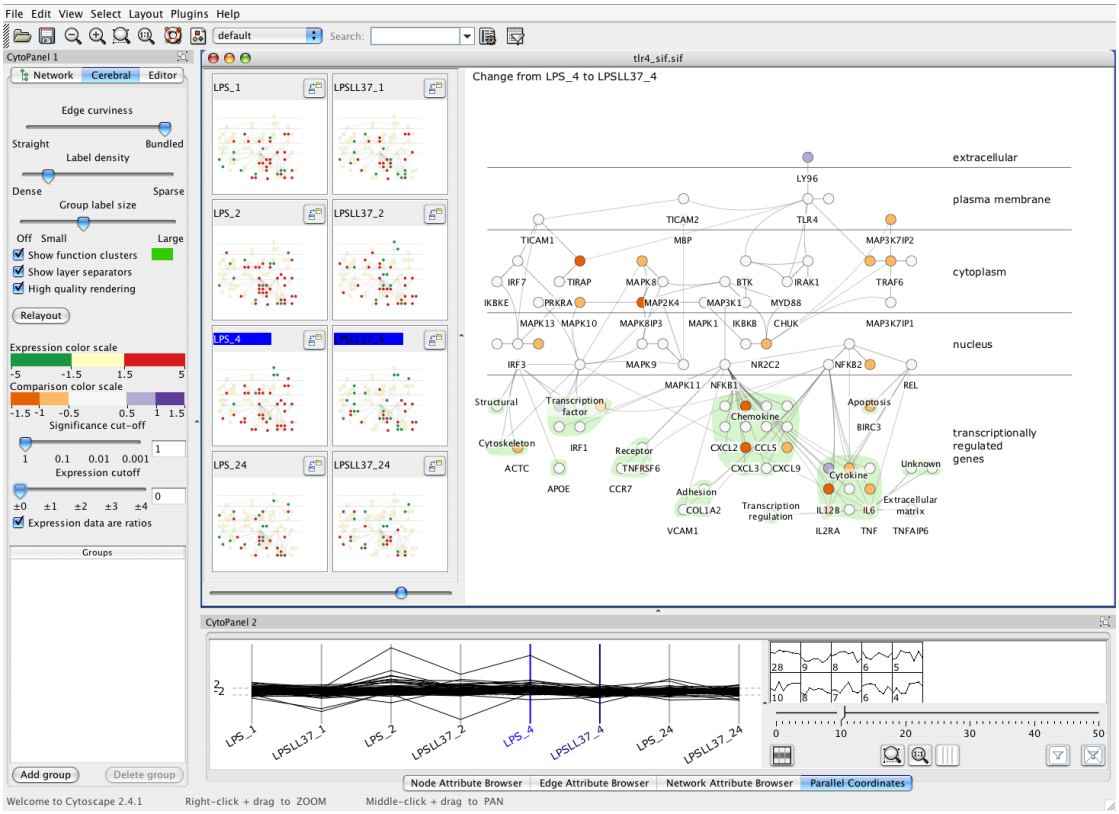


Figure 1.7: A linked view system comparing two time series of a single graph. An overview of all timesteps can be viewed alongside a parallel coordinate and a more detailed node-link view. (Image courtesy of Barsky et al. [BMGK08])

1.2.4.2 Linking & Brushing

Linking and brushing is a technique for using linked views. A single selection is accessed by all linked views, so by brushing entities in one view, they are selected in all linked views. This makes it easy to see correlations between views. An example of this is shown in Figure 1.8.

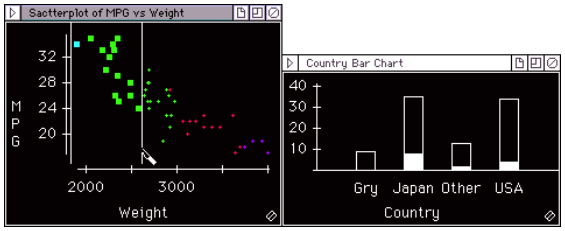
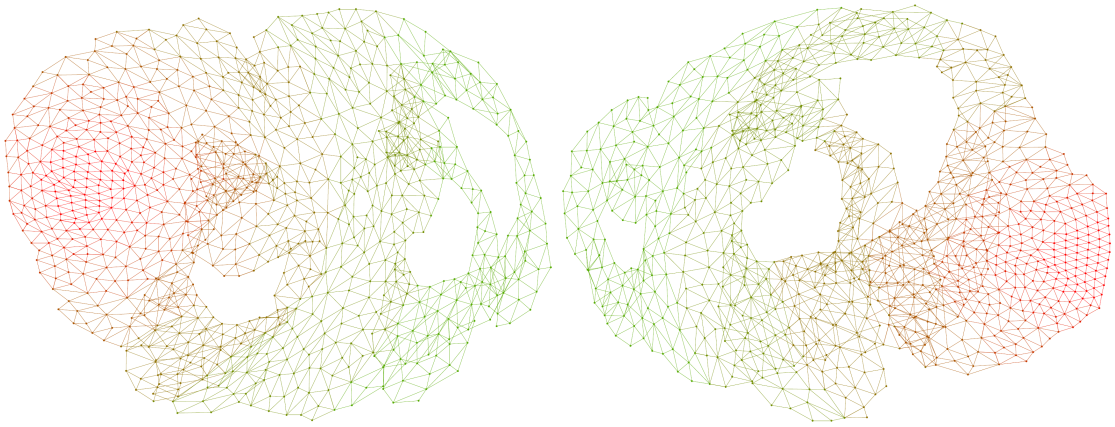


Figure 1.8: Using linking & brushing, the entries selected in a scatter plot are also highlighted in a linked bar chart. (Image courtesy of Wills [Wil99])

1.2.4.3 Focus & Context

An image that employs focus & context consists of multiple parts (see Figure 1.9). A user-specified part of the data that is of high interest is focused and shown in high resolution. The rest of the data is displayed in lower resolution to show the context in which the focused data is embedded. This can be accomplished by means like a fish-eye view, which transforms the coordinate system of the view to magnify the focus.



(a) Focusing on the left side.

(b) Focusing on the right side.

Figure 1.9: Effect of a fish-eye view on a graph. (Images courtesy of Gansner et al. [GKN04])

1.2.4.4 Overview & Detail

Overview & detail shares the goal of focus & context of providing a highly detailed image of an interesting part of the data while being able to show the complete data at once (see Figure 1.10). In contrast to focus & context, overview & detail accomplishes this by utilizing multiple linked views to provide separate views for overview and detail. The combination of these images is performed by the user mentally.

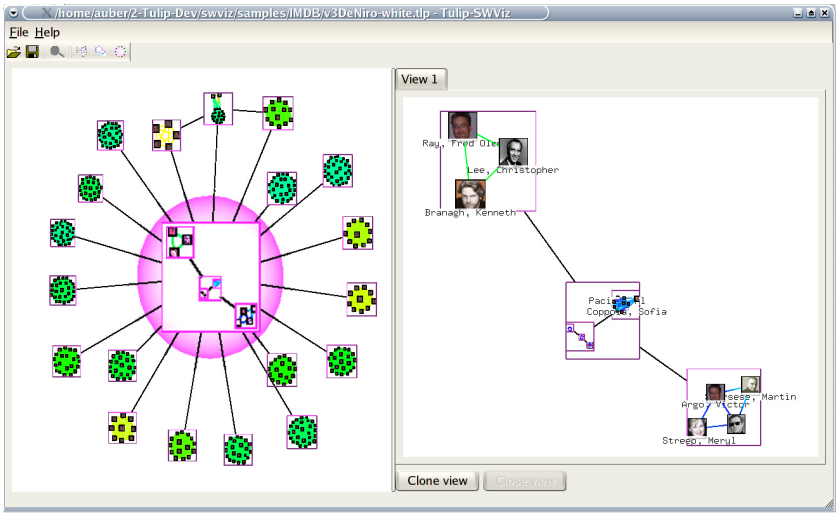


Figure 1.10: Two views, one showing an overview of a graph with the other showing the selected component in greater detail. (Image courtesy of Auber et al. [ACJM03])

1.2.4.5 Exploratory Data Analysis

Originally, the analysis of large data sets consisted of forming a hypothesis and utilizing statistical methods to either verify or falsify it. In contrast, exploratory data analysis, introduced by Tukey [Tuk77], does not need a hypothesis formulated in advance. Instead, it relies on the human visual system to discover structure in the visualized data and *create* a hypothesis from the resulting insights. This approach has the advantage that the resulting model generally fits the data better then one that has been created prior to the investigation of the data. In addition, exploratory data analysis can just as easily be used when little is known about the data at hand which would significantly reduce the efficiency of an analysis performed using hypothesis verification.

1.3 Layouting Algorithms for Node-Link Diagrams

This section gives an overview of the main approaches to place nodes in a node-link diagram. General aesthetic criteria that should be fulfilled by a layouting algorithm are listed in Section 1.2.3.1. Many refinements and modifications to these approaches have been proposed. This section only lists the most common basic algorithms.

1.3.1 Orthogonal Layout

Orthogonal layouts only allow horizontal and vertical edges which may contain bends (see Figure 1.11). Orthogonal layouts originally stem from other areas, such as the calculation of a layout for a printed circuit board. To create the layout the algorithm first planarizes the graph by either removing edges until the graph is planar or by replacing edge crossings with vertices. The algorithm then minimizes the bends and restores the modified edges. An example of an algorithm that produces an orthogonal layout is GIOTTO [TBB88].

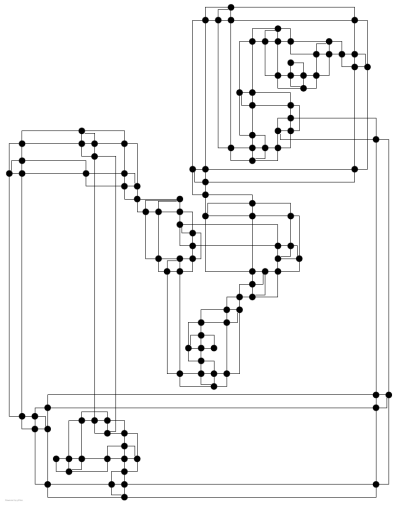


Figure 1.11: Graph drawn using an orthogonal layout. (Image courtesy of Dwyer et al. [DMS*08])

1.3.2 Radial Layout

Radial layouts only work on trees and place the nodes on concentric rings around a central node (see Figure 1.12). After choosing a root node, each subtree of the tree is placed in its own sector of a circle centered at the root node. The distance from the root node denotes the distance of the node from the center. Nodes with identical distance

from the center occupy non-overlapping sectors of the same ring. An example of an algorithm that produces a radial layout has been proposed by Eades [Ead92].

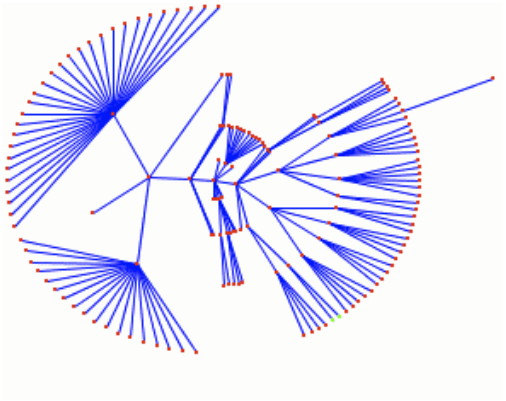


Figure 1.12: Graph drawn using a radial layout. (Image courtesy of Herman et al. [HMM00])

1.3.3 Circular Layout

Circular layouts position the nodes of the graph at distinct positions along the perimeter of a circle (see Figure 1.13). The main responsibility of the algorithm is ordering the nodes to minimize line crossings. An example of such an approach has been proposed by Baur and Brandes [BB04].

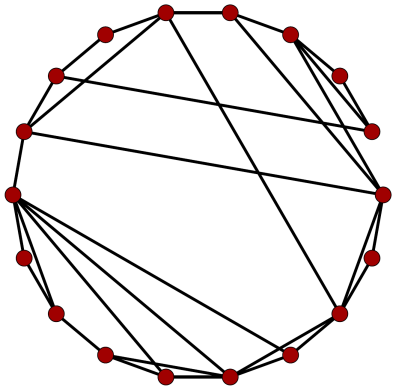


Figure 1.13: Graph drawn using a circular layout. (Image courtesy of Baur and Brandes [BB04])

1.3.4 Spectral Layout

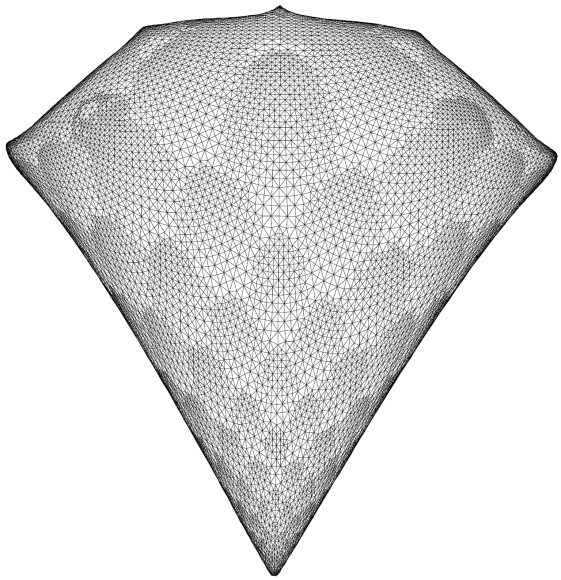


Figure 1.14: Graph drawn using a spectral layout. (Image courtesy of Koren [Kor02])

Spectral layouts use the eigenvectors of related matrices such as the adjacency matrix or the Laplacian to compute positions for the nodes (see Figure 1.14). An example of an algorithm that produces a spectral layout has been proposed by Koren [Kor02].

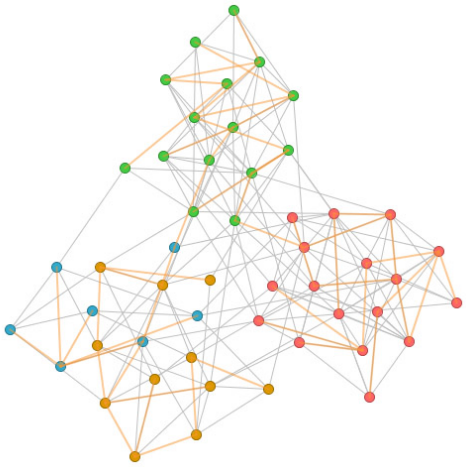


Figure 1.15: Graph drawn using a force-directed layout. (Image courtesy of Ham and Wattenberg [vHW08])

1.3.5 Force-Directed Layout

Force-directed layouting algorithms model the graph as a physical system in which nodes repulse each other and connected nodes additionally attract each other. The algorithm minimizes the energy function of the system after creating an initial layout. An example of an algorithm that produces a force-directed layout has been proposed by Fruchterman and Reingold [FR91] and an example of a graph layouted using a force-directed layout can be seen in Figure 1.15.

1.4 Task Taxonomy

1.4.1 Task Taxonomy for Information Visualization

A task taxonomy is a collection of common tasks that are performed to reach a given goal. Building a system based on a critiqued and refined collection of tasks ensures a more systematical sampling of the space of tasks, as described by Carroll [Car00].

The following task taxonomy for low-level tasks in information visualization, published by Amar et al. [AES05], will be referenced for the rest of this thesis.

Retrieve Value Retrieve the attribute of a specific entity.

Filter Find cases that satisfy a condition on their attributes.

Compute Derived Value Compute a new value derived from the existing data.

Find Extremum Find the case(s) possessing the extreme value of a specific attribute.

Sort Rank the data cases according to some ordinal metric.

Determine Range Find the span the values of a specific attribute lie in.

Characterize Distribution Characterize the distribution of a specific quantitative attribute.

Find Anomalies Find statistical outliers of a given set.

Cluster Find subsets containing similar attribute values.

Correlate Determine the relationship between the data values of two given attributes.

1.4.2 Task Taxonomy for Graph Analysis

Lee et al. [LPP*06] have created a list of tasks that are commonly performed while analyzing graphs. Complex tasks can be broken down into a combination of the tasks listed here.

1.4.2.1 Topology-based Tasks

The basic tasks users need to accomplish when analyzing a network deal with its topology. These tasks form the basis for the more complex tasks.

- Adjacency
  - Find adjacent nodes.
  - Find node with maximum number of adjacent nodes.
- Accessibility
  - Find accessible nodes (i.e. nodes that have a finite distance).
  - Find nodes within a given distance.
- Common connection
  - Given a set of nodes, find all nodes in the graph that are connected to all nodes in the set.
- Connectivity
  - Find the shortest path between two nodes.
  - Identify clusters of nodes.
  - Identify connected components.
  - Find articulation points.

1.4.2.2 Attribute-based Tasks

Topology-based tasks can be extended to compute one or multiple values of the nodes. The results of such computations allow a more high-level view of the network.

- Count all results of a topology-based task.
- Count entities with a specific result for a topology-based task.
- Find all nodes that have a specific result for a topology-based task.
- Obtain the range a topology-based task will return.
- Obtain the distribution of the results of a topology-based task.
- Filter entities that have a specific result for a topology-based task.

1.4.2.3 Browsing Tasks

Browsing tasks move the focus of the user from one node to another.

- Follow a path.
- Return to a previous node.

1.4.2.4 Overview Tasks

In some cases, it is more important to obtain a value quickly than it is to get the exact result. A good visualization will enable the user to estimate the result of a task.

1.4.2.5 High-Level Tasks

There are high-level tasks that are not covered by the above tasks. These tasks generally deal with comparing two different nodes or graphs.

- Compare multiple graphs.
  - Search common features.
  - Search different features.
  - Is some information missing?
  - Is some information from the graphs conflicting?
- Identify nodes that represent the same entity due to erroneous data.
- How has the graph changed over time?

1.5 Thesis Overview

First, Chapter 2 gives an overview of graph analysis research. Chapter 3 describes the underlying system the work of this thesis was implemented in.

Chapter 4 and Chapter 5 introduce an approach to analyze multivariate graphs and a novel visualization for searching node structures in a graph, respectively. Chapter 6 gives some details concerning the implementation of the work done for this thesis.

Chapter 7 gives an example of how the novel techniques can be used to analyze a network. Chapter 8 lists some improvements that could still be made to the approaches described in this thesis. Finally, Chapter 9 gives a brief summary of the contributions of this thesis while Chapter 10 contains the conlusion.

CHAPTER 2

State of the Art

A wide body of research into how graphs can be analyzed has already been conducted. This chapter summarizes the papers that have had the largest influence on this thesis. It starts with modifications to existing algorithms that can easily be integrated into other systems. Next are complete views that are still general enough to be integrated into other systems. Finally, specialized views and applications that offer interesting approaches but cannot be integrated into a larger system without changes are presented.

2.1 Interaction Techniques for Subgraphs

Listing 1 The key actions of the interface created by McGuffin and Jurisica [MJ09].

Left-press-release on node: toggle selection state of node
Left-press on node: popup node-specific radial menu
Left-press-release on whitespace: popup global radial menu
Left-press-drag on whitespace: rectangle/lasso selection
Ctrl-Left-press-drag on node: select and move one node
Ctrl-Left-press-drag on whitespace: move all selected nodes
Ctrl-move: popup hotbox

McGuffin and Jurisica [MJ09] developed a set of techniques for interacting with subgraphs. A single mouse button in combination with a single modifier key to define the currently active mode is used for interaction. By combining the mouse button with the modifier key and pressing either on a node or white space, it is possible to create or modify a selection or move a single or all selected nodes. Listing 1 shows how each action is performed.

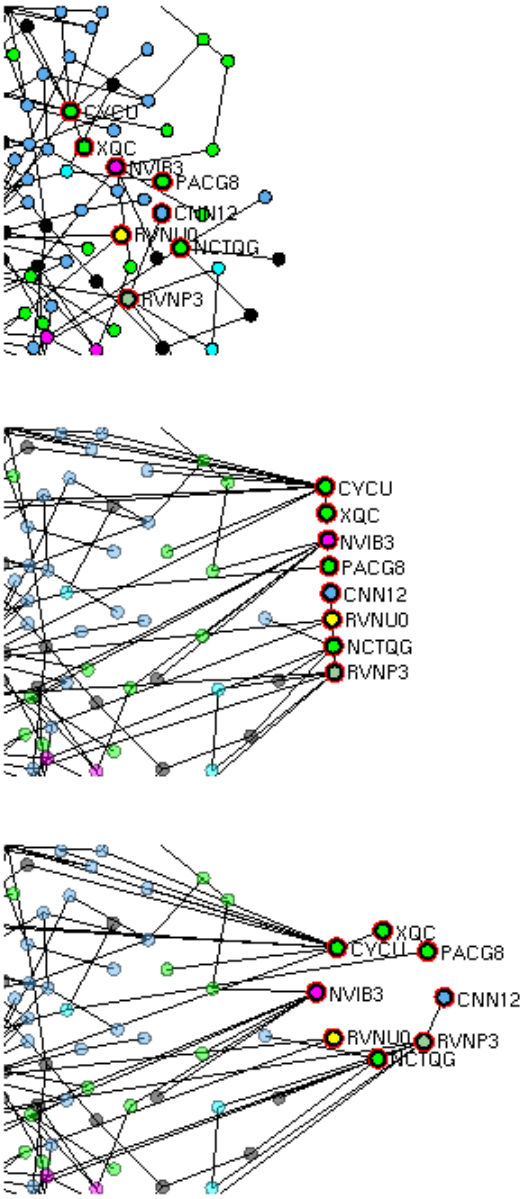


Figure 2.1: Changing the layout of the selected nodes in the system of McGuffin and Jurisica. Left: Original layout. Middle: Moved to the side and linear layout. Right: Circular layout. (Image courtesy of McGuffin and Jurisica [MJ09])

Using the hotbox, it is possible to further modify the selection by adding all nodes that are directly connected to any currently selected node or adding all nodes that are in a connected component that is at least partially selected. It is further possible to adjust the layout of the selected nodes to one of several predefined shapes. An example of this is shown in Figure 2.1. The position of the nodes can then be modified by rotating or scaling the current layout of the selected nodes. Another option is to give the icons of the nodes in a current selection a different shape or size to make it easier for users to keep track of important nodes.

2.2 Constraint-based Cooperative Layout

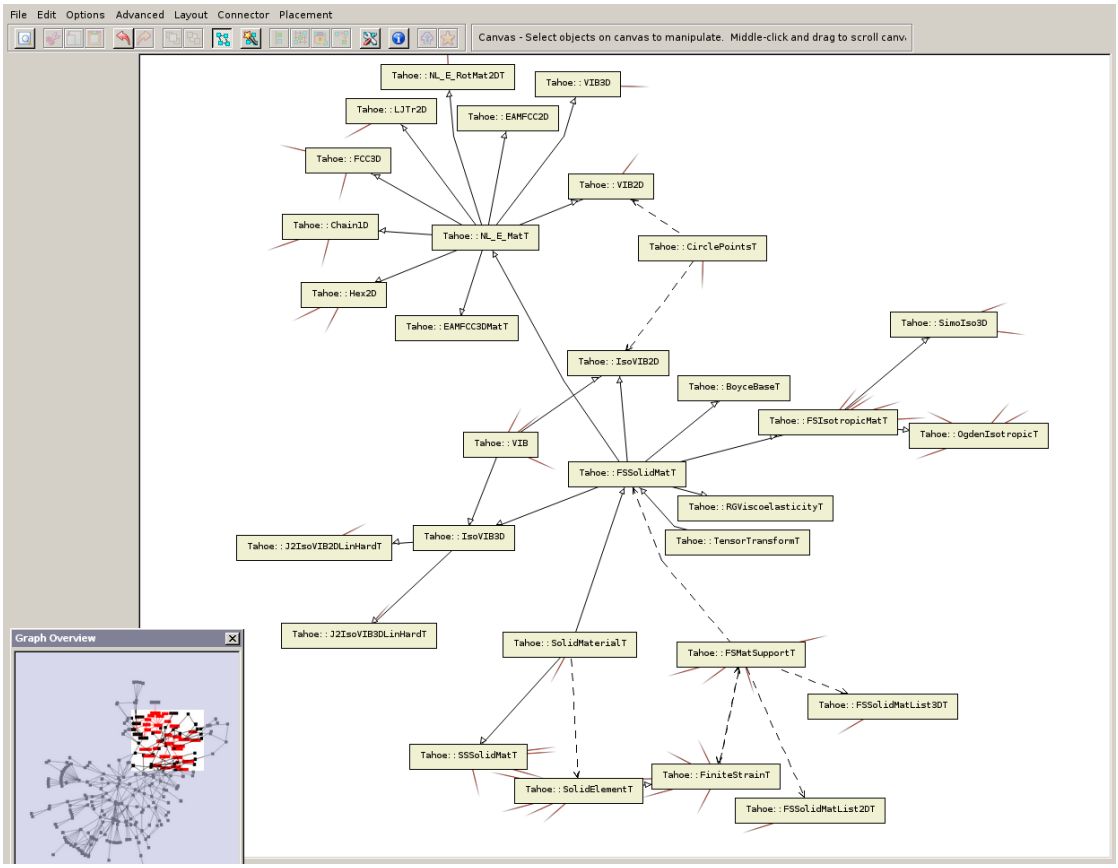


Figure 2.2: Dwyer et al. use different layouts for the overview and detail views. The detail layout is a refined version of the layout used for the overview. (Image courtesy of Dwyer et al. [DMS*08])

Dwyer et al. [DMS*08] proposed to use different node-link layouting algorithms for the overview and detail views of a system with multiple views. This way, complex algorithms can be used to layout the small part of the network that is currently being manually analyzed. The calculation of a layout for the overview can be done with a cheaper algorithm that scales well to the larger number of nodes that is contained in the complete dataset. While the placement of the nodes in the detail view does not have to be identical to the placement in the overview, a large divergence would nonetheless be confusing to the user since a different layout in the detail view would be unexpected. Therefore, it is necessary to find a layout which has a topology that is rather close to the one used for the overview. Since the detailed layout is generally of a higher quality, the higher quality layout of the detail view can also be used to update the layout of the overview. Figure 2.2 shows an example of how the system by Dwyer et al. looks like.

2.3 MatLink

In an effort to combine the advantages of node-link diagrams and matrix-based visualizations of graphs, Henry and Fekete [HF07] combined them into a single link. Their view, called 'MatLink', consists of a matrix that additionally shows edges along the top and left side, overlapping the matrix. This was done to allow users to follow indirect connections that pass through additional nodes before reaching their destination. The shortest connections between selected nodes are highlighted in a different color and are drawn as outward arcs, running over the labels. Another highlight shows the shortest connection between the selection and the node the mouse cursor currently resides over. An image showing the view can be found in Figure 2.3.

2.4 NodeTrix

Henry et al. [HFM07] combined a node-link diagram and a matrix-based visualization to combine their advantages. For this, they cluster dense parts of the network into matrix representations that are displayed inside the node-link diagram. This way, both nodes and clustered matrices are shown and connected within the node-link diagram.

Since this view expects the network to be locally dense while remaining globally sparse, its main use is in analyzing social networks. Social networks commonly follow a power law distribution (see Section 1.2.2.3), which means that the number of nodes that have a certain number of connections declines exponentially with a linear increase of the number of connections. This leads to a topology of loosely connected clusters which are in turn internally highly connected, which is desired for this view.

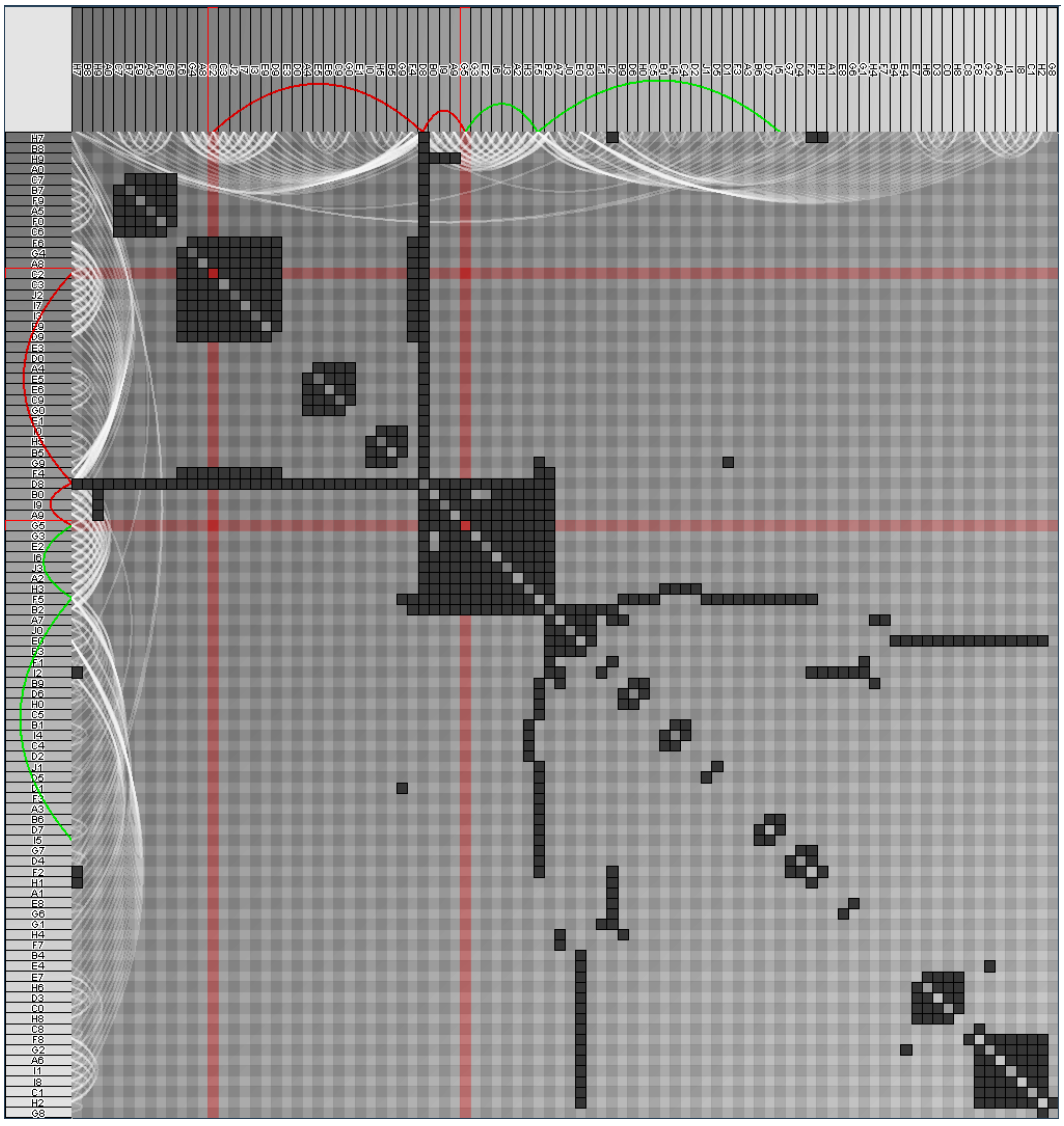


Figure 2.3: The MatLink view by Henry and Fekete has an additional representation of the connections to the top and left border of a matrix diagram to facilitate following paths. (Image courtesy of Henry and Fekete [HF07])

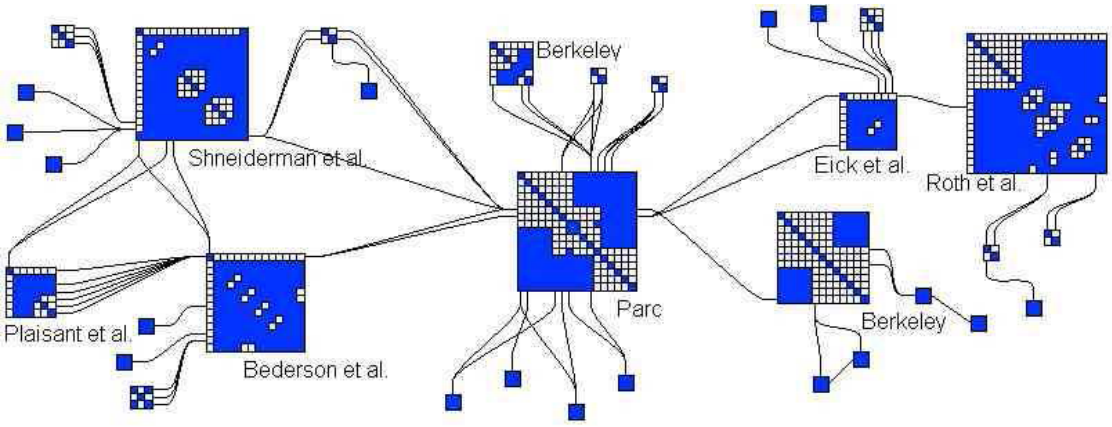


Figure 2.4: The NodeTrix view uses a node-link diagram in which clusters are collapsed to miniature matrix diagrams. (Image courtesy of Henry et al. [HFM07])

As the decision which nodes should be placed into a matrix is not always obvious, the user has the option of adding nodes to an existing matrix or fuse multiple nodes into a new matrix. Removing nodes from a matrix is possible as well. Existing matrices can also be disbanded, placing the clustered nodes into a circular layout at the previous position of the matrix. An image of how the resulting view looks can be found in Figure 2.4.

2.5 MatrixExplorer

Henry and Fekete [HF06] developed a linked-view system that uses a matrix-based view as its basis for the analysis of graphs. In addition to the matrix-based view, their system 'MatrixExplorer' contains a node-link diagram as well as a basic overview that shows fundamental data about the graph like the number of edges it contains. There are also overview versions for the node-link and matrix-based view which show a tiny representation of the completely zoomed out dataset. For the matrix view, there is also an additional view that shows which connected components were calculated to arrange the matrix. All views are linked by a common selection and also use the same filtering rules to decide which nodes are to be omitted from the view. Figure 2.5 shows an image of this system.

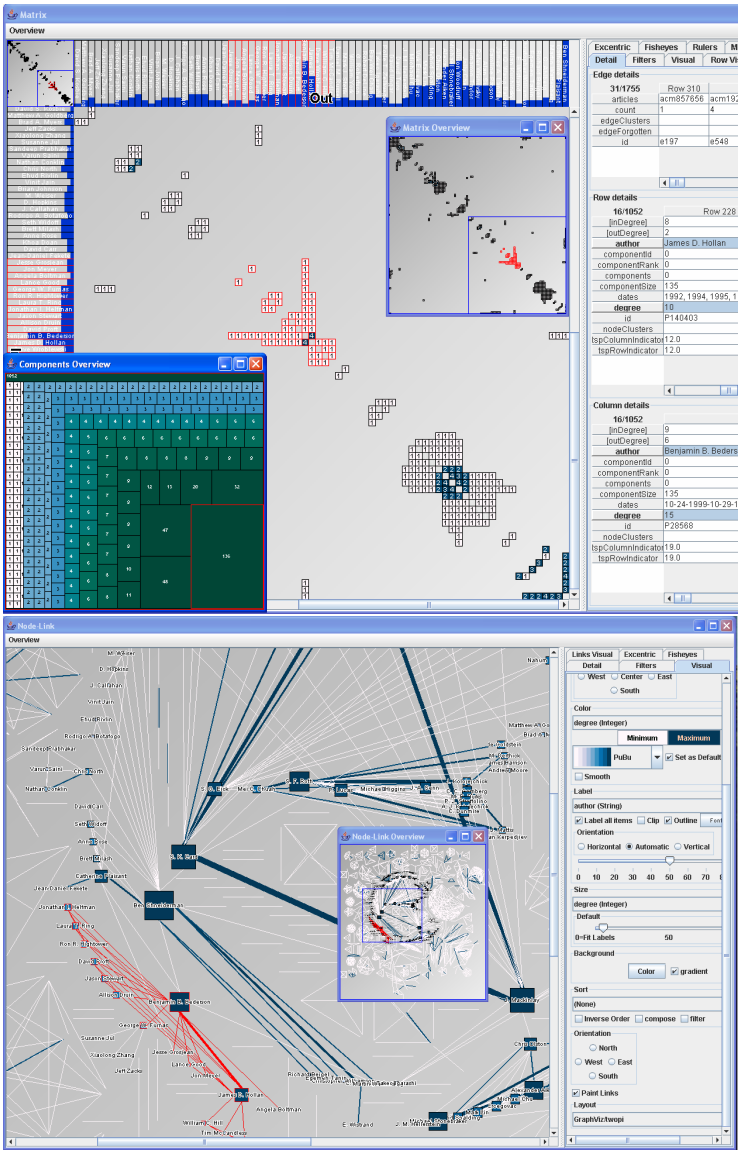


Figure 2.5: MatrixExplorer employs node-link and matrix diagrams in combination with overviews of both to analyze networks. (Image courtesy of Henry and Fekete [HF06])

2.6 PivotGraph

To show connections in multivariate networks, Wattenberg [Wat06] created a visualization called 'PivotGraph' (see Figure 2.6). In this view, the user can assign two categorical node attributes to the horizontal and vertical axis. The different categories of the attributes that are mapped to the axes then form a grid, aggregating all nodes that belong

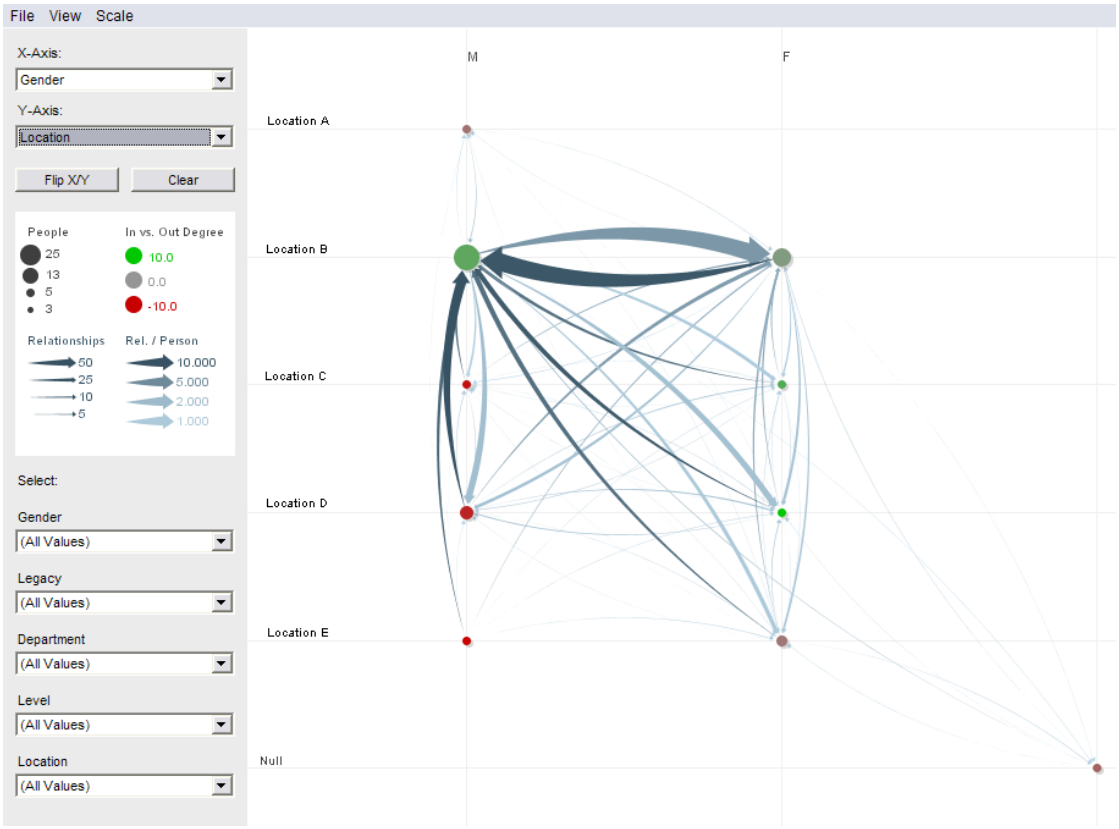


Figure 2.6: Wattenberg maps attributes to the horizontal and vertical axes and aggregates nodes with the same attributes. The resulting view gives an overview of how large and well connected different groups are. (Image courtesy of Wattenberg [Wat06])

to similar categories. To support continuous data values, these have to be binned first to create categories. The size of each grid node represents the number of graph nodes that it represents. The connections between each distinct pair of grid nodes are drawn. As with the nodes, the width of the connection represents the number of connections that are represented by it.

2.7 SocialAction

Perer and Shneiderman [PS06] created a linked-view system that utilizes a node-link diagram as its base view. It contains an ordered list view, a scatter plot and a matrix-based view for the ranking of nodes according to attributes. It can also filter out those nodes that do not belong to a specific interval of the shown attributes. Nodes that are filtered out are still shown in the node-link view, albeit with a reduced size and opacity.

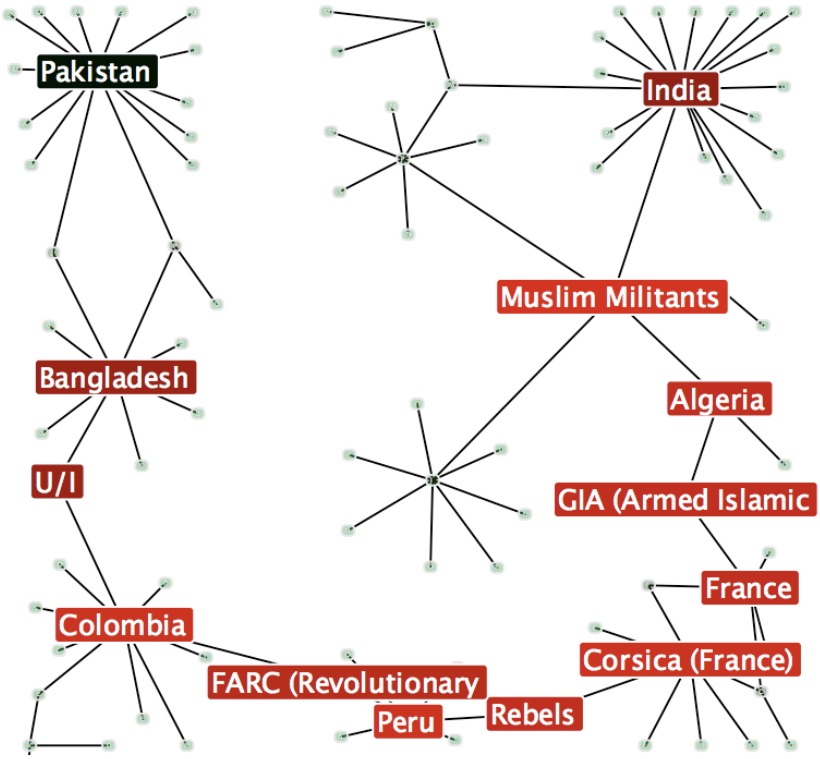


Figure 2.7: Perer and Shneiderman have created a system in which node labels can be colored and filtered based on attributes. (Image courtesy of Perer and Shneiderman [PS06])

Their system also supports timeseries data, which means that multiple sets of edge data are present in the dataset. When the user switches to a different time period, the layout of the node-link diagram remains. It is only updated when the user explicitly requests a new layout or the user decides to analyze a subgroup. This is done to preserve the user's mental map.Figure 2.7 contains an image coming from their system.

2.8 NoodleView

'NoodleView', proposed by Pretorius and Wijk [PV06], is a view that was designed to analyze state transition graphs (see Figure 2.8). It consists of three parts. On the top is a tree view in which each level splits its parent according to a particular attribute. This shows the partitioning of graph nodes into the various nodes of the tree. In the middle of the visualization, at the bottom end of the tree, is an arc diagram, a special form of a node-link diagram in which the nodes are arranged in a single dimension instead of a two-dimensional space, connecting the leaf nodes if there is an edge in the

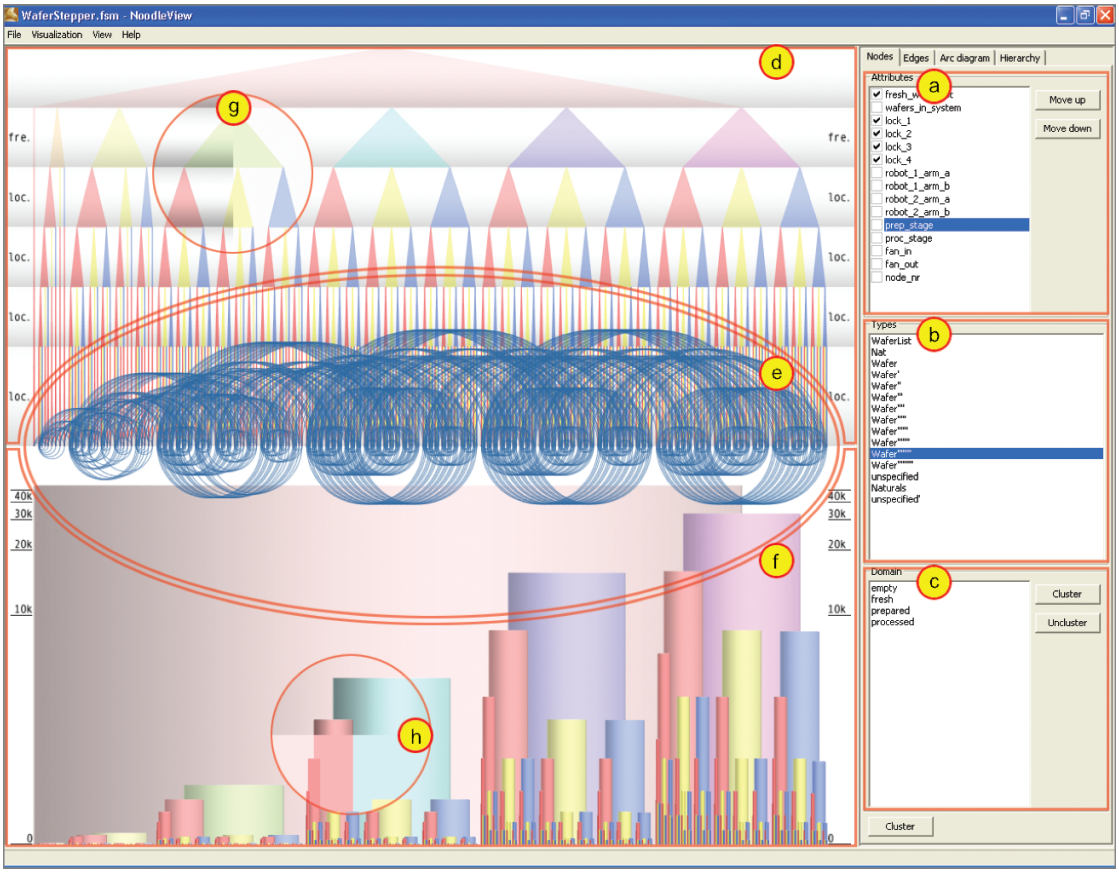


Figure 2.8: NoodleView utilizes bar trees, an arc diagram and a hierarchy tree to analyze sets of nodes that are grouped based on user-selected attributes. (Image courtesy of Pretorius and Wijk [PV06])

dataset connecting them. The bottom of the view extends the tree on top with a novel visualization technique called the bar tree. While the bottom part looks just like a bar chart, the height of each bar once again represents the number of nodes that are present in the node of the tree. Furthermore, it shows not only the leaf nodes of the tree, but all nodes that are in the tree above it. To remain readable, bars that represent nodes that are closer to the root of the tree are drawn behind those that are further from the root to avoid covering them.

2.9 AttriGraph

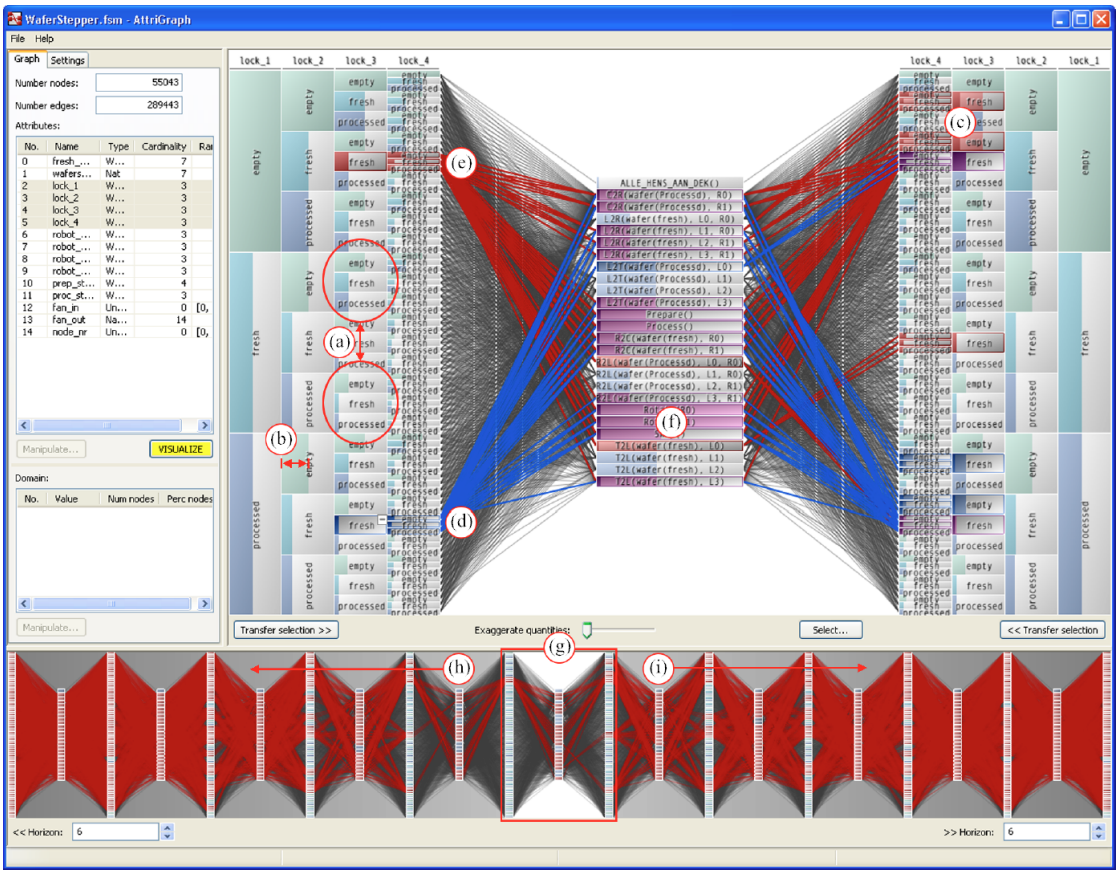


Figure 2.9: The system of Pretorius and Wijk groups nodes and edges and shows which groups of nodes can be reached using different types of connections. (Image courtesy of Pretorius and Wijk [PvW08])

Pretorius and Wijk [PvW08] created a view that is specifically tailored to the analysis of state transition graphs and focuses on node attributes and edge labels (see Figure 2.9). In the center of the visualization is a list of unique edge labels that are available in the dataset. To both the left and the right, the set of nodes is partitioned based on the categories they belong to in user-chosen attributes. Lines connect the nodes with the edge labels if an edge with such a label is connected to at least one of the nodes in the cluster. It is thus possible to see which actions can be performed from which state as well as which state can be reached with a specific action. A view at the bottom of the visualization shows which nodes can be reached after multiple steps. This can be used to see how many steps are necessary to reach a specific state from another specified state.

2.10 Network Visualization by Semantic Substrates

Shneiderman and Aris [SA06] modified a node-link view to split the nodes into non-overlapping groups according to their attributes. To keep the visualization tidy and easy to work with, no edges are visible by default. Edges have to be specially enabled for each combination of groups. Each combination of starting- and target group can be selected with a checkbox. Only edges that start and end in groups that are selected are shown. To further reduce the number of visible edges, the user can select a region an end node of the edge has to lie in to be visualized. The view that resulted from their work can be seen in Figure 2.10.

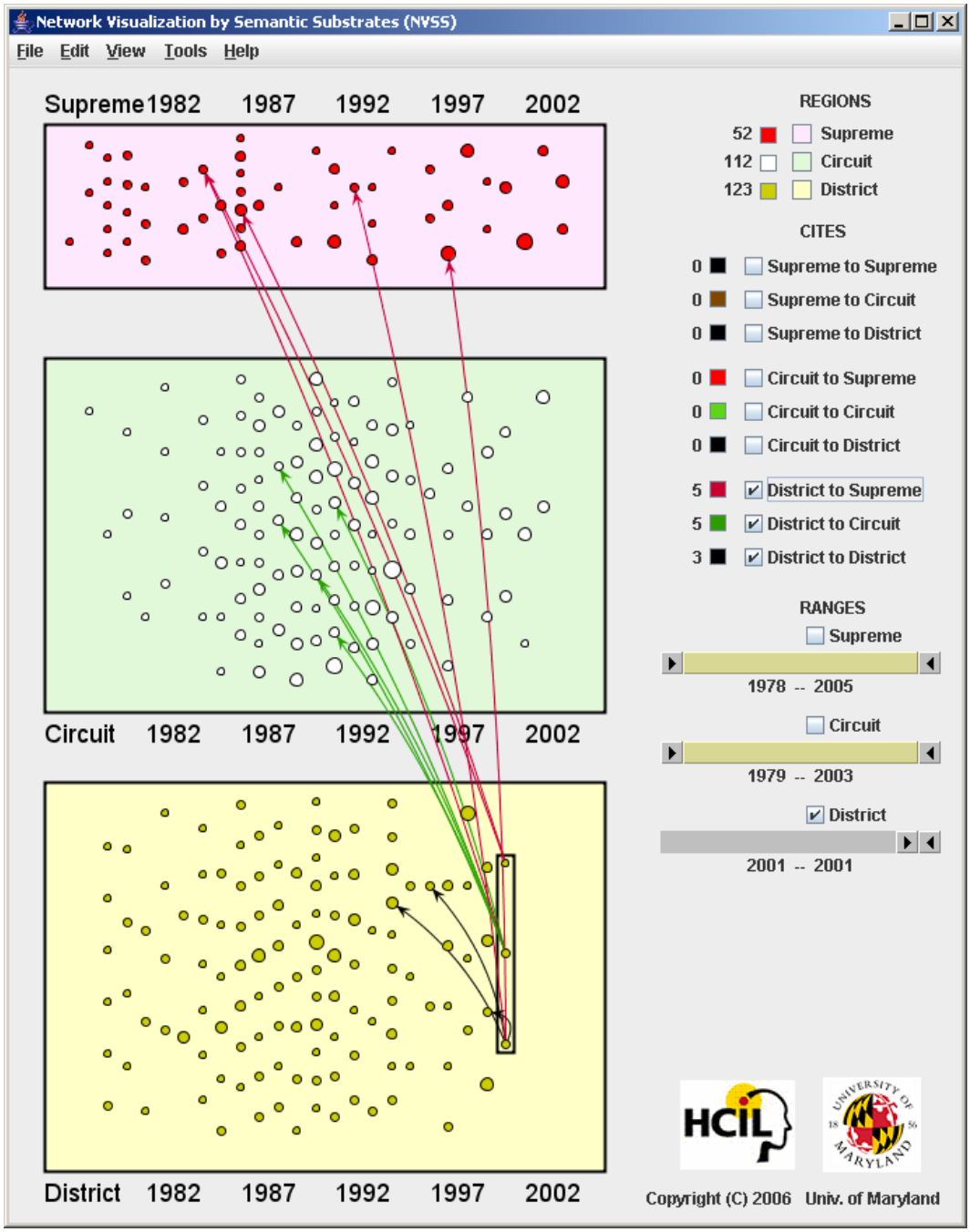


Figure 2.10: Shneiderman and Aris split a node-link diagram into non-overlapping regions containing nodes with similar values of an attribute. Only links for which at least one node is selected are shown. This highlights the connection between regions. (Image courtesy of Shneiderman and Aris [SA06])

CHAPTER 3

The Framework

3.1 Introduction

All contributions of this thesis have been implemented as plugins for VISPLORE. VISPLORE is a system for the interactive analysis of large datasets using exploratory data analysis as described in Section 1.2.4.5. It implements multiple linked views (see Section 1.2.4.1) and currently has more than 10 different views, including both well-known (e.g., histograms) and specialized ones [PBH08].

Section 3.2 describes VISPLORE from the perspective of a user. Section 3.3 provides a more technical explanation of the system and summarizes the technological concepts that are used in order to achieve visual feedback at interactive rates even for massive datasets.

3.2 User Experience

The first element of VISPLORE a user experiences is the importer. Importers for comma-separated value files (.csv) as well as several more specialized file formats are available.

After importing data, the system presents the user with its main interface, a labeled version of which can be seen in Figure 3.1. The left part accommodates the data manager as well as the settings of the currently active visualization. The data manager contains a structured list of channels. Each channel contains a single value for each entry in the imported dataset (e.g., the temperature of each measurement). Below the data manager are the visualization settings. The available options differ for each view. The central part of the system contains the actual visualizations of the data. In this area

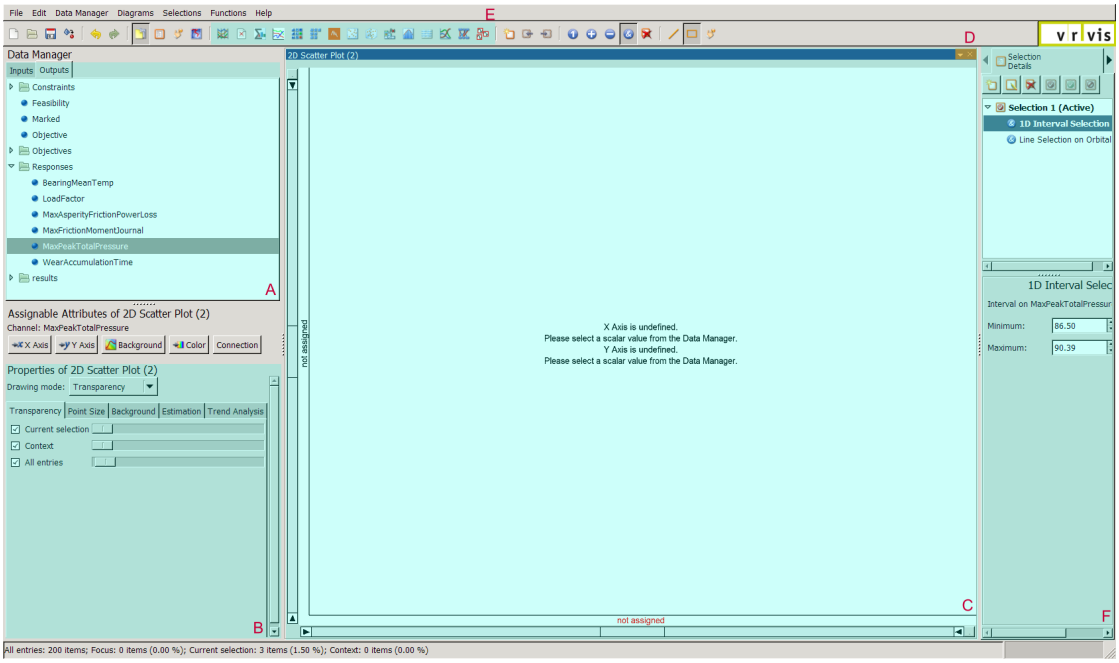


Figure 3.1: Parts of the VISPLORE system. A: Data manager B: Active view settings C: View area D: View menu E: Buttons to open new view F: Selection details

each view has its own subwindow. On the top right of each subwindow is the view menu which contains a button to hide the view and a dropdown for various tasks such as saving an image of the view or closing it. On the top of the system is a toolbar. In addition to shortcuts for saving, loading, undo, redo and selection options, which will be discussed further on, the toolbar contains buttons to open additional views. On the right side, an additional pane displaying the current state of the selection(s) is shown if the user has activated it.

The border between views can be moved by dragging it to adjust the area assigned to the views. The arrangement of the views can also be modified by the user by dragging a view to its desired location. It is also possible to move a view into a floating window from the dropdown. The number of views that can be open simultaneously is not limited by the system. To keep the layout organized when a large number of views is open, views can also be hidden to be reopened later.

Most views need to be parameterized before they can be used. For this, channels from the data manager can be assigned to the available attributes of the visualization. The number and type of the required attributes differ for each view.

Attributes that are not explicitly defined in the imported dataset but can be calculated from the available data can be assigned by using derived channels. Derived channels are composed of a computation, parameters and source channels and can be created directly from the data manager through the context menu of a channel. An example of a derived channel is 'month', derived from a timestamp.

For visualizations in which the spatial layout is significant, icons that lie in close proximity to each other might overlap, leading to occlusion which can significantly distort the message the image sends. To alleviate this problem, the mapping from the space the data lies in to the screen space can be modified. VISPLORE provides a widget for the modification of the applied mapping to assure a uniform style. The widget can be seen in Figure 3.2 on the bottom and left side of each view. Only the mapping widget on the bottom of the curve view is currently used to zoom into the data in the image. By dragging the slider on either side, the range that is displayed in the view can be restricted. In many cases, it is preferable to see the context in which the area of interest is embedded. For this, focus & context techniques, explained in Section 1.2.4.3, can be employed. A Bifocal Display distortion [LA94] can be created by interacting with the central part of the widget. A Bifocal Display uses two magnification factors: a positive one for the region of interest and a negative one for the context to map the complete range onto the screen. It provides spatial continuity but contains two magnification discontinuities. A restriction and a distortion of the displayed area can be used simultaneously. The view is updated during both interactions. This allows the user to adjust the goal of the interaction while it is still being performed.

In some fields, various values the entries possess might be optional. VISPLORE can deal with datasets that contain such missing data. Users are notified that some values a visualization relies upon are missing by a widget that displays two bars. One bar represents the available entries while the other one represents the missing entries. These bars are scaled by the number of available and missing entries. If all entries are available, the widget is automatically hidden.

When multiple views of the same dataset are open at the same time, matching multiple representations of the same entry is important. The concept of using multiple linked views, described in Section 1.2.4.1, uses selections that affect all views to allow users to search for an individual entry or a whole range of entires in all views simultaneously by brushing the entries in a single view. The system supports the creation of multiple user-defined selections and each selection in turn consists of an arbitrary number of selection components. These selection components are low-level brushes that are defined and modified interactively by the user, e.g., an interval in a histogram or a rubber-band in 2D scatter plots. The components of each selection are composited using different set operations. The available combinations consist of replace, unison, complement and

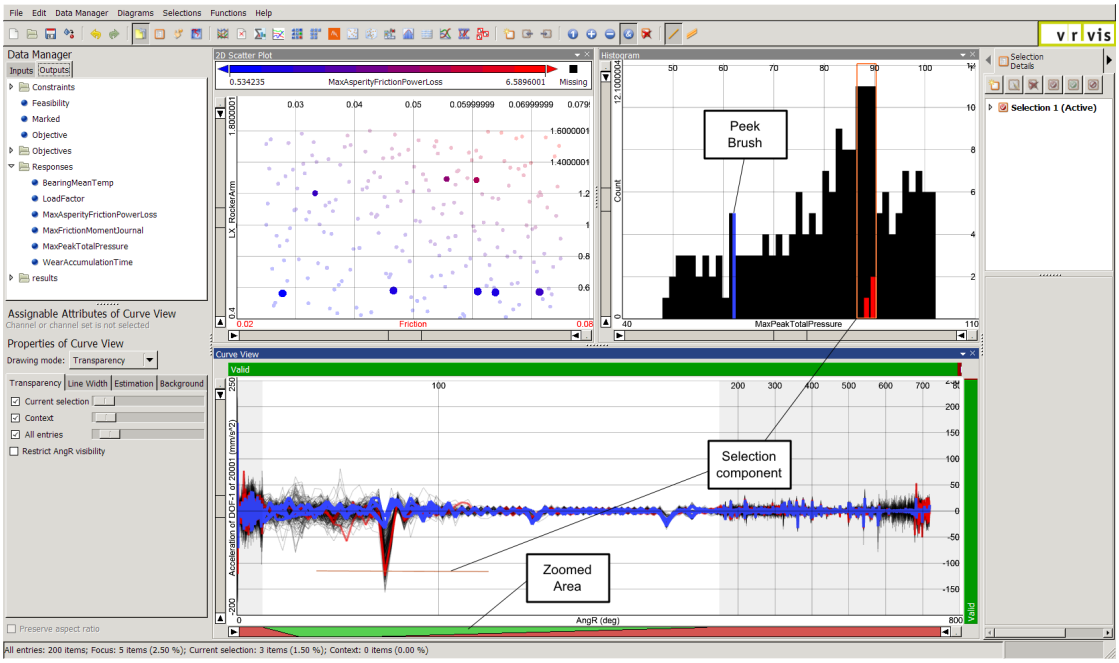


Figure 3.2: The VISPLORE system, showing multiple views simultaneously. Several key elements are annotated in the image.

intersection. The currently defined selections can be reviewed and modified by using the toggleable selection sidewindow. It is also possible to assign exact values to the borders of the defined selections through the sidewindow.

VISPLORE also employs a technique called 'Peek Brush' [BP10] to help reveal relationships across multiple views. A Peek Brush is a lightweight brush that temporarily selects the entities under the mouse cursor. Similar to regular brushes, the results of the interaction are highlighted in all views to allow the user to quickly identify relationships between different data projections. Some views do not support selecting entries using the Peek Brush but provide passive support for it by highlighting the entries that are selected by the Peek Brush in other views. Since hovering over a visualization requires minimal interaction effort, the Peek Brush can be efficiently used to define the starting point of a more focused inspection using brushes with higher complexity. The Peek Brush does not affect selections and is visually distinct from both active and inactive selections.

Besides selections and the Peek Brush, color can be efficiently used in information visualization to map the values of a data dimension onto a color band. VISPLORE uses its default mapping widget for this mapping operation and also provides separate colors

for values that are outside of the mapped range (one color for higher values and one for lower values) or missing entirely. A uniform coloring legend that displays the mapped colors is provided by the system.

VISPLORE also implements several amenity functions necessary for the acceptance of a productivity system in a professional environment. These functions include the obligatory save and load as well as undo and redo operations. It is also possible to either export the state of the current session to a structured xml file or export (either part of or the complete) data to a comma separated value file for use in other systems.

3.3 Technical Aspects

VISPLORE is written in C++, uses GTK+ as its GUI library and OpenGL for rendering. The system uses a single table in which the imported data as well as all created derived channels are stored in. The table is spanned by the individual entities and the available attributes.

The following data types can currently be used by the system for the individual data entries:

- Boolean
- Integer
- Float
- Timestamp
- String
- Curve (1-dimensional function)
- Surface (2-dimensional function)

The actual views of the system as well as the importers needed to load the various data files are implemented as plugins, which provides a number of advantages. Since plugins are naturally isolated from each other, multiple programmers can each work on their own plugin without the danger of creating conflicting code. This separation makes it possible to create new plugins without negatively affecting the stability of the complete product by only shipping plugins that are already in a mature state. Customers can further be enabled to add their own functionality on top of an existing framework without having access to the actual source code. Should a plugin be discontinued, it can easily be removed without causing any dependency problems.

3.4 Multithreading Architecture

Systems that focus on exploratory data analysis (see Section 1.2.4.5) need to ensure a low latency between interaction and visual feedback to allow for continuous user interactions. The founding block VISPLORE relies upon to provide the interactive framerates needed for a smooth and efficient exploration is its multi-threading architecture [PTMB09].

The employed multi-threading architecture has been designed from the ground up to provide a responsive interface that displays visual feedback as quickly as possible. The acceptable delay between a user action and the visual response has been decided to be 100 ms. It has been designed to scale well both with regards to the number of concurrently displayed visualizations and the size of the dataset, providing interactive framerates even for millions of entries.

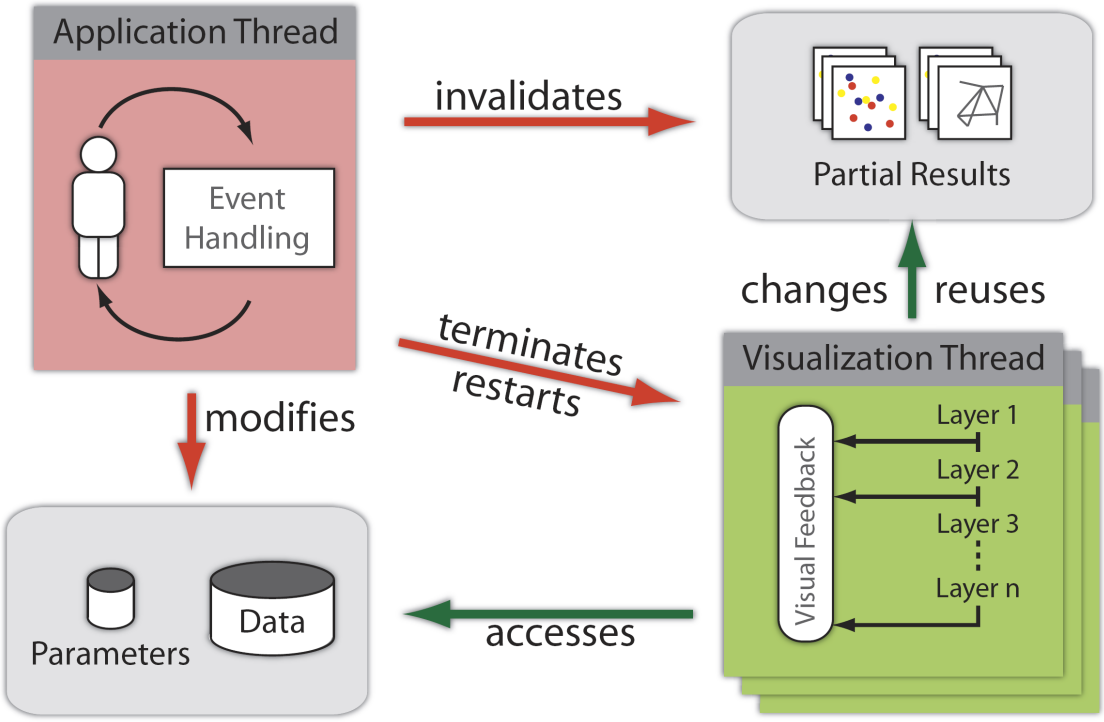


Figure 3.3: An overview of the threading architecture used by VISPLORE. It shows how threads access the data and interact with each other. (Image courtesy of Piringer et al. [PTMB09])

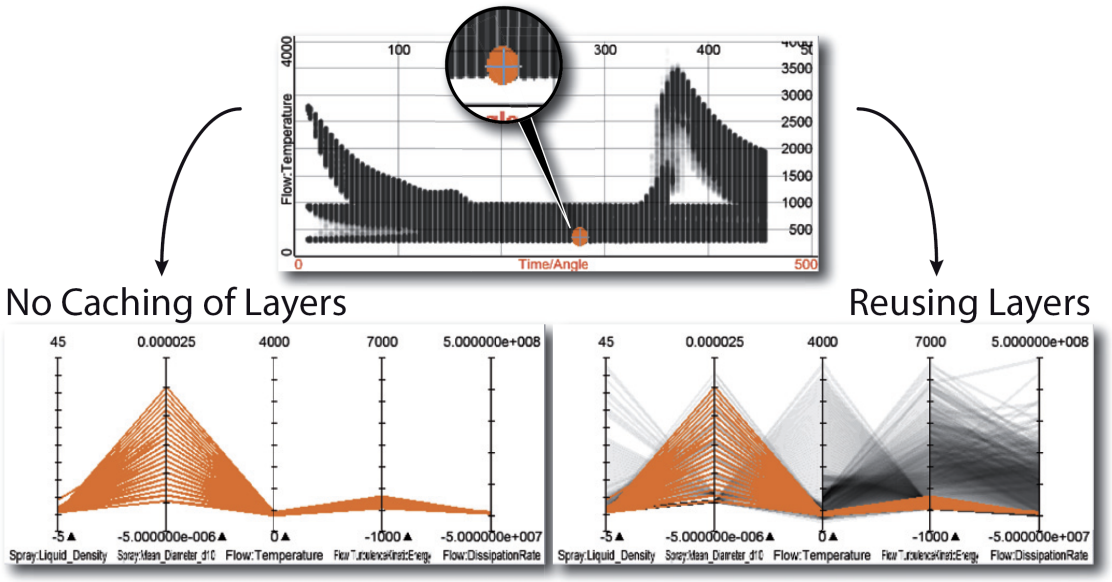


Figure 3.4: Displaying the selected entries in a parallel coordinate view with and without reusing image data of the unselected entries. Both images take approximately the same time to display. (Image courtesy of Piringer et al. [PTMB09])

VISPLORE runs one dedicated visualization thread for each open view in addition to its main application thread. The main application thread is responsible for handling user requests. To keep the event loop alive, and therefore the application responsive, this thread is restricted to perform only inexpensive tasks. The tasks of the main thread are therefore limited to adjusting parameters, invalidating calculated results and triggering updates. Each visualization thread is responsible for creating the image its parent view should display, handling both the necessary computations and the drawing of the final image. An overview of this architecture is presented in Figure 3.3. During work, visualization threads need to check frequently if they should terminate early due to a state change caused by user interaction.

The delay between a user action and the image resulting from it is a result of the delay between the interaction and restart of the visualization thread plus the time needed to calculate and present the result. To keep the time required for calculations short, the main thread only invalidates results that have become obsolete. Reusing results of a costly computation like binning or clustering the data can significantly reduce the time needed for the visualization thread to finish.

Parallel to this caching of results in data space, it is also possible to cache results in view space by splitting the complete image into multiple layers and storing images

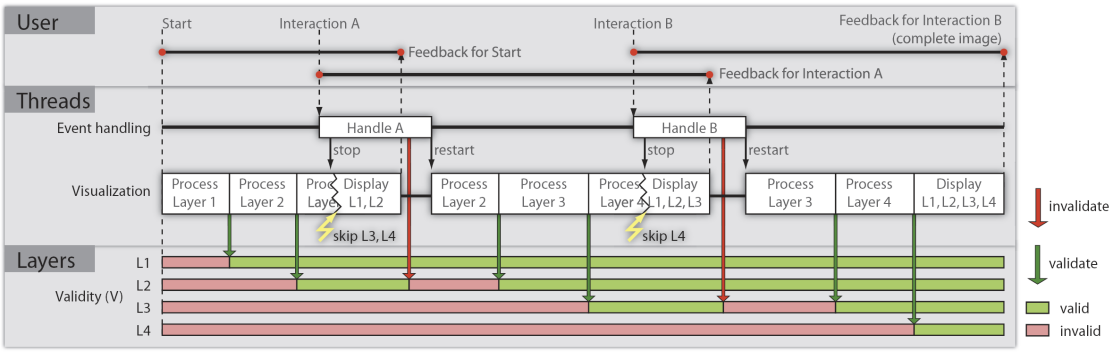


Figure 3.5: An example showing how user interaction interrupts the calculation and invalidates partial results. The results that remain valid do not have to be recomputed after the interaction is over. (Image courtesy of Piringer et al. [PTMB09])

of completed layers. Layers can either be semantically different parts of the visualization, i.e., a background or the selected entries, or disjunct subsets of the entire dataset. Similar to cached calculation results, the images can be invalidated by user interaction or used while they are valid to skip the drawing operations necessary to replicate them. Figure 3.4 shows an example image with and without cached layers. Both images take approximately the same time to draw.

A combination of large dataset, complex views and continuous user interaction can cause the visualization thread to continuously terminate before showing the final image. To counteract this, some views start by drawing a lower quality preview image before starting with the final drawing. As with caching, a preview rendering can contain reduced complexity in both data and view space. A preview with reduced complexity in data space might display only major trends, clusters or other aggregations of the data while a reduced complexity in view space might disable anti-aliasing or display only a sample of the whole dataset. In addition to the obvious delay in the production of the final image, the main disadvantage of such preview renderings is that it can cause flickering while switching between preview and final image. This has to be taken into account when implementing preview rendering for a view. An example of how the state of a view changes during view interactions can be seen in Figure 3.5.

CHAPTER 4

Feature-based Network Visualization

4.1 Introduction

When considering the basic tasks that need to be performed during the analysis of a graph, which have been listed in Section 1.4.2, it can be seen that most topology- and attribute-based tasks result in either a set of nodes or one or more numerical values. To build on the strength of the underlying multi-view system and its preexistent visualizations, providing access to the numerical results of such tasks is necessary. By providing the results of topology- and attribute-based tasks, it becomes possible to analyze and navigate through them like any other available data dimension.

For example, by mapping node degree, which counts the number of nodes each node is connected to, to one axis of a scatter plot, it is possible to search for correlations with other data dimensions and identify outliers. These outliers could then be used as a starting point for another examination of the data.

Another example is the parameterization of a histogram with the distance of each node to a previously identified node of interest. The histogram can then be used to examine the distribution of distance values. It is also possible to use a histogram with such a parameterization alongside a node-link or matrix view of the graph and use it to select interesting distance ranges from the original node of interest. This selection then highlights the nodes in the node-link or matrix view. By dragging the selection in the histogram, different distance ranges can be studied in a swift and efficient way.

This thesis uses the term ‘feature-based network visualization’ to describe the analysis of topological graph data using visualizations for conventional multivariate data analysis and their usage to interactively select parts of a network for further analysis. Section 4.2 explains how attributes that describe the whole graph (e.g., number of nodes) are handled while Section 4.3 contains the approach taken to analyze attributes that vary by node (e.g., numberof adjacent nodes). Finally, Section 4.4 contains the features of a node-link view that has been implemented in the scope of this thesis to supplement the existing visualizations with a view that can explicitly visualize edge data.

4.2 Global Graph Properties

Some tasks give a single value to describe the entire graph. An example of such a task is counting the number of connected components (see Section 1.2.1.10). Values that do not describe a single node, but rather the entire graph are shown in a specialized view which can be seen in Figure 4.1.

This view consists of a table in which each line gives the result of a different computation. Computations can be added to or removed from the view at any time by the user. In addition to the result for the complete network, results for the current selection and the union of inactive selections is displayed as well in different columns. All values are updated in realtime, enabling users to adjust selections while seeing how this change affects the results during the interaction.

For example, to find out how well-connected different age groups in a social network are, one could drag a range selection over the age parameter. This can then be seen by watching how the number of nodes and edge updates since the relation of edges to nodes implicitly gives the number of edges selected per node. Using this approach, this view can be used to analyze a selected subset of nodes as well as the complete network.

The following graph attributes have been implemented in the scope of this thesis:

- Number of nodes
- Number of edges
- Number of connected components (see Section 1.2.1.10)

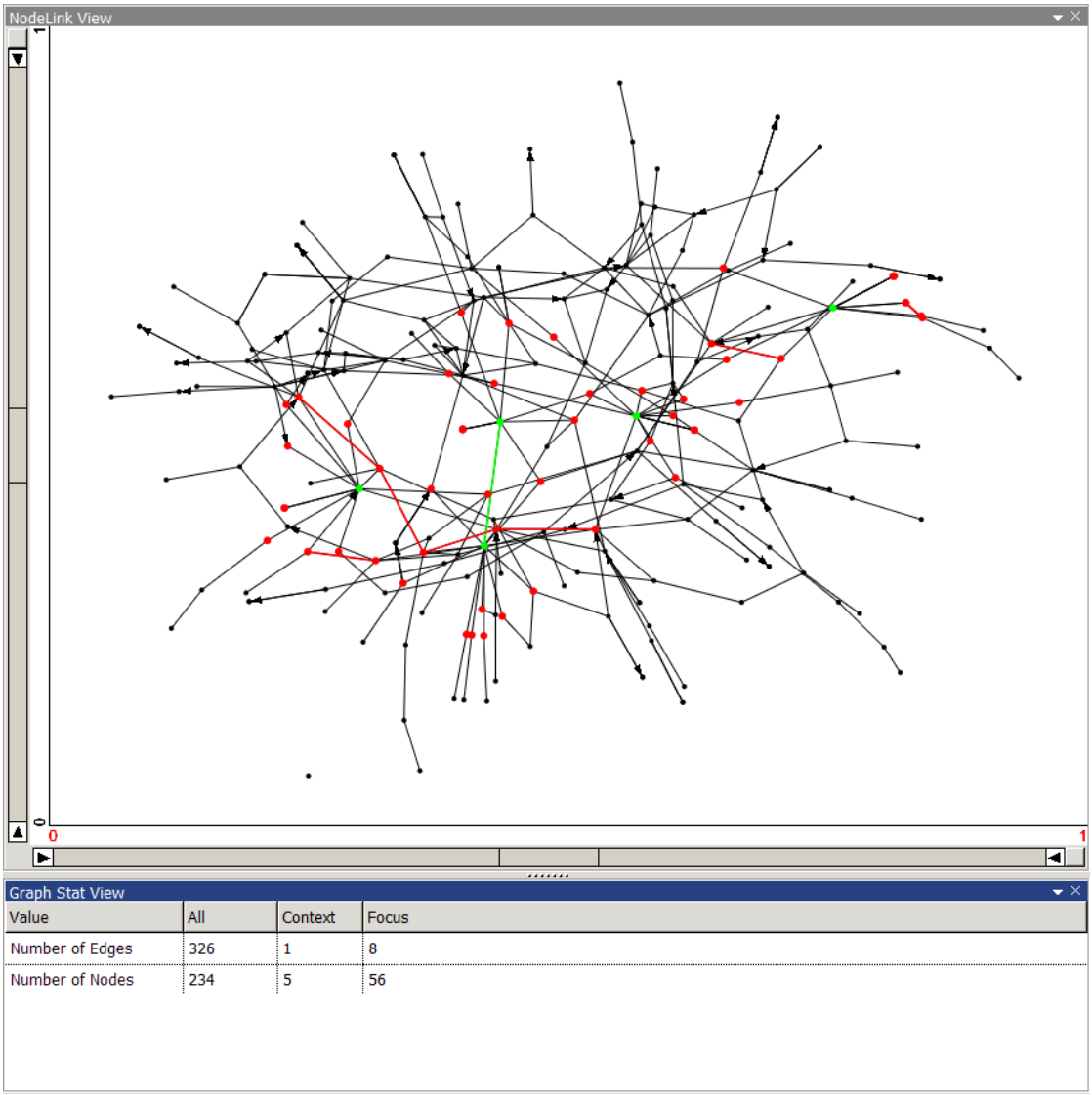


Figure 4.1: The view for graph properties with a node-link view at the top to show the context. (Green: inactive selections, red: active selection)

4.3 Node Properties

A common result of a graph-based task is a numerical value that is valid for a single node. Examples of such tasks are counting incident edges to get the degree of a node or determining the distance between two nodes. Figure 4.2 shows a view in which a graph attribute of the node is plotted against one of the values explicitly included in the dataset.

The results of tasks that give per-node values can be added to the set of available channels as a derived channel. These channels can then be assigned to any view in the system that accepts channels containing numerical values similar to channels that have been created at import time. The views can then be used to investigate the distribution of result values for the task. To obtain the result of a specific node the computed channel needs to be assigned to a view. The entity can then be identified by brushing and linking and the value read from the view. This way, the approach taken in this thesis puts emphasis on the inspection of the distribution of the result values at the cost of a more direct way to obtain the result for a specific node.

Just like graph attributes, node attributes can also be computed considering the active selection or a union of all inactive selections. This is set during the creation of the channel, so unlike graph attributes, multiple instances need to be created to view the computation of the complete graph and a subgraph at the same time.

Typically, computed channels for node attributes that give a result for a subgraph are automatically updated every time the subgraph changes to reflect the new state of the subgraph. However, since node attributes often serve as the foundation of an in-depth inspection of the graph, this behavior can be disabled during the creation of the computed channel. Updates of the computed channels are performed in a background thread to allow the system to stay responsive during updates.

The following node attributes have been implemented in the scope of this thesis. It is possible to specify that only the current selection should be used for the computation. Other computation-specific parameters are included in the list.

Node degree calculates the number of edges that are connected to the node (see Section 1.2.1.3). For directed networks, is is possible to specify if only in- or outbound edges should be counted.

Distance from selection calculates the shortest distance (see Section 1.2.1.5) from a specific node to any node in the current selection.

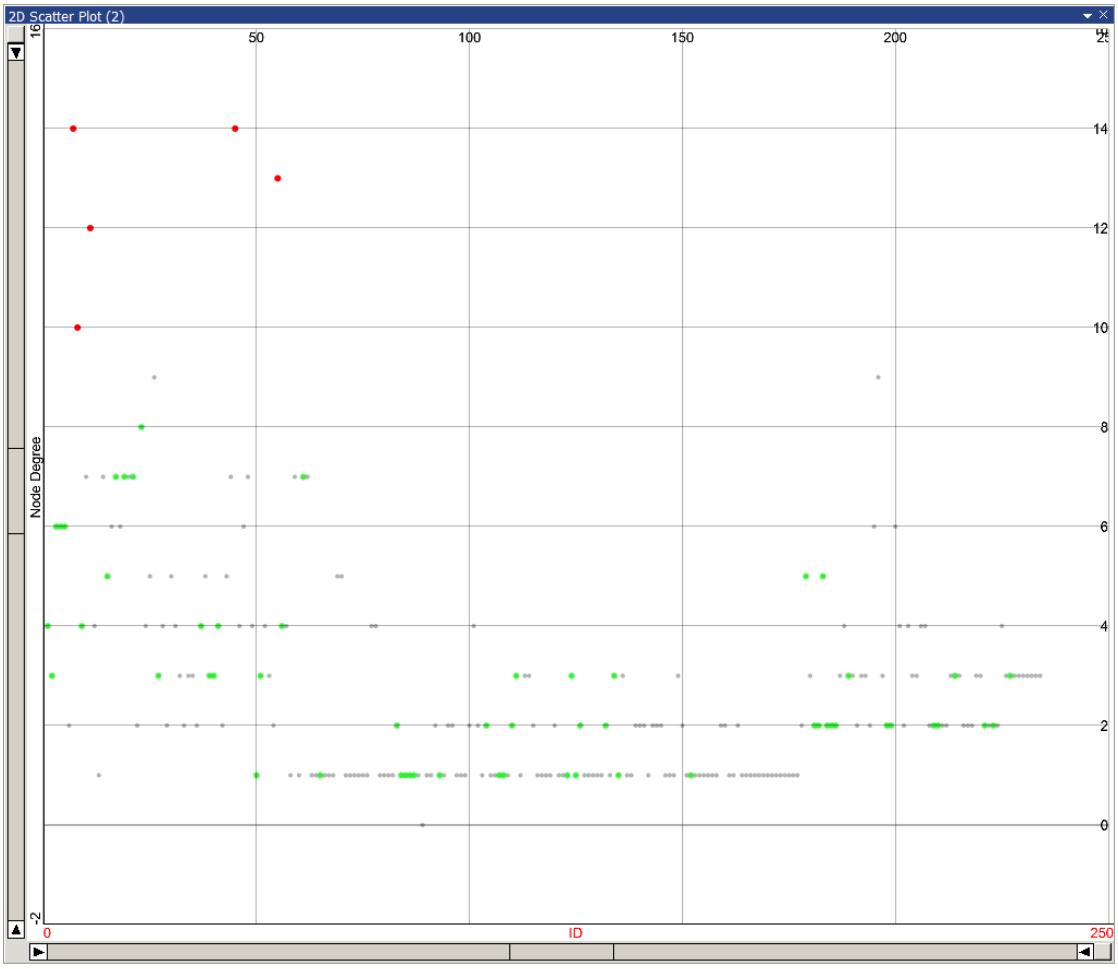


Figure 4.2: A scatter plot showing the entity ID plotted against the number of connected nodes of the entity. It can be seen that the entities with an ID in the upper and lower third of the range are better connected than those in between, with the lower IDs having the highest average number of connections.

Betweenness centrality indicates how often a specific node is passed through on a shortest path (see Section 1.2.1.7). The user can choose between the absolute and relative betweenness centrality.

Component enumeration indicates in which connected component each node lies (see Section 1.2.1.10).

Articulation points mark the nodes whose removal would split the whole network into multiple parts (see Section 1.2.1.10).

Current selection stores for each node if it is currently selected. This is used to define groups for the Parallel Distance view (Chapter 5).

4.4 Node-Link Diagram

4.4.1 Overview

While the analysis of a graph using derived channels can lead to valuable insights, it cannot replace traditional views that display an explicit visual encoding of the edges that make up the graph. A node-link view has been implemented over the course of this thesis to overcome this insufficiency. While node-link diagrams are harder to read for large networks, a node-link representation has been chosen due to the fact that they are easier for non-experts to understand, as shown by Ghoniem et al. [GFC05]. In our experience, the node-link view has proven to be a valuable asset when performing a feature-based analysis of a network.

This section will explain the view and the extensions made to it. The main goal of this view is the exploration of smaller networks and subgraphs of large networks. For large networks, in addition to a feature-based analysis and the node-link view, a specialized visualization has been developed in the scope of this thesis. This visualization is presented in Chapter 5.

4.4.2 Layouting

The readability of a node-link diagram depends mainly on the number of displayed nodes and their layout. A short overview of existing layouting algorithms has been given in Section 1.3.

In addition to previously published layouting algorithms, the implemented node-link view can also position the nodes according to two user-defined channels. These channels then specify the x- and y-position of the nodes similar to a scatter plot. Aside from allowing this view to be used like a scatter plot which draws edges in addition to

nodes, parameterizing the node-link view with a layout from two channels can also be used as a way to import a layout computed in another system. For this the channels containing the precomputed layout are imported with the dataset and then assigned to the view.

When a new layout is assigned to the node-link view, an animation showing a linear interpolation from the old layout to the new layout is displayed. This animation makes it easier for users to preserve their mental model of the data compared to a simple switch to the new layout. Section 4.4.4 contains an approach that further helps users keep track of relevant nodes during a layout change.

After the animation to the new layout has finished, the user can drag individual nodes around using the mouse. This can be used to position nodes the user deems important more prominently or relocate nodes the algorithm has positioned awkwardly.

The following layouting strategies are available in the node-link view:

Position from channels places the nodes at the positions specified in two data dimensions chosen by the user. Unlike the other available layouting algorithms, the positions produced by this layout have a meaning that is derived from node data available in the dataset. As a drawback, it does not explicitly consider any of the aesthetic considerations outlined in Section 1.2.3.1.

Fruchterman-Reingold is a force-directed algorithm. It models a physical system in which nodes repulse each other with an additional attractive force between connected nodes (see Section 1.3.5).

Unstructured assigns each node a random position.

Gürsoy-Atun uses a self-organizing map to layout the nodes in a user-specified shape. The resulting layout matches the shape while keeping connected nodes together [GA00]. The available shapes are square, circle and heart.

Circle arranges nodes along the perimeter of a circle (see Section 1.3.3).

4.4.3 The Visualization

The view displays the nodes as circles and the edges that connect them as straight lines. Directed edges additionally have an arrowhead that points toward the target node. The size and opacity of nodes and edges can be adjusted by the user.

The node-link diagram is connected to other visualizations by linking & brushing (see Section 1.2.4.2). This makes creating and modifying selections one of the main interactions with the view. Two different selections have been implemented into the view. The first one is a 2D interval selection, which selects all nodes that lie in a given rectangle in the view. The second one is a node selection which selects all nodes that currently lie under the mouse cursor. This selection has proven to be very useful to quickly add or remove nodes to a larger selection.

'Peek Brush', a lightweight temporary selection of items under the mouse cursor (see Section 3.2), is also supported by the view. When the mouse cursor hovers over one or more nodes, these nodes form the Peek Brush selection. When no nodes are under the mouse cursor but at least one edge is, the two nodes connected by each edge under the mouse cursor are added to the Peek Brush selection.

Each node is colored according to the highest ranking layer it is selected in. Each edge is colored according to the highest ranking layer that contains both incident nodes. Alternatively, nodes and edges can be colored according to a channel from the node dataset. If this is chosen, each node is colored according to its value in the color channel and a user-modifiable transfer function. The color of edges then interpolates the colors of the incident nodes in color space.

While the color of nodes and edges is usually defined by a layer, the opacity of each layer is not predefined and can be modified by the user.

4.4.4 Visual Scalability

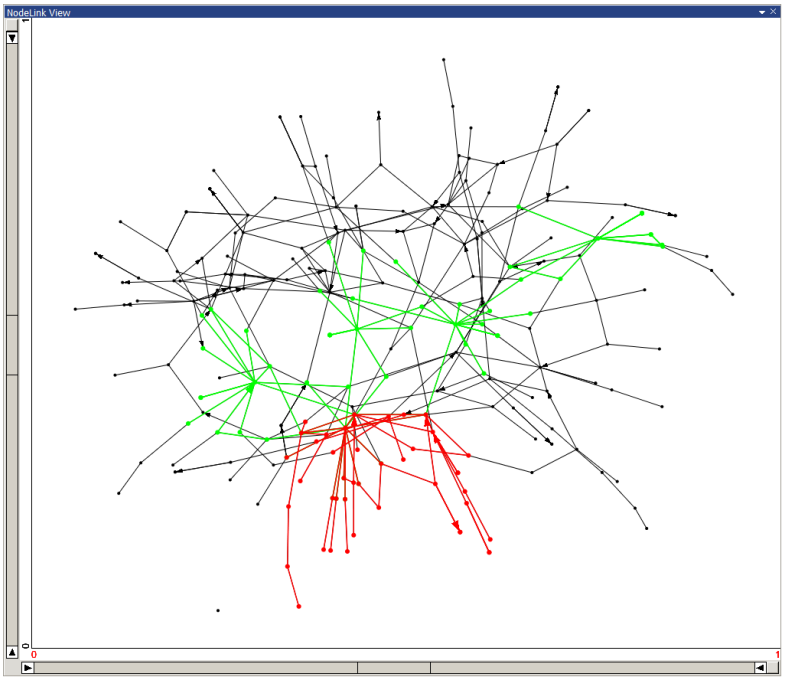
One of the main challenges of a node-link diagram is to remain readable when a large network is being analyzed. The implemented node-link view includes several approaches to help users study interesting nodes and subgraphs of a large network. Chapter 5 describes the approach taken to deal with large datasets in which no nodes of interest have been identified a priori.

The most elementary approach to focus solely on a subgraph of a large network in a node-link diagram is to display only the nodes and edges of the subgraph. For this, the user can choose to only display the current selection when layouting. This sets all nodes that are not selected at that time as background. Background nodes remain at their previous position and are drawn either highly transparent to show the context the focused nodes lie in or not at all, as chosen by the user. The set of background nodes is only modified when a new layout is calculated, so the selection can be modified to analyze part of the subgraph without removing nodes from the set of visible nodes.

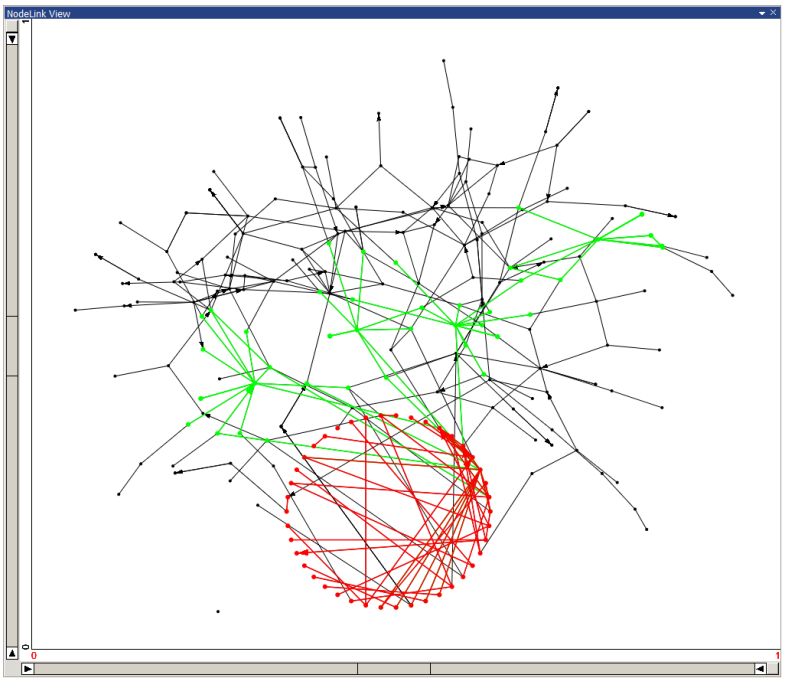
Another option that has been implemented to analyze a subgraph is to re-layout only the currently selected nodes when a new layout is assigned. The nodes will remain in the same bounding box as before but rearrange according to the selected layout, ignoring unselected nodes. This way, it is possible to create a basic layout for the complete graph and then refine it for the subgraphs of interest while keeping the different subgraphs spatially separate. Figure 4.3 shows the effect of relayouting part of a graph. As a drawback, overlaps with unselected nodes or edges can be created since the new layout does not account for them. In these cases, the layout needs to be fixed manually in the current version.

While layout changes are animated, it is still difficult to follow individual nodes during a layout change in all but the smallest networks. To alleviate this issue, individual nodes can be declared keynodes. Keynodes are drawn with a different color, size and shape, making them easy to perceive even during a layout animation. The shape of keynodes can be set by the user to either diamond, pyramid or inverted pyramid. Since all keynodes look alike, only a small number of them should be defined at any time so they can stand out. In our experience, three to five keynodes, preferably in different parts of the visualization, has proven to be a good number.

The standard mapping widget of the system (described in Section 3.2) is also available for both axes of the view. This allows both restricting the displayed range and zooming into the data using a Bifocal Display distortion.



(a) Graph before relayouting the selected nodes.



(b) Graph after relayouting the selected nodes.

Figure 4.3: Relayouting part of a graph.

CHAPTER 5

Parallel Distances

5.1 Introduction

The parts that have been presented so far work well for the analysis of small networks or if an analysis using only derived attributes is sufficient. This section introduces a new visualization that is targeted toward the analysis of large networks that cannot be analyzed efficiently with the implemented node-link view.

The presented view is designed for searching for a user-defined structure of nodes within the network. The VAST challenge 2009[1] presents such a task. In the challenge, four different non-disjunct groups have been created on the graph. The task defines a query that asks for the nodes of each group that are in a specific relationship. This relationship is defined as a node from the first group that has a connection to any node in the second group that has a connection to three different nodes in the third group that each has a connection to a node in the fourth group. For each group, the nodes that comprise the relationship are to be found and returned. The exact definition from the VAST challenge including group definitions and an evaluation of how an analysis with this view can be performed can be found in Chapter 7.

The search for a structure of nodes in a network does not always yield a single correct result. It is also possible to obtain multiple result nodes for a single node of the structure, or to find no matching nodes. In such a case, it is helpful to see how each part of the structure affects the result. The view presented in this chapter is based on Parallel Sets [BKH05] and has been designed for the analysis of the resulting sets of nodes of a query for a structure of nodes.

[1] http://hcil.cs.umd.edu/localphp/hcil/vast/index.php/taskdesc/index/mini_challenge_2

5.2 Definitions

Leaving aside the requirement of having to be connected to multiple nodes in the next set for now, the structure defines g sets $N_i \subseteq N \quad i = 1, \ldots, g$ and poses the task to find the subsets $S_i \subseteq N_i$ defined as:

$$S_i := R_i \cap T_i$$

with R_i and T_i defined as:

$$R_i := \begin{cases} N_1 & i = 1 \\ \{n \in N_i \mid \exists\, m \in R_{i-1} \quad (m,n) \in E\} & i > 1 \end{cases}$$

$$T_i := \begin{cases} \{n \in N_i \mid \exists\, m \in T_{i+1} \quad (n,m) \in E\} & i < g \\ N_g & i = g \end{cases}$$

An equivalent formulation of this is:

$$S_i := \begin{cases} T_1 & i = 1 \\ \{n \in T_i \mid \exists\, m \in S_{i-1} \quad (n,m) \in E\} & i > 1 \end{cases}$$

with T defined like before.

With the requirement of being connected to multiple nodes of the next group, it can formally be defined as $c_i \in \mathbb{N} \quad i = 1, \ldots, g-1$ where each c_i denotes how many nodes from the set S_i need to be connected to a node in T_{i+1} in order for that node to be in S_{i+1}. Including this requirement, the task is to find the sets $S_i \quad i = 1, \ldots, g$ defined as:

$$S_i := \begin{cases} T_1 & i = 1 \\ \{n \in T_i \mid |\{m \in S_{i-1} \mid (m,n) \in E\}| \geq c_{i-1}\} & i > 1 \end{cases}$$

$$T_i := \begin{cases} \{n \in N_i \mid |\{m \in T_{i+1} \mid (n,m) \in E\}| \geq c_i\} & i < g \\ N_g & i = g \end{cases}$$

Going beyond the requirements of the mini-challenge, support for weighted edges is included in the view. Between each set N_1 and N_{i+1}, only edges, with a weight between d^- and d^+ are considered. This puts the formal capabilities of the view as:

$$S_i := \begin{cases} T_1 & i = 1 \\ \{n \in T_i \mid |\{m \in S_{i-1} \mid \exists\, d \in [d^-_{i-1}, d^+_{i-1}] \quad (m,n,d) \in E\}| \geq c_{i-1}\} & i > 1 \end{cases}$$

$$T_i := \begin{cases} \{n \in N_i \mid |\{m \in T_{i+1} \mid \exists\, d \in [d^-_i, d^+_i] \quad (n,m,d) \in E\}| \geq c_i\} & i < g \\ N_g & i = g \end{cases}$$

5.3 Visualization

The network pictured in Figure 5.1 will be used as an example to explain this view. All images contained in this section are based on this network and each step includes an explanation of how the visualization represents this network. Even though nodes can be members of multiple groups, this example only uses disjunct groups. While all figures are explained in the text, the reader might find it helpful to compare the resulting visualization to the network they are made of.

First, we will explain how the view looks when just a single group i of nodes has been added to the view. An example image showing only group 1 of the example network can be found in Figure 5.2. A single axis representing the group can be seen. This axis is divided into two parts. The upper part, displayed in green, represents the nodes that are in the group: N_i. The lower black part represents the nodes that are not in the group: $N \setminus N_i$. Both parts are scaled according to the number of nodes they represent.

For the example network, this axis represents group 1. The group N_1 contains three nodes, which are shown at the top in green. The size of the green part is $|N_1|/|N| = 3/18 = 1/6$ of the size of the complete axis.

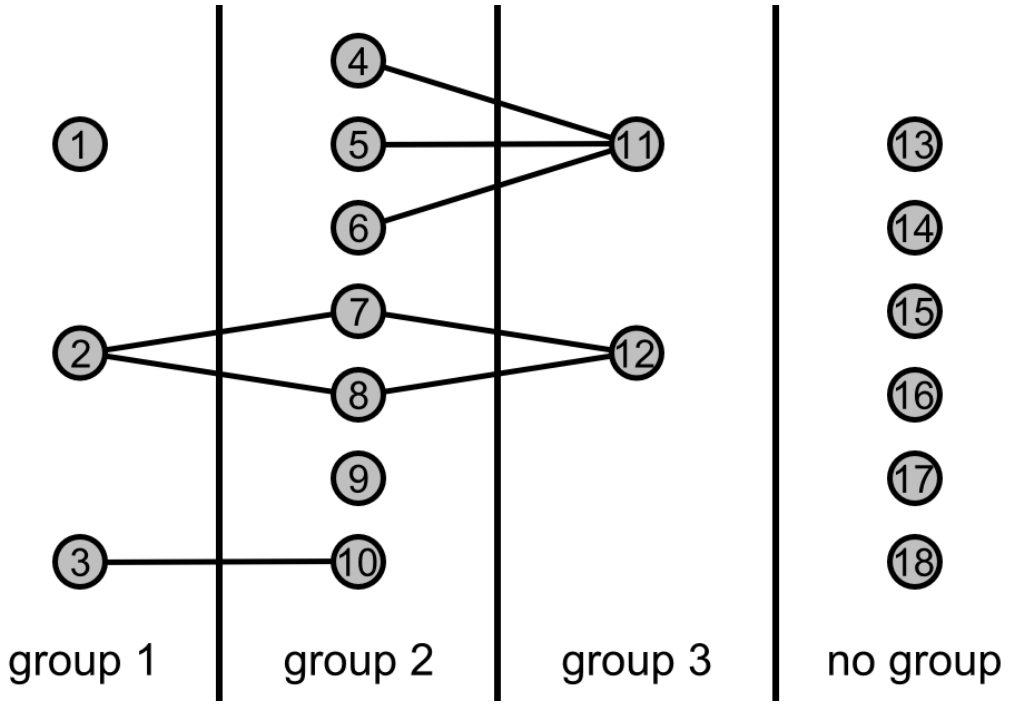


Figure 5.1: The example network visualized by the Parallel Distance views in Chapter 5.

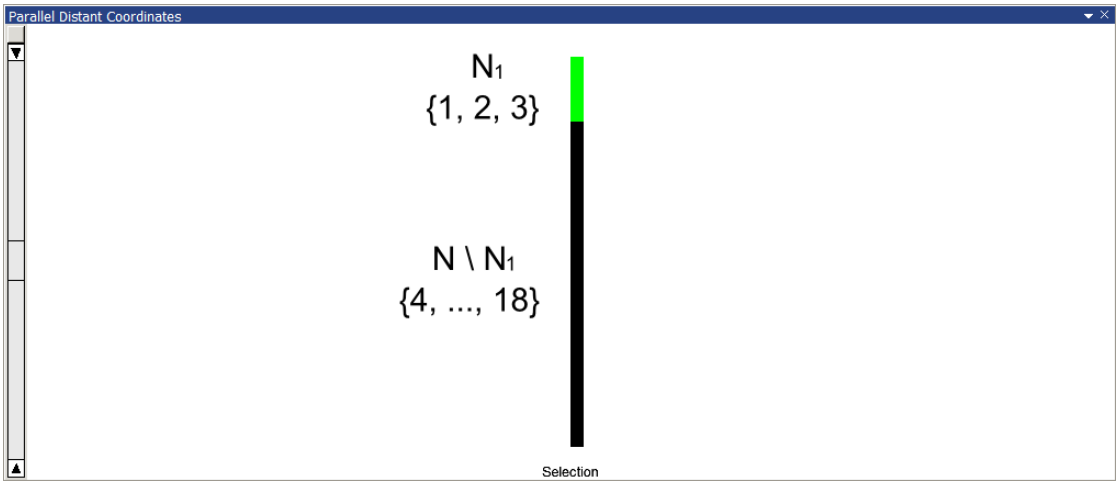


Figure 5.2: Parallel Distance view showing group 1 of the example network.

Now we will explain what happens when a second group is added to the view. Figure 5.3 displays the look of the view with two axes. Two axes is the minimum size for the S_i sets to be displayed, since the result for a single axis would trivially contain the complete set N_i and no query would have to be performed on the groups. Each axis is now divided into 3 parts. While the black part at the bottom, $N \setminus N_i$, still looks the same as before, the top green part which previously displayed the group N_i has now been divided into two different parts. The red part at the top shows how many nodes in the group fulfill the complete query S_i. The green part between $N \setminus N_i$ and S_i represents $N_i \setminus S_i$, the nodes that are in the group but fail the query. This coloring is used to stay consistent with the rest of the system, which uses red for active selections and green for the inactive selections.

In the example network, two of the three nodes in N_1 (nodes 2 and 3) have a connection to a node in group 2 and are therefore in S_1. Of the seven nodes in N_2, three have a connection to a node in group 1: 7, 8 and 10. These nodes make up S_2.

A textual representation of the parameters is displayed between the axes. Between the axes at the top of the view, the number of required connections for a node on the left is displayed: c_i. A node from the group of the left axis, S_i, has to be connected to *at least* this many nodes of the group to the right, S_{i+1}, to be considered a valid result of this query. Below the number of required connections, at the bottom of the view, we display how large the distance between a node from the left group and a node from the right group is allowed to be: d^- to d^+. This can be very useful for networks in which each node has a specific weight that represents a value like physical distance between nodes. If, for example, each node represents a city, this value can represent the distance

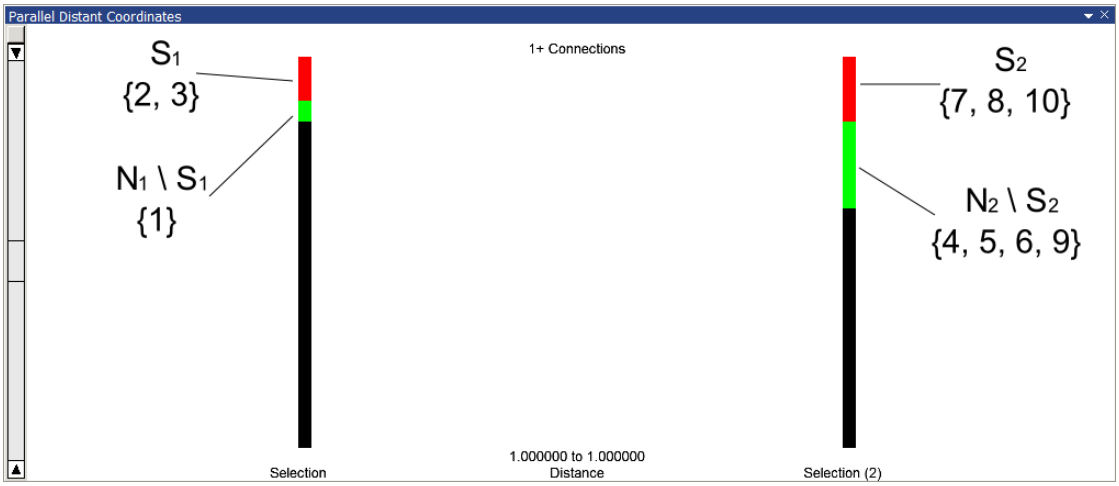


Figure 5.3: Parallel Distance view showing groups 1 and 2 of the example network.

a person has to travel from one city to reach the other one. This way, only cities within a certain distance would be considered for the query.

When a group is selected by clicking on an axis, additional information about this group is displayed. This is in accordance to the visual information seeking mantra of only offering details on demand [Shn96]. After selecting a group, each space between two axes is used to show how the connection requirement filters out the nodes of the selected group that do not fulfill the requirement (see Figure 5.4 for an example). Between the axes are now two connecting segments that represent the nodes in the selected group. One segment always stays at the top while one points downward from the selected axis. The segment that stays at the top represents the nodes that fulfill the requirements of the query while the segment that points toward the bottom represents the nodes that do not fulfill the query. Thus, the height of each segment is equal to the respective height on the axis.

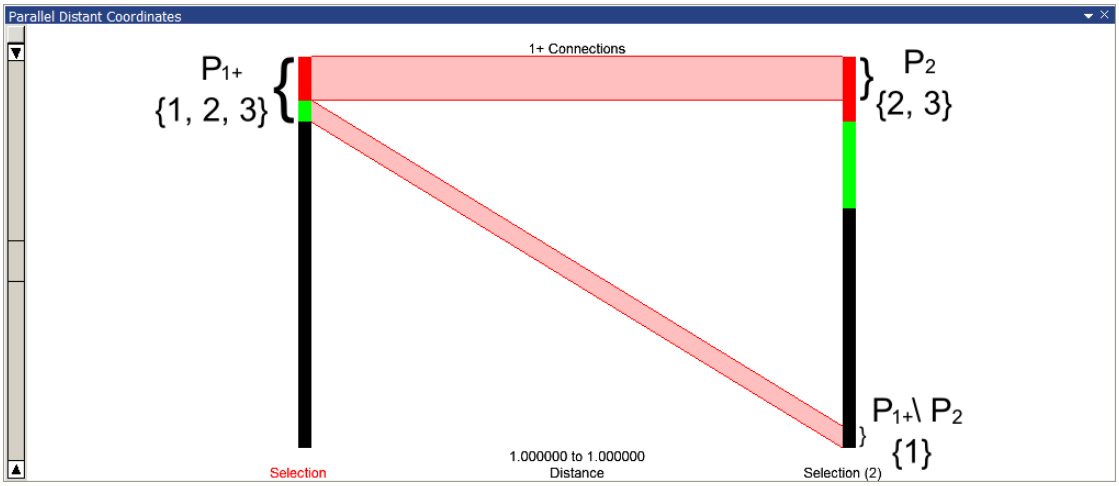
This feature is very useful when the number of nodes in a group that fulfills the query does not match the expectations. In this case, the group can be selected to obtain information about where the disparity between expectation and actual result occurs.

More formal, when group i has been selected, the size of the red segment P_j at group j that is shown near the top is defined as:

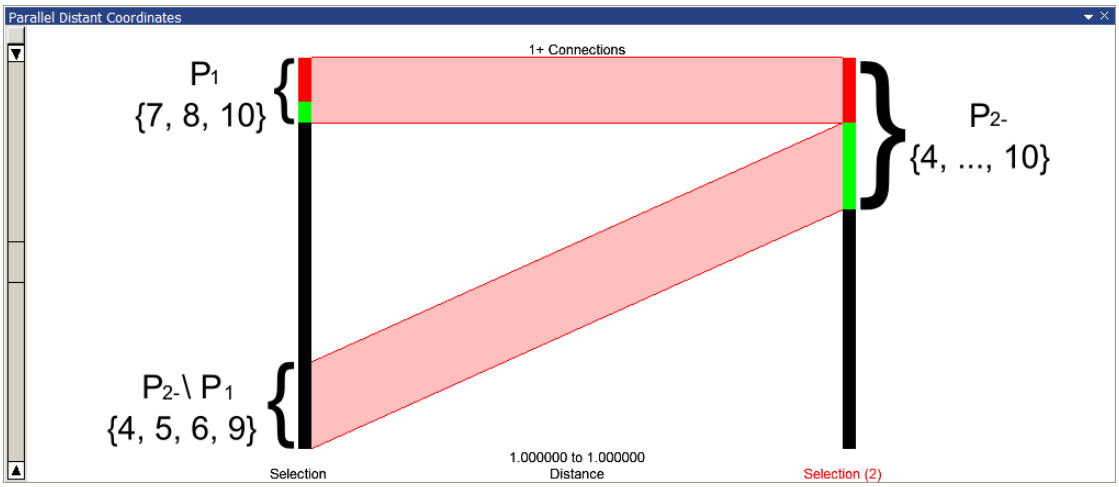
$$P_j := \begin{cases} S_i, \text{ as if the query were only performed for groups } j \dots g & j < i \\ S_i, \text{ as if the query were only performed for groups } 1 \dots j & j > i \end{cases}$$

$P_{i-} := S_i$, as if the query were only performed for groups $i \dots g$

$P_{i+} := S_i$, as if the query were only performed for groups $1 \dots i$



(a) Group 1 selected.

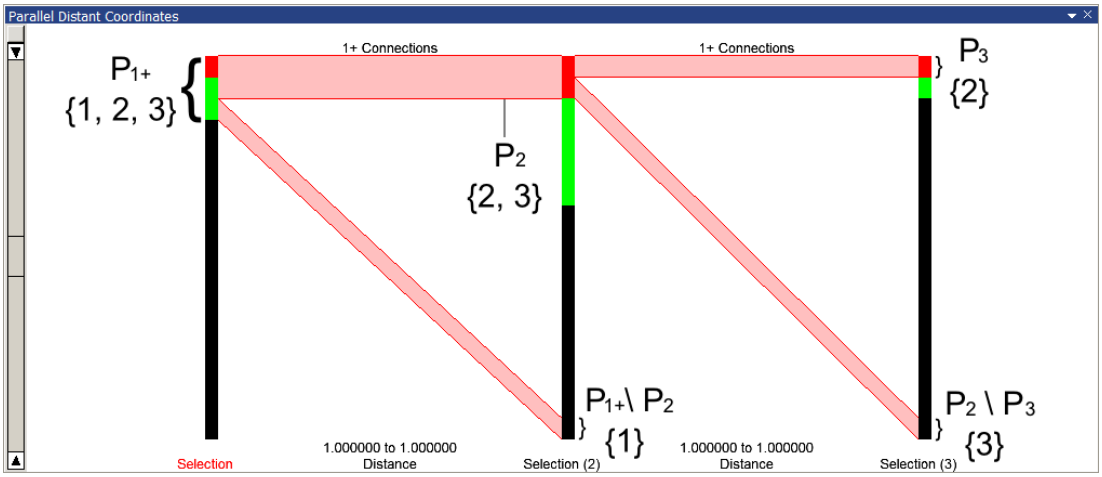


(b) Group 2 selected.

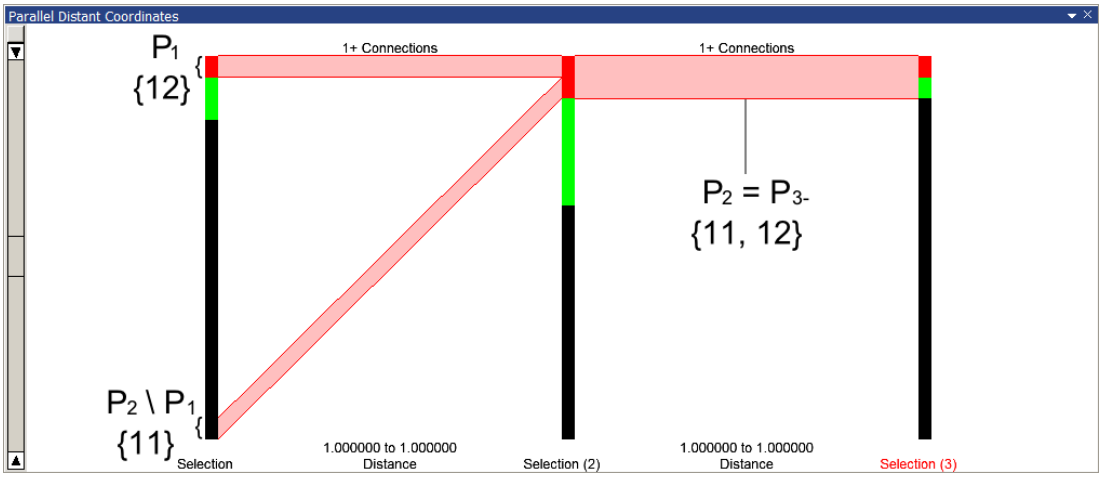
Figure 5.4: Parallel Distance view showing groups 1 and 2 of the example network with one axis selected.

with P_{i-} and P_{i+} being used for group i for segments that connect to $j > i$ and $j < i$, respectively. In addition, the difference between P_j and P_{j+1} is displayed at the bottom near the axis that is farther from S_i. This shows the effect each group has on the final result.

If group 1 is selected in the example network, P_{1+} contains all nodes in N_1. P_2 contains all nodes of P_{1+} that are connected to a node in group two, which are node 2 and 3.



(a) Group 1 selected.



(b) Group 3 selected.

Figure 5.5: Parallel Distance view showing all 3 groups of the example network with one of the outer axes selected.

When group 2 is selected, P_{2-} contains all nodes in N_2. P_1 contains all nodes of P_{2-} that are connected to a node in group 1, which are nodes 7, 8 and 10.

Figure 5.5 shows how the view looks when the third group has been added. To fulfill the query, a node from the left group now has to be connected to the specified number of nodes from the middle group that in turn have to be connected to the specified number of nodes from the right group. Let us now assume that the user has selected either the left or the right axis. The connecting segments adjacent to the selected axis only take the selected axis and the middle axis into account. The connecting segments between

the two unselected axes show the number of nodes that fulfill the whole query and the nodes that fulfill only the first part.The segment representing the nodes that fulfill the complete query is displayed at the top while the segment that represents the nodes that fulfill only the first part points toward the bottom. The size of the segment pointing toward the bottom thus equals the difference between the two segments at the top. This allows the user to see which part of the selected group fails the requirements at each part of the query.

When group 1 is selected in the example network, P_{1+} and P_2 are defined the same as when there were only two groups in the network. P_3 contains only those nodes of P_2 which are connected to a node in group 2 which is in turn connected to a node in group 3. Node 3 does not fulfill this since the only node in group 2 it is connected to, node 10, is not connected to any node in group 3. Node 2 does fulfill this since both nodes 7 and 8 are connected to node 12, which is in group 3. Therefore, P_3 consists only of node 2.

When group 3 is selected, P_{3-} contains N_3: nodes 11 and 12. P_2 also contains nodes 11 and 12 since both of them are connected to nodes in group 2. P_1 consists only of node 12. Node 12 is connected to nodes 7 and 8 of group 2. Those nodes have a connection to a node in group 1: node 2. For node 11 however, none of the three different nodes in group 2 it is connected to has a connection to any node in group 1.

The connections look a little different if the user selects the group in the middle (see Figure 5.6). Each connecting segment has been further split. The red segment at the very top represents the number of nodes that fulfill the complete query. An additional gray segment pointing to the top represents nodes that fulfill this connection's requirements but fail the requirements between the middle axis and the opposite axis. The gray segment pointing toward the bottom stands for nodes that fulfill neither this requirement nor the one from the other side. The second red segment, between the first red segment and the gray segment pointing to the top, that points toward the bottom represents the nodes that fail the requirements between these axes but fulfill the requirements between the other two axes. Thus the red segments represent the nodes that fulfill the requirements of the *other* direction while the gray segments fail the other direction. This can be used to check how many nodes *could* fulfill this part of the query if the other part were changed or removed.

A formal definition of the size of the gray segment Q_j at group j is similar to P_j:

$$Q_j := \begin{cases} \{S_i, \text{as if the query were only performed for groups } j \dots i\} \setminus P_j & j < i \\ \{S_i, \text{as if the query were only performed for groups } i \dots j\} \setminus P_j & j > i \end{cases}$$
$$Q_{i-} := N_i \setminus P_{j-}$$
$$Q_{i+} := N_i \setminus P_{j+}$$

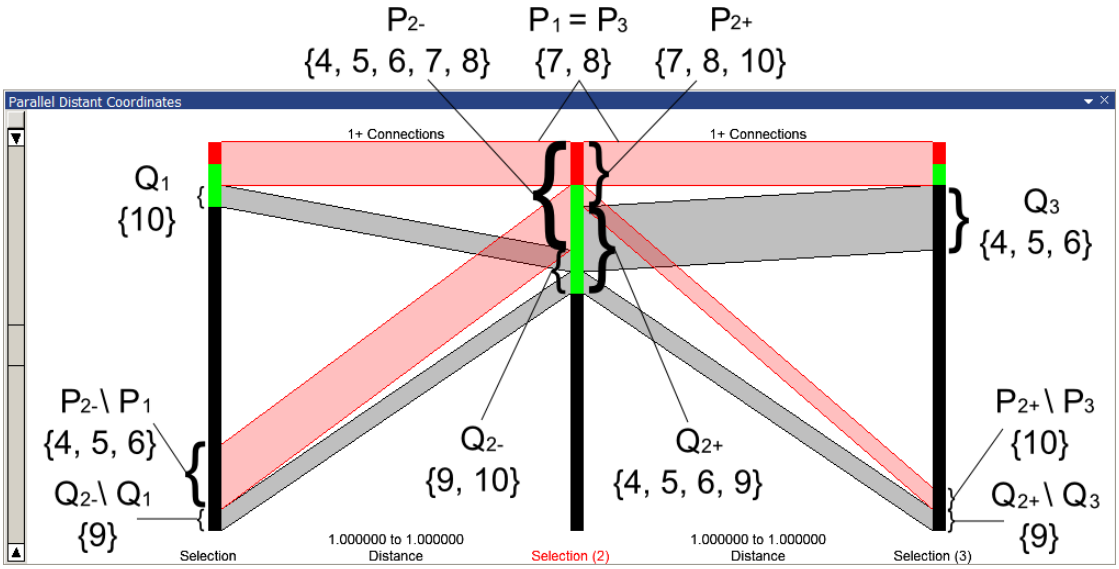


Figure 5.6: Parallel Distance view showing all 3 groups of the example network with the central axis (group 2) selected.

Once again, the difference between P_j and P_{j+1} is shown at the bottom near the axis that is farther from S_i. This shows the effect each group would have on the final result if the query would only consist of the groups i to j.

For the example network, P_{2+} contains all nodes that are connected to at least one node in group 1, just like S_2 before group 3 had been added to the view. P_{2-} contains all nodes that are connected to a node in group 3: nodes 4, 5, 6, 7 and 8. Q_{2-} and Q_{2+} contain all nodes of N_2 that are not in P_{2-} and P_{2+}, respectively. P_1 contains all nodes of P_{2-} that are connected to a node in group 1 and P_3 contains all nodes of P_{2+} that are connected to a node in group 3. Both therefore contain those nodes that are connected to nodes in group 1 and 3 and therefore contain the same nodes as S_2. Q_1 then contains all nodes that are connected to a node in group 1 but not connected to group 3 and Q_3 contains nodes that are connected to group 3 but have no direct connection to group 1.

This view is not limited to three axes but can show an arbitrary number of groups. All definitions in this chapter can be used without any modification for larger queries. An example using more than three axes can be found in Chapter 7.

The visualization scales well with the number of groups displayed. A minimal distance between axes is enforced. When the space between two groups would shrink too much due to the number of groups, we display a scrollbar and keep the distance at the minimum. It is then possible to scan over a query of arbitrary size.

5.4 Interaction

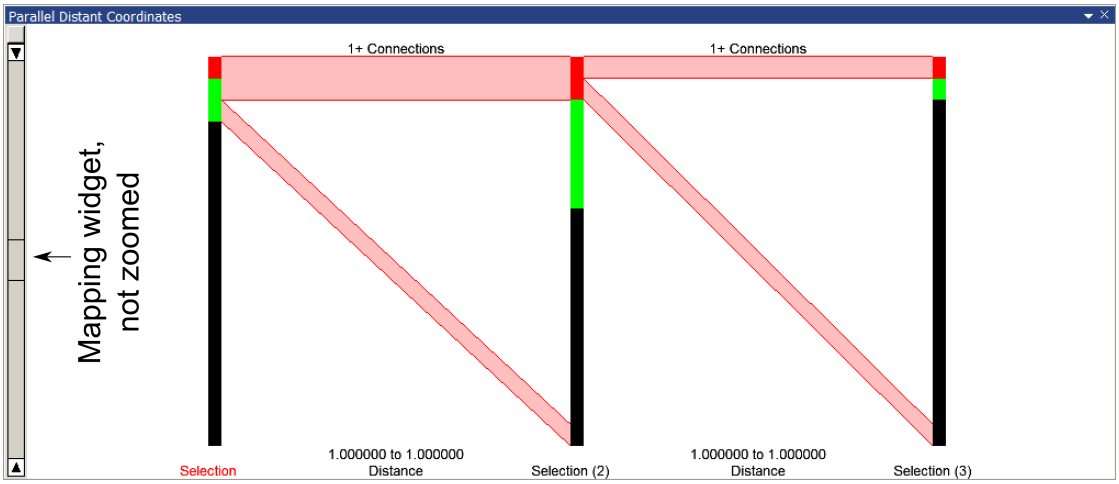
Since the visualization presented in this chapter is designed to assist the user in exploring the dataset using other views in conjunction with this one, interaction is very important. First, we will explain how a user can interact with our view, then explain the offered interactions of our view with a linked-view system.

The user may select an axis by clicking on it. Selecting an axis updates all connecting segments to reflect the number of filtered and non-filtered nodes of the newly selected group as described in Section 5.3.

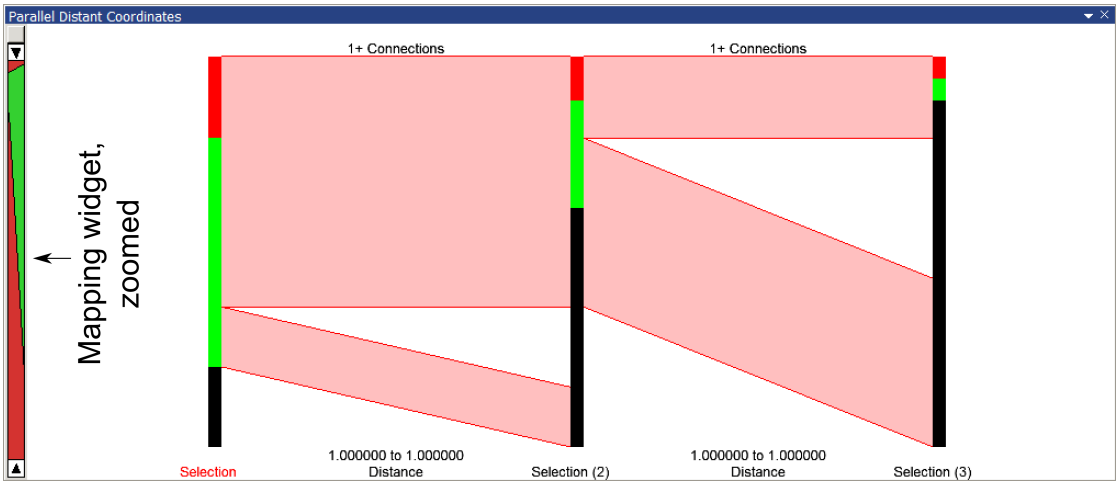
It is also necessary to give the user means to adjust the constraints of the query. Both the minimal number of required connections and the range of allowed distances between nodes needs to be adjustable by the user.

The view allows the user to zoom into any region he or she is interested in. For this, the system wide zooming widget is used to provide a Bifocal Display distortion, the exact functionality of which has already been described in Section 3.2. The distortion only affects the currently selected axis. The reason for this is that the number of nodes selected in different groups can vary greatly. This makes it impractical to zoom all axes with the same function. The disadvantage of this approach is that the user can no longer discern the sizes of the groups with a single glance when zooming is utilized. Zoom also affects the connecting segments that originate from this axis since they represent the same nodes. This accentuates the interactive nature of our view by allowing the user to study the effects of this part of the query in detail. A zoom operation is not reset after the user selects a new axis. An image showing the effect of zooming is depicted in Figure 5.7.

Interaction with other views in the system is also a feature of the Parallel Distance view. The user can select any subset of nodes from any axis i: those that pass the query (S_i), those that do not pass the query ($N_i \setminus S_i$) and those that are not in the group ($N \setminus N_i$). A selection allows the user to transfer the results of a query from this view to other linked views. It can be performed with a single click of a mouse button over an axis. All nodes of the set under the mouse cursor are then selected, highlighting them in all linked views. The framework this view was implemented in supports composite queries based on set operations. This way, selecting the results of the query from all groups can be performed by choosing the unison operation and selecting the desired subset of each group separately.



(a)



(b)

Figure 5.7: A Parallel Distance view with three axes, the first of which is selected. Image (b) results from image (a) after modifying the mapping of the first axis.

Finally, the Peek Brush (see Section 3.2) is also supported. Hovering over any visualized subset (S_i, $N_i \setminus S_i$ or $N \setminus N_i$) of an axis contained in the view selects all nodes of the set using the Peek Brush selection. Entities selected either by a Peek Brush or a regular selection in another view are not highlighted in any way in the Parallel Distance view. While such a visual feedback would technically be possible, highlighting up to three different levels of selection (Peek Brush, active selection and inactive selections) simultaneously would make this visualization hard to read.

5.5 Summary

This chapter presented a novel view for systems that utilize multiple linked views targeted at users that are interested in searching for complex relations within a multivariate network using a multiple linked view environment. Our approach supports the study of queries concerning relationships between groups in a network. The view shows for each group how many nodes it consists of and how many of those pass the query. To further investigate the results of such queries, other views can be used. A useful combination with our view is a node-link view. This combination has proven to be effective for both small networks and cases where the user already has at least a basic idea of the kind of relations he is interested in investigating. An example of how such an analysis can be performed is given in Chapter 7.

CHAPTER 6

Implementation

6.1 Overview

This section contains some implementation details about the work done for this thesis. Section 6.2 describes the importer, which has been mostly ignored for now. Section 6.4 covers the feature-based attributes, including the channel computations for node attributes and the view for singular attributes. Section 6.5 contains details about the node-link view. And finally, Section 6.6 deals with the Parallel Distance view.

6.2 Importer

Before a network can be analyzed, it needs to be loaded into the system. Since no importer could import edge data before this thesis, it was necessary to create an importer plugin for the system.

There are two ways in which the importer can be used. The first is to use it to import both node and edge data. However, only a handful of node attributes is supported by the importer. Section 6.3 describes the file format, including the supported attributes.

The second way to use the importer is to enrich a multivariate dataset by adding edge information. For this, a dataset is imported using a different importer before adding the edge data from another file. The requirement for this is that the file containing the edge data can only reference nodes that have been imported from the original dataset. Since this mapping is based only on an enumeration of the nodes, this can fail when the node dataset that has been imported has fewer nodes than the edge dataset being imported.

6.3 Network File Format

The created importer can import ASCII-based Pajek [BM98] network files (.net). Listing 2 contains an example of a very simple network using this format.

Pajek networks start by defining the nodes. The first line contains `*Vertices` followed by the number of nodes. Each of the following lines defines the values of a single node. Each node definition starts with the number of the node, followed by its name enclosed by quotation marks. All remaining attributes are optional. If the line contains further attributes, the next values are the x- and y-position Pajek should use. Further attributes are not imported into VISPLORE and contain values such as the shape or color of the node Pajek should draw.

The second part contains the edges. If it is prefaced by `*Edges`, each line in this part contains one line, with the first two values being the numbers of the nodes of the edge. The third number is optional and indicates the weight of the edge.

If the edge part is prefaced by `*Edgeslist`, one line contains all edges for a single node. The line starts with the number of the node the edges are being defined for with all following numbers designating the nodes it has connections to. This mode cannot contain edge weights.

Instead of `*Edges` or `*Edgeslist`, `*Arcs` or `*Arcslist` might be encountered. In this case, the edges are directed, with the first node on each line being the source and the remaining node(s) being the target.

Listing 2 Sample network file

```
*Vertices     5
  1 "A"    0.0000 0.5000
  2 "B"    0.0000 0.0000
  3 "C"    0.0000 1.0000
  4 "D"    0.5000 0.5000
  5 "E"    1.0000 0.5000
*Arcs
*Edges
  1 2   4
  1 3   9
  1 4   7
  4 5   2
```

6.4 Feature-based Attributes

6.4.1 Global Graph Properties

The view for displaying graph values uses a single visualization thread for the calculation of the attributes. The validity of the value of each attribute/layer combination is stored and each layer change invalidates all values of the layer. The displayed values are only updated after all values have been brought up to date.

Available computations:

Number of nodes is provided by the system.

Number of edges counts edges for which both incident nodes are in the layer being calculated.

Number of connected components uses the calculation provided by the Boost Graph Library [SLL02]. Only the nodes and edges in the layer being calculated are passed to the Boost computation.

6.4.2 Node Properties

The scheduling of computed node attributes is handled by the system. It uses a dedicated shared thread for the computation of all computed channels.

For the attributes 'Distance from selection', 'Betweenness centrality', 'Component enumeration' and 'Articulation points', computations provided by the Boost Graph Library [SLL02] are used. 'Current selection' represents selections using float values. The value 1 is written for selected values and the value 0 for unselected ones. This way, uncertainty could be represented by using values between 0 and 1 in the future.

6.5 Node-Link View

The node-link view uses layered drawing with cached images of lower layers, the Peek Brush as described in Chapter 3 and utilizes its own background thread.

The Peek Brush is accelerated by a grid in which each grid element stores all adjacent nodes. This way, only nodes referenced in grid elements that are close to the mouse cursor need to be checked.

For the transition between layouts, the view stores both the previous and target position in separate arrays. During the animation, the position of each node is calculated by using linear interpolation between the positions stored in the arrays. When the animation finishes or a new layout has been calculated, the current position of each node is stored in the array containing the previous positions.

6.6 Parallel Distances

The Parallel Distance view evaluates the query by doing two passes over the groups. The first pass calculates T_i for each group, starting from the last group. The second pass calculates S_i for each group, starting from the first group.

After the main query has finished, $Q_j \quad j = 2 \ldots g-1$ is calculated for each selectable group $i = 2 \ldots g-1$. While this additional computation is running, the view can already be used albeit no segments between axes are shown when an axis has been selected.

The view also supports the Peek Brush. Should the user modify a parameter of the query, the computation is automatically aborted and restarted.

CHAPTER 7

Results

7.1 Usage Example

This section demonstrates how the presented approaches can be utilized to analyze a dataset. The task is to search for structure A from the second mini-challenge of the VAST challenge 2009[1]. This challenge asks the user to search the given dataset containing 6000 nodes for an employee at an embassy who communicates with exactly three 'handlers' who pass on his information to a common middle man who then forwards it to the mastermind of a criminal organization called 'Fearless Leader' . A visual representation of the target subnetwork is provided in Figure 7.1. It is further known that the employee has about 40 contacts, each handler has 30–40 contacts, the middle man has 4–5 contacts and the fearless leader has more then 100 contacts including international ones. The handlers are not allowed to communicate with each other.

After loading the data into the system, groups for the different roles are defined. A derived channel that contains the number of edges each node has is created and assigned to a histogram (see Figure 7.2). It can be seen that the node degrees follow a power law distribution, making this a scale-free network (see Section 1.2.2.3). By selecting a range of connections in a histogram, it is possible to define the groups. To define the employee group, all nodes with 38–42 edges are selected to incorporate the expressed uncertainty. The handler group is set by selecting the nodes with 30–40 edges. Since these groups overlap, some of the nodes could be either employee or handler. The middle man group is created as nodes with 4–5 edges and all users with at least 100 contacts comprise the fearless leader group. Each selection is stored as a derived channel so it can be assigned to a view.

[1] http://hcil.cs.umd.edu/localphp/hcil/vast/index.php/taskdesc/index/mini_challenge_2

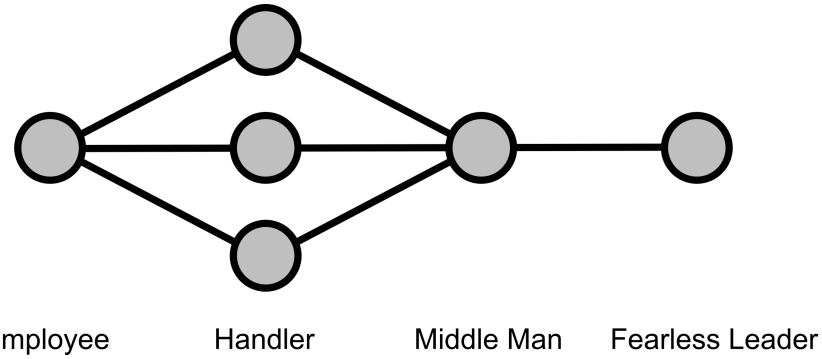


Figure 7.1: The target subnetwork that needs to be found for the VAST mini-challenge.

Next, a Parallel Distance view is opened and the created groups are added in the order employee–handler–middle man–fearless leader (see Figure 7.3). The next step is to modify the parameters of the connections between the groups. Since a direct connection is the default setting and just the employee needs more than one connected node, the number of connections between the first two axes (employees to handlers) is set to three.

Figure 7.4 shows the structure that can be found using the created Parallel Distance view. The difference to the target structure stems from the fact that while each handler needs to be connected to a middle man, it is not checked if all handlers are connected to the *same* middle man. To filter out nodes that fit this structure but do not

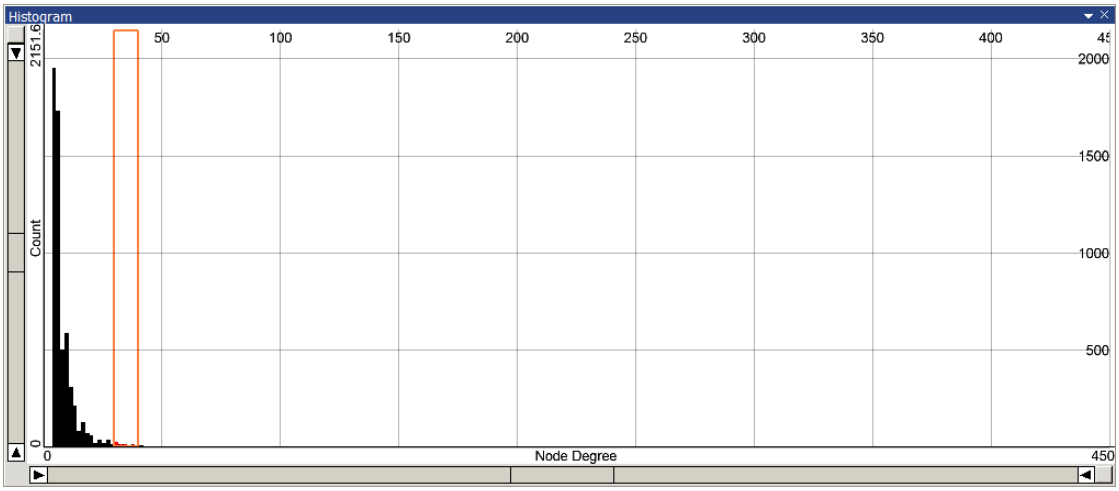


Figure 7.2: A derived channel containing the number of connections is assigned to a histogram and used to define the groups.

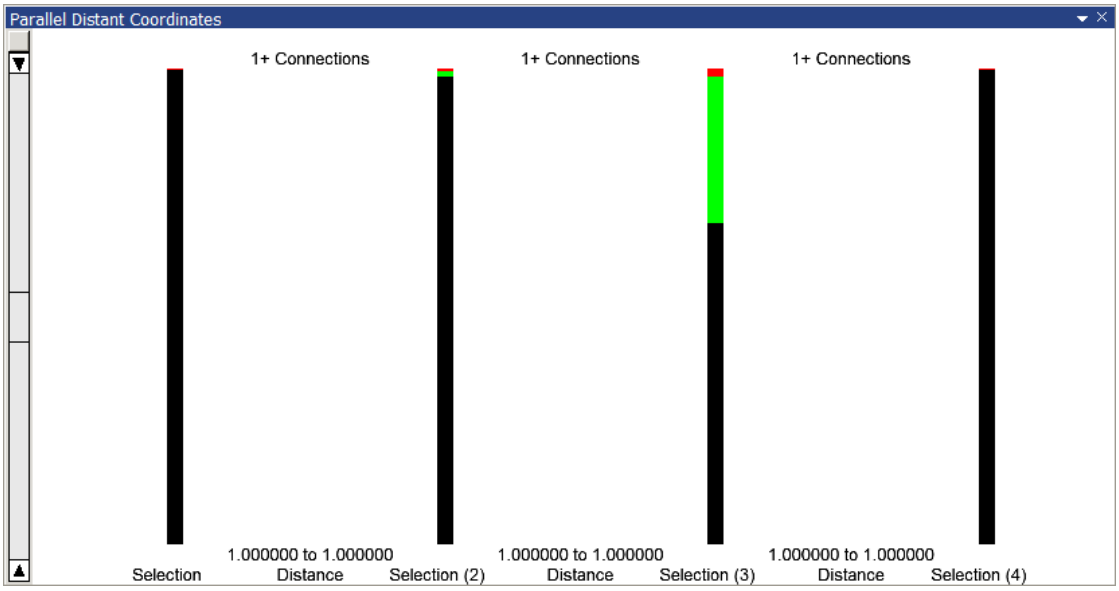


Figure 7.3: Parallel Distance view with the four groups defined in the VAST mini-challenge.

fulfill the requirements of the mini-challenge, another Parallel Distance view is opened and the groups are assigned in reverse order: fearless leader—middle man—handler—employee. This time, the required number of connections between the second and third axis (middle man to handler) is set to three. Figure 7.5 shows the structure that can be found using this Parallel Distance view.

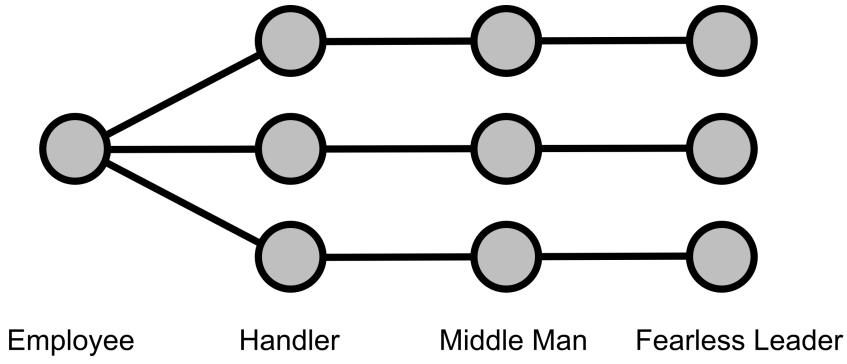


Figure 7.4: The structure that can be found using the Parallel Distance view in Figure 7.3.

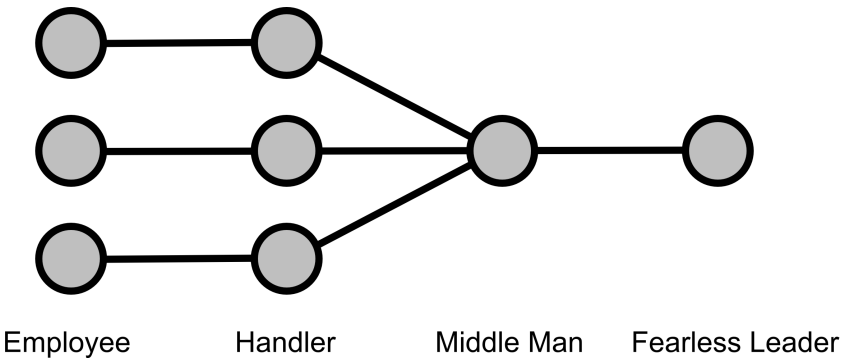


Figure 7.5: The structure that can be found by a Parallel Distance view to which the groups have been added in reverse order.

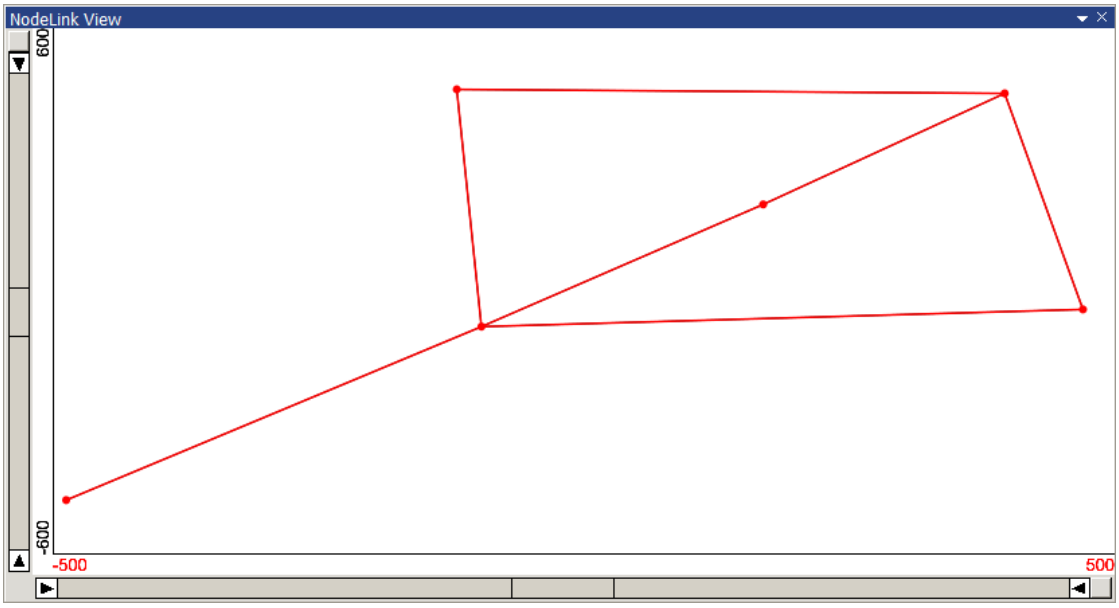


Figure 7.6: The possible results visualized in a node-link view. Since only the six visualized nodes have been found, they form the correct result for the second VAST mini-challenge.

An intersection between corresponding groups (employee–employee, handler–handler, middle man–middle man and fearless leader–fearless leader) from the two parallel distance views now yields the nodes that fulfill the requirements of both views. While an intersection between the results from the two queries is a good approximation of the target structure, the resulting nodes do not necessarily fulfill the complete query of the mini-challenge. To verify the results, a node-link view is added to the system to visu-

alize the results. The resulting node-link view can be seen in Figure 7.6. Finally, the results have to be verified to fit the target structure. It can be verified that the node at the top right lies in the defined employee group, the three connected nodes lie in the defined handler group, the node those three are connected to lies in the defined middle man group and the node at the bottom left lies in the defined fearless leader group. Since no other nodes have been identified as possible results, it can be concluded that these nodes form the criminal network that has been suspected to exist in the data.

7.2 Extended Usage Example

The second usage scenario is based on the same VAST mini-challenge as the previous one. However, the assumptions concerning uncertainty have been modified so that the employee group includes all persons with 35–45 contacts and the handler group includes persons with 27–43 contacts. The middle man group as well as the fearless leader group remain unchanged and thus include persons with 4–5 and more than 100 contacts respectively.

Like in Section 7.1, the groups are defined using a histogram and assigned to two Parallel Distance views, one starting from the employee suspects and one starting from the fearless leader suspects. The parameters of the Parallel Distance views are set to the same values as before. An intersection of the possible results from the corresponding groups of each view is assigned to the node-link view. This time the node-link view contains additional nodes, as seen in Figure 7.7.

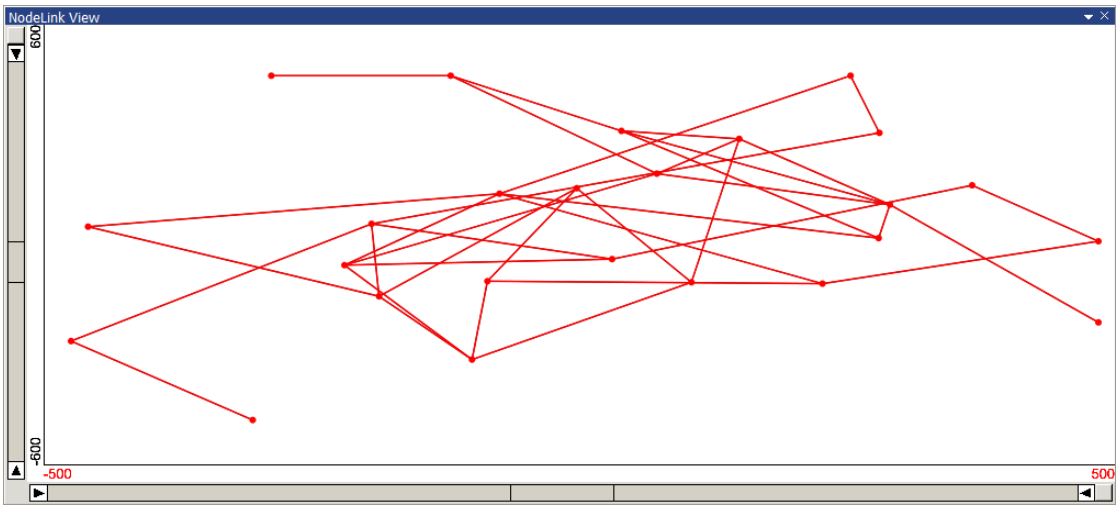
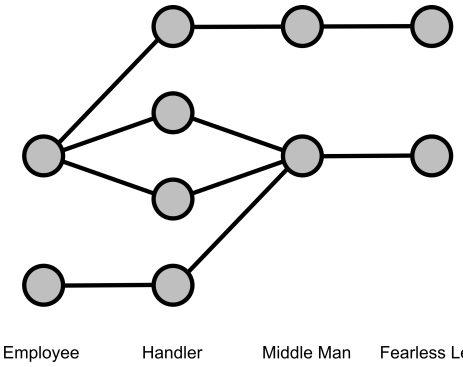
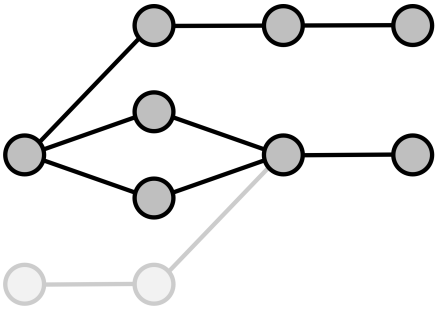


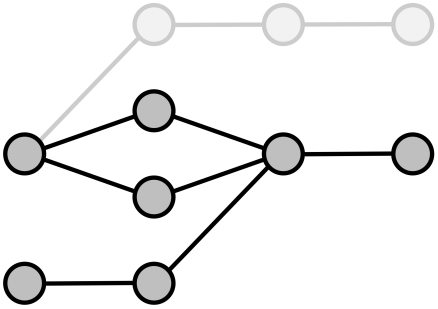
Figure 7.7: The possible results of the query with increased uncertainty visualized in a node-link view.



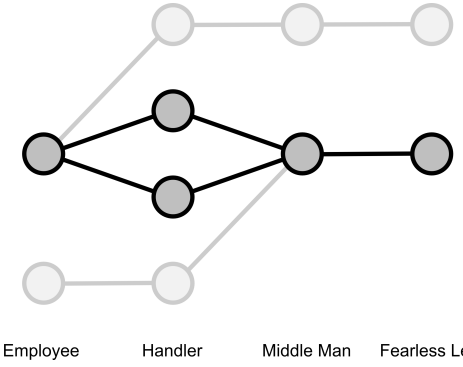
(a) The complete example network.



(b) Nodes that fulfill the query starting from the employee (see Figure 7.4).



(c) Nodes that fulfill the query starting from the fearless leader (see Figure 7.5).



(d) Nodes that fulfill both queries. There are only two handlers connecting the employee to the middle man instead of the required three.

Figure 7.8: An example of a network that contains nodes that fulfill both queries but do not fulfill the requirements of the mini-challenge.

So how can nodes that do not fulfill the query remain in the results? By modeling the target structure (Figure 7.1) as an intersection of two modified structures (Figure 7.4 and Figure 7.5), nodes that have been filtered from the results in one query could still be used as a valid connection in another query. Consider the example nodes pictured in Figure 7.8(a). Figure 7.8(b) and Figure 7.8(c) show the nodes that pass the main query starting from the employee suspects (see Figure 7.4) and the nodes that pass the reverse query starting from the fearless leader suspects (see Figure 7.5), respectively. Figure 7.8(d) shows the nodes that pass both queries. These nodes would now have been assigned to the node link view, even though there are only two handlers connecting the employee to the middle man instead of the required three. The problem is that both views compute their result independently so nodes that fail in one of the Parallel Distance views are not removed from the groups in the other Parallel Distance view.

To obtain a clear result, the nodes that have passed both queries have to be checked manually to remove such fragments. Additionally, it needs to be confirmed that the result fulfills the constraints that have not been modeled. The nodes are manually rearranged in a way that places possible results together while making it easy to identify which group a node belongs to (see Figure 7.9). Note that the employee and handler groups overlap and a part of the nodes in each group also belongs to the other group. For the images, nodes that do not have a connection to a possible middle man have been placed in the area labeled 'employee' since handlers need to have a connection to a middle man. Nodes that can only be employees due to the number of connections have also been placed in this area. Nodes that can be either employee or handler are in the area

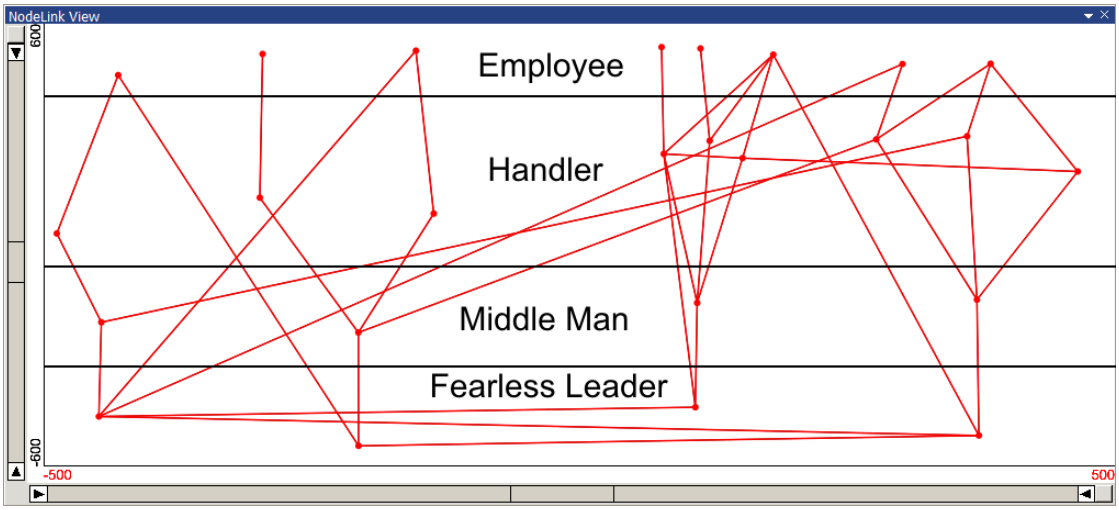


Figure 7.9: The possible results of the query with increased uncertainty in a node-link view arranged to allow faster recognition of possible results.

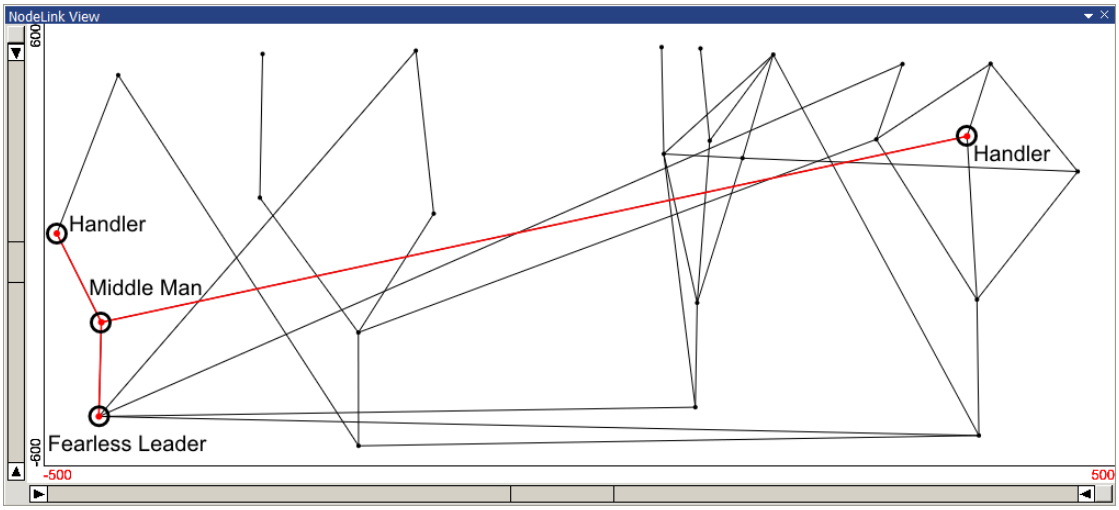


Figure 7.10: Checking the validity of the leftmost nodes.

labeled ‘handler’. The review of the remaining nodes is done starting from the possible fearless leaders.

The single middle man candidate the first possible fearless leader is connected to only has two nodes that might be handlers connected to it, as can be seen in Figure 7.10. Since three handlers are necessary, it can be concluded that this candidate is not the real fearless leader.

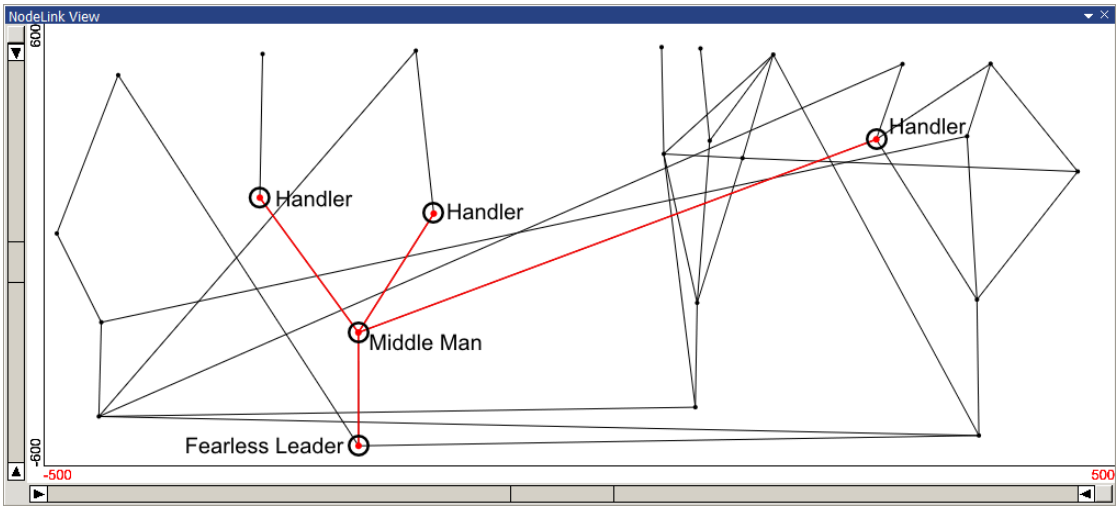


Figure 7.11: Checking the validity of the nodes slightly to the left.

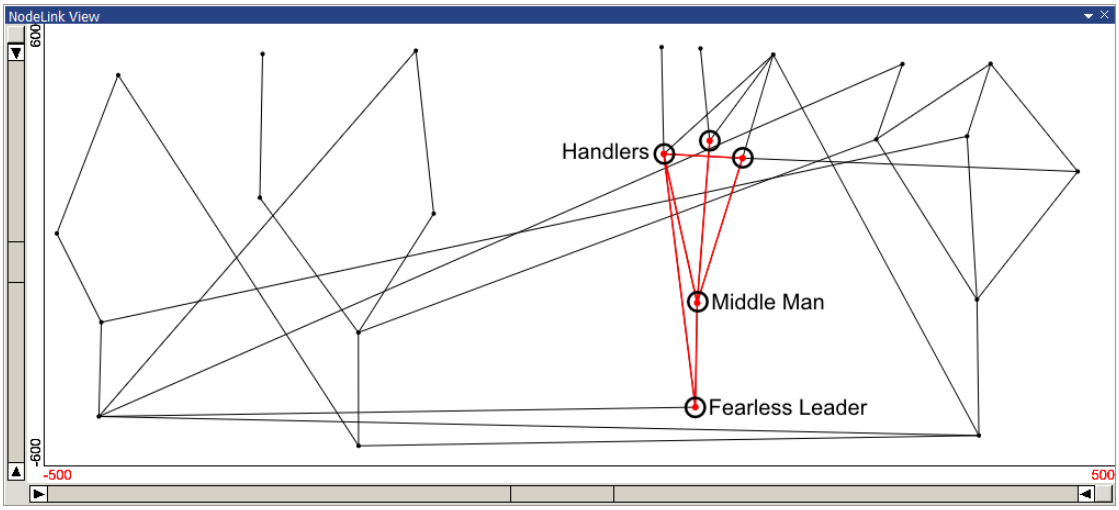


Figure 7.12: Checking the validity of the nodes slightly to the right.

The second fearless leader candidate (see Figure 7.11) has the required connections to a middle man which has three possible handlers. However, the three handlers do not share any nodes that could be the requested employee. Therefore, the second candidate cannot be the real fearless leader.

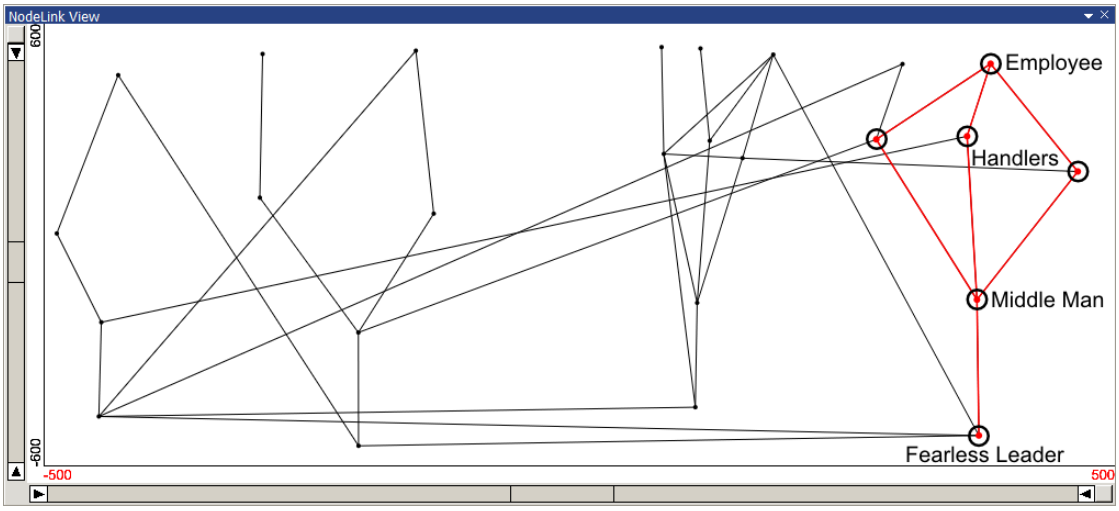


Figure 7.13: Checking the validity of the rightmost nodes.

Figure 7.12 shows the network of the third fearless leader candidate. While the corresponding middle man is connected to three handlers, two of those handlers share a connection. Since the problem forbids connections between handlers, it can be concluded that this candidate is not the real fearless leader.

The final candidate is the same one that has been identified in Section 7.1. Figure 7.13 shows the required connection to a middle man which is connected to three handlers (that do not communicate with each other) that attempt to extract information from a single employee. This candidate can therefore be surmised to be the leader of the organization.

Like in Section 7.1 only a single candidate for the suspected fearless leader remains. It is notable that despite a considerable increase in the size (~40 → 35–40, 30–40 → 27–43, 4-5 → 4–5 and 100+ → 100+) of some of the initial groups, the previous set of nodes is still the only combination of nodes that fulfills all requirements.

CHAPTER 8

Future Work

Although the approaches described in this thesis work very well, there are numerous ways in which they could be extended. The topics are listed in the same order as they appear in the thesis.

For feature-based network visualization, it is obviously possible to add additional computations for both node and global graph properties. A useful extension would be the ability to restrict the computations to arbitrary edge selections. However, this is currently not possible due to limitations placed on the edge table by the system. While it is currently possible to restrict computations to selected nodes and edges for which both nodes are selected, restricting allowed edges to an arbitrary selection on an edge table with dozens of attributes rather than the current single one offers new possibilities.

One feature of the node-link view that does not yet reach its full potential is layouting a selection while keeping the current layout of the remaining nodes in the background. Even though this feature works well in some cases, nodes and edges that are assigned a new position can obstruct background nodes. While this is a reasonable tradeoff to see the background nodes in large graphs, additional work needs to be done to prevent this in smaller graphs. Running a background thread that incrementally adjusts the current layout while starting with a cheap approximation (similar to the work of McGuffin and Jurisica [MJ09]) would be another direction in which the node-link view could be developed.

Clustering is a feature that could drastically improve the node-link view, the usability of the node property computations and potentially improve the Parallel Distance view. By merging multiple nodes into a cluster, both visual overhead and computation time could be reduced. Currently, clustering can only be performed in a preprocessing step

before importing the data. Having this information in the system would make it possible to switch between the original nodes and the clusters on the fly.

For the Parallel Distance view, more complex structures like being connected to multiple nodes that are all connected to the *same* node (for example, the structure shown in Figure 7.1) could be supported. Such structures could be defined using an additional interface for specifying the target structure.

CHAPTER 9

Summary

This thesis explores the analysis of multivariate networks.After explaining the terms of the field (Chapter 1) and giving examples of how graphs are currently being analyzed (Chapter 2), this thesis familiarizes readers with the underlying system it builds upon. VISPLORE is a linked view system that uses a dedicated thread for each visualization that may be preempted by user interaction.

To take advantage of the system, the thesis introduces a type of feature-based network analysis that centers around analyzing data derived from the structure of the network in traditional visualizations for multivariate data. Data with a different value for each node is simply added to the dataset as another data dimension. Values that describe the structure of the complete network or the complete selected subnetwork are visualized using a new visualization.

Chapter 4 also presents a node-link view to show the structure of the network. Layout changes in the node-link view are animated and features for relayouting the selected set of nodes or showing only the selected nodes are available.

Chapter 5 introduces a new visualization that supports the study of queries concerning relationships between different groups of nodes in a network. It displays each group as a vertical axis and visually separates the nodes in the group that fulfill the query from those that do not. This allows a user performing a search for a multi-leveled relationship within a graph to see how each part of the query affects the resulting set of nodes. By selecting nodes that fulfill the query or nodes that do not fulfill the query from the axes, the view can be utilized in a linked-view environment to supplement well-established visualizations.

Chapter 6 contains some additional information about the implementation of the plugins. It also presents the importer implemented during this thesis. To evaluate the contributions of the thesis, Chapter 7 shows how they can be used to solve the second mini-challenge of the VAST challenge 2009.

CHAPTER 10

Conclusion

This thesis presents three main contributions to the field of multivariate graph analysis. The contributions are listed in the same order as they appear in the thesis.

The first contribution is feature-based network visualization, as defined in this thesis. While techniques for the analysis of multivariate data have been used for graph analysis before, making structural data of the graph available in a multiple view system gives new opportunities to study the correlation between different structural values or between structural and data values of the nodes.

The implemented node-link diagram has been created for use in a multi-view environment and contains extensions for studying selected nodes. It also shows transitions for structural changes to preserve the mental model of the user.

The Parallel Distance view combines existing methods to create a novel view for searching nodes that form a specific structure. This makes it an efficient tool when a structure that can be expressed by the view needs to be searched but requires prior knowledge (for example from an initial analysis) of the graph.

The combination of these contributions creates a system that offers users a lot of possible ways to analyze a network. The presented techniques offer additional ways to analyze multivariate graphs, especially thanks to their ease of being combined with other techniques due to being developed for a multiple view system.

Bibliography

[ACJM03] Auber D., Chiricota Y., Jourdan F., Melançon G.: Multiscale visualization of small world networks. In *INFOVIS* (2003), pp. 75–81.

[Ada06] Adar E.: GUESS: a language and interface for graph exploration. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems* (New York, NY, USA, 2006), ACM, pp. 791–800.

[AES05] Amar R., Eagan J., Stasko J.: Low-level components of analytic activity in information visualization. In *Proceedings of the 2005 IEEE Symposium on Information Visualization* (Washington, DC, USA, 2005), IEEE Computer Society, p. 15.

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

[Aub03] Auber D.: Tulip : A huge graph visualisation framework. In *Graph Drawing Softwares*, Mutzel P., Jünger M., (Eds.), Mathematics and Visualization. Springer-Verlag, 2003, pp. 105–126.

[AvHK06] Abello J., van Ham F., Krishnan N.: ASK-GraphView: A large scale graph visualization system. *IEEE Transactions on Visualization and Computer Graphics 12* (2006), 669–676.

[BB04] Baur M., Brandes U.: Crossing reduction in circular layouts. In *WG* (2004), Springer, pp. 332–343.

[BETT94] Battista G. D., Eades P., Tamassia R., Tollis I. G.: Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry: Theory and Applications 4* (October 1994), 235–282.

[BKH05] Bendix F., Kosara R., Hauser H.: Parallel sets: Visual analysis of categorical data. In *INFOVIS '05: Proceedings of the 2005 IEEE Symposium on*

Information Visualization (Washington, DC, USA, 2005), IEEE Computer Society, pp. 133–140.

[BM98] Batagelj V., Mrvar A.: Pajek - Program for large network analysis. *Connections 21* (1998), 47–57.

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

[BP10] Berger W., Piringer H.: Peek brush: A high-speed lightweight ad-hoc selection for multiple coordinated views. In *IV* (2010), pp. 140–145.

[Car00] Carroll J. M.: *Making Use: Scenario-Based Design of Human-Computer Interactions*. MIT Press, Cambridge, MA, USA, 2000.

[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.

[DKM06] Dwyer T., Koren Y., Marriott K.: IPSep-CoLa: An incremental procedure for separation constraint layout of graphs. *IEEE Transactions on Visualization and Computer Graphics 12* (2006), 821–828.

[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 (2008), 1293–1300.

[Ead92] Eades P.: Drawing free trees. In *Bulletin of the Institute for Combinatorics and its Applications, Vol. 5* (1992), pp. 10–36.

[ED07] Ellis G., Dix A.: A taxonomy of clutter reduction for information visualisation. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1216–1223.

[EH00] Eades P., Huang M. L.: Navigating clustered graphs using force-directed methods. *Journal of Graph Algorithms and Applications 4* (2000), 157–181.

[FK46] Forsyth E., Katz L.: A matrix approach to the analysis of sociometric data. *Sociometry 9*, 4 (1946), 340–347.

[FKCP99] Fekete J.-D., Kastler R. A., Chantrerie L., Plaisant C.: Excentric labeling: Dynamic neighborhood labeling for data visualization. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1999), ACM, pp. 512–519.

[FR91] Fruchterman T. M. J., Reingold E. M.: Graph drawing by force-directed placement. *Software - Practice and Experience 21*, 11 (1991), 1129–1164.

[Fre77] Freeman L. C.: A set of measures of centrality based on betweenness. *Sociometry 40*, 1 (1977), 35–41.

[GA00] Gürsoy A., Atun M.: Neighbourhood preserving load balancing: A self-organizing approach. In *Proceedings from the 6th International Euro-Par Conference on Parallel Processing* (London, UK, UK, 2000), Euro-Par '00, Springer-Verlag, pp. 234–241.

[GFC05] Ghoniem M., Fekete J.-D., Castagliola P.: On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Information Visualization 4*, 2 (2005), 114–135.

[GKN04] Gansner E., Koren Y., North S.: Topological fisheye views for visualizing large graphs. In *INFOVIS '04: Proceedings of the IEEE Symposium on Information Visualization* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 175–182.

[Hea] Healey C. G.: Perception in visualization. `http://www.csc.ncsu.edu/faculty/healey/PP/`, [last accessed on April 15, 2009].

[HF06] Henry N., Fekete J.-D.: MatrixExplorer: a dual-representation system to explore social networks. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 677–684.

[HF07] Henry N., Fekete J.-D.: MatLink: Enhanced matrix visualization for analyzing social networks. In *Proceedings of the International Conference Interact* (2007), pp. 288–302.

[HFM07] Henry N., Fekete J.-D., Mcguffin M.: NodeTrix: Hybrid representation for analyzing social networks. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1302–1309.

[HHE07] Huang W., Hong S.-H., Eades P.: Effects of sociogram drawing conventions and edge crossings in social network visualization. *J. Graph Algorithms Appl. 11*, 2 (2007), 397–429.

[HJ04] Hachul S., Jünger M.: Drawing large graphs with a potential-field-based multilevel algorithm. In *Graph Drawing* (2004), pp. 285–295.

[HMM00] Herman I., Melançon G., Marshall M. S.: Graph visualization and navigation in information visualization: a survey. *IEEE Transactions on Visualization and Computer Graphics 6* (2000), 24–43.

[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.

[Kat53] Katz L.: A new status index derived from sociometric analysis. *Psychometrika 18*, 1 (March 1953), 39–43.

[KFOB99] Kurtenbach G., Fitzmaurice G. W., Owen R. N., Baudel T.: The hotbox: efficient access to a large number of menu-items. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit* (New York, NY, USA, 1999), CHI '99, ACM, pp. 231–237.

[KMS02] Keim D. A., Müller W., Schumann H.: Information visualization and visual data mining; state of the art report. In *Proceedings Eurographics 2002* (Saarbrücken, 2002).

[Kor02] Koren Y.: On spectral graph drawing. In *Proceedings of the 9th annual international conference on Computing and combinatorics* (2002), Springer-Verlag, pp. 496–508.

[LA94] Leung Y. K., Apperley M. D.: A review and taxonomy of distortion-oriented presentation techniques. *ACM Trans. Comput.-Hum. Interact. 1* (June 1994), 126–160.

[LPP*06] Lee B., Plaisant C., Parr C. S., Fekete J.-D., Henry N.: Task taxonomy for graph visualization. In *BELIV '06: Proceedings of the 2006 AVI workshop on BEyond time and errors* (New York, NY, USA, 2006), ACM, pp. 1–5.

[MBK97] McGrath C., Blythe J., Krackhardt D.: The effect of spatial arrangement on judgments and errors in interpreting graphs. *Social Networks 19*, 3 (1997), 223–242.

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

[Noa05] Noack A.: Energy-based clustering of graphs with nonuniform degrees. In *Proceedings of the 13th International Symposium on Graph Drawing (GD 2005* (2005), Springer-Verlag, pp. 309–320.

[OS04] Olinky R., Stone L.: Unexpected epidemic thresholds in heterogeneous networks: The role of disease transmission. *Phys. Rev. E 70* (Sep 2004), 030902.

[PBH08] Piringer H., Berger W., Hauser H.: Quantifying and comparing features in high-dimensional datasets. In *IV* (2008), pp. 240–245.

[PS06] Perer A., Shneiderman B.: Balancing systematic and flexible exploration of social networks. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 693–700.

[PTMB09] Piringer H., Tominski C., Muigg P., Berger W.: A multi-threading architecture to support interactive visual exploration. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1113–1120.

[PV06] Pretorius A. J., Van Wijk J. J.: Visual analysis of multivariate state transition graphs. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 685–692.

[PvW08] Pretorius A. J., van Wijk J. J.: Visual inspection of multivariate graphs. *Comput. Graph. Forum 27*, 3 (2008), 967–974.

[SA06] Shneiderman B., Aris A.: Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 733–740.

[Shn94] Shneiderman B.: Dynamic queries for visual information seeking. *IEEE Softw. 11*, 6 (1994), 70–77.

[Shn96] Shneiderman B.: The eyes have it: A task by data type taxonomy for information visualizations. In *VL '96: Proceedings of the 1996 IEEE Symposium on Visual Languages* (1996), IEEE Computer Society, p. 336.

[Shn97] Shneiderman B.: *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[SLL02] Siek J. G., Lee L.-Q., Lumsdaine A.: *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[SLN05] SARAIYA P., LEE P., NORTH C.: Visualization of graphs with associated time-series data. In *INFOVIS '05: Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization* (Washington, DC, USA, 2005), IEEE Computer Society, p. 30.

[TBB88] TAMASSIA R., BATTISTA G. D., BATINI C.: Automatic graph drawing and readability of diagrams. *IEEE Trans. Syst. Man Cybern. 18*, 1 (Jan. 1988), 61–79.

[Tuk77] TUKEY J. W.: *Exploratory Data Analysis*. Addison-Wesley, 1977.

[vH03] VAN HAM F.: Using multilevel call matrices in large software projects. *Information Visualization, IEEE Symposium on 0* (2003), 29.

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

[War04] WARE C.: *Information Visualization: Perception for Design*. The Morgan Kaufmann Series in Interactive Technologies. Morgan Kaufman, 2004.

[Wat02] WATTENBERG M.: Arc diagrams: Visualizing structure in strings. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)* (Washington, DC, USA, 2002), INFOVIS '02, IEEE Computer Society, pp. 110–116.

[Wat04] WATTS D. J.: *Six Degrees: The Science of a Connected Age*. W. W. Norton & Company, February 2004.

[Wat06] WATTENBERG M.: Visual exploration of multivariate graphs. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems* (New York, NY, USA, 2006), ACM, pp. 811–819.

[WF94] WASSERMAN S., FAUST K.: *Social Network Analysis: Methods and Applications*. Structural Analysis in the Social Sciences. Cambridge University Press, 1994.

[Wil97] WILLS G. J.: NicheWorks - interactive visualization of very large graphs. In *GD '97: Proceedings of the 5th International Symposium on Graph Drawing* (London, UK, 1997), Springer-Verlag, pp. 403–414.

[Wil99] WILLS G.: Linked data views. *Statistical and Computing Graphics Newsletter 10* (1999), 24.