Abstract

Title of dissertation:	Measuring and Improving the Readability of Network Visualizations
	Cody Dunne, Doctor of Philosophy, 2013
Directed by:	Professor Ben Shneiderman Department of Computer Science

Network data structures have been used extensively for modeling entities and their ties across such diverse disciplines as Computer Science, Sociology, Bioinformatics, Urban Planning, and Archeology. Analyzing networks involves understanding the complex relationships between entities as well as any attributes, statistics, or groupings associated with them. The widely used node-link visualization excels at showing the topology, attributes, and groupings simultaneously. However, many existing node-link visualizations are difficult to extract meaning from because of (1) the inherent complexity of the relationships, (2) the number of items designers try to render in limited screen space, and (3) for every network there are many potential unintelligible or even misleading visualizations. Automated layout algorithms have helped, but frequently generate ineffective visualizations even when used by expert analysts. Past work, including my own described herein, have shown there can be vast improvements in network visualizations, but no one can yet produce readable and meaningful visualizations for all networks.

Since there is no single way to visualize all networks effectively, in this disser-

tation I investigate three complimentary strategies. First, I introduce a technique called **motif simplification** that leverages the repeating patterns or motifs in a network to reduce visual complexity. I replace common, high-payoff motifs with easily understandable glyphs that require less screen space, can reveal otherwise hidden relationships, and improve user performance on many network analysis tasks. Next, I present new **Group-in-a-Box layouts** that subdivide large, dense networks using attribute- or topology-based groupings. These layouts take group membership into account to more clearly show the ties within groups as well as the aggregate relationships between groups. Finally, I develop a set of **readability metrics** to measure visualization effectiveness and localize areas needing improvement. I detail optimization recommendations for specific user tasks, in addition to leveraging the readability metrics in a user-assisted layout optimization technique.

This dissertation contributes an understanding of why some node-link visualizations are difficult to read, what measures of readability could help guide designers and users, and several promising strategies for improving readability which demonstrate that progress is possible. This work also opens several avenues of research, both technical and in user education.

Measuring and Improving the Readability of Network Visualizations

by

Cody Dunne

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2013

Advisory Committee:

Professor Ben Shneiderman, Chair
Professor Bonnie Dorr
Assistant Professor Jon Froehlich
Professor Amitabh Varshney
Associate Professor Alan Neustadtl, Dean's Representative

To my loving family

Acknowledgments

I would like to first start by thanking my advisor, Ben Shneiderman, whose assistance was instrumental in helping me prepare this dissertation. Ben is a truly amazing graduate mentor, and he has helped me navigate the complexities of a research career and shaped me into a valuable member of the scientific community. Ben is a great professor with creativity, depth and breadth of knowledge, and vision; and his enthusiasm and encouragement constantly inspires the people around him. Ben helped me set iterative goals while keeping the big picture in mind, was always available to talk, encouraged collaboration with domain experts to develop innovative dissertation topics, and kept the dissertation in mind as the main goal for every project. He constantly dedicated his time, money, and connections to ensure I had opportunities to do interesting research and present the results. I was very pleased that my nomination of him for the University of Maryland's Graduate Faculty Mentor of the Year Award was selected as one of only four in 2013.

However, I have also had a large support network of other mentors and collaborators throughout my dissertation work. My committee members have played a pivotal role, both in my dissertation work and my job search, and I would like to thank them in particular: Bonnie Dorr, Jon Froehlich, Amitabh Varshney, and

Alan Neustadtl. Other collaborators I would like to thank are Catherine Plaisant, Marc Smith, Nathalie Henry Riche, Derek Hansen, Tony Capone, Ping Wang, Leah Findlater, Adam Perer, Anne Rose, Seth Powsner, Manuel Freire, Elizabeth Bonsignore, Eduarda Mendes Rodrigues, Natasa Milic-Frayling, Judith Klavans, Robert Gove, Saif Mohammad, Bongshin Lee, Ron Metoyer, George Robertson, Snigdha Chaturvedi, Zahra Ashktorab, Rajan Zacharia, Puneet Sharma, Udayan Khurana, Krist Wongsuphasawat, Darya Filippova, Awalin Sopan, Alex Quinn, Peter Fontana, Nick Gramsky, Rose Kirby, Emre Sefer, Meirav Taieb-Maimon, Andreea Olea, Eylul Dogruel Vladimir Barash, Eric Gleave, Dana Rotman, Ryan Blue, Adam Fuchs, Kyle King, Aaron Schulman, Yiyan Liu, and many, many more.

I also appreciate the support of many funding sources for my dissertation work, including the Social Media Research Foundation; the Connected Action Consulting Group; National Science Foundation grants SBE 0915645, IIS 0705832, and IIS 0968521; HHS SHARP grant 10510592; Microsoft External Research; Microsoft Research; the National Cancer Institute, and several University of Maryland travel grants.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Networks have long been common data structures in Computer Science, but have only recently exploded into popular culture. Publishers like the New York Times now frequently including elaborate and interesting networks with their articles.[1] Online communities like Facebook, Twitter, Flickr, MySpace, and YouTube (to name only a handful) enjoyed enormous growth over the last few years and provide rich datasets of interpersonal relationships, which social scientists are now fervently exploring. Networks have also found applications in such diverse disciplines as bioinformatics, scientometrics, urban planning, politics, and archeology.

Analysis of these datasets requires knowledge of the connectivity, clusters, and centrality of the nodes: tasks which necessitate relationship visualizations. Statistical analysis and conventional visualization tools like bar and pie charts are often inadequate when faced with these varied and oftentimes immense datasets. www.visualcomplexity.com and its associated book [Lim13] provide many beautiful alternative visualizations for these data which are surveyed by [Ari08; SA06],

[1] http://www.nytimes.com/2009/03/29/technology/internet/29face.html

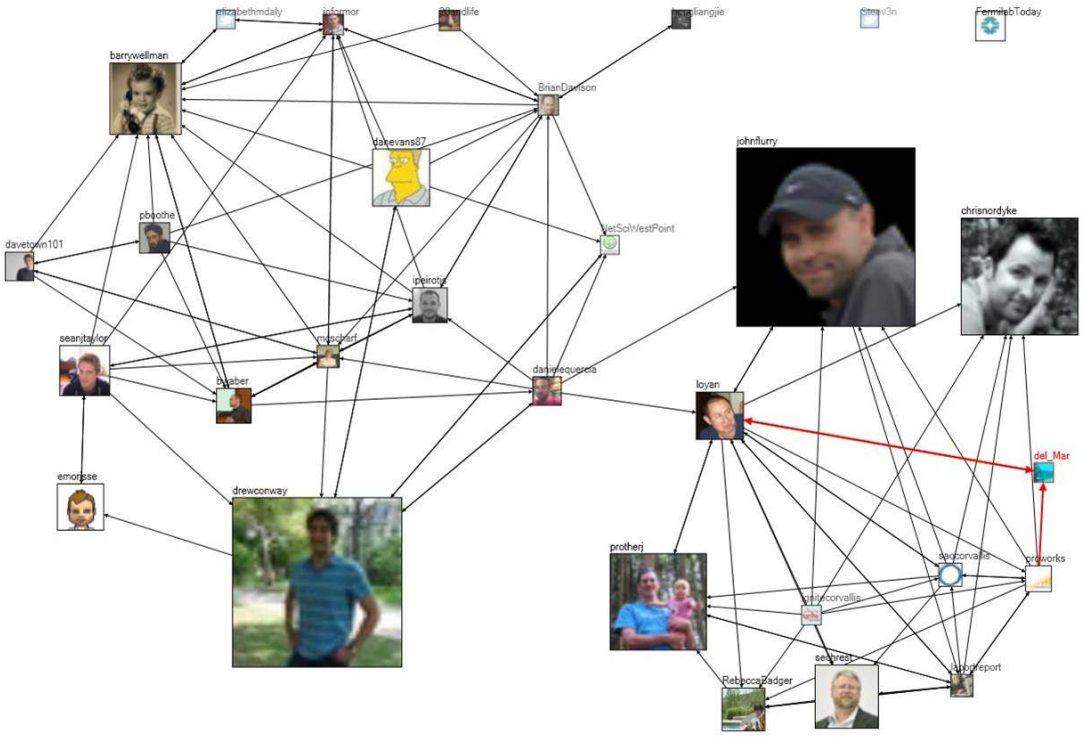


Figure 1.1: A node-link visualization of relationships among Twitter users mentioning the hashtag "#WIN09", which was used by participants at a network science conference in September 2009. Each Twitter user is represented by a node containing its image, and edges between users indicate follow, mention, or reply relationships. The force-directed layout used to position the nodes highlights interesting patterns of connectivity like the two large communities of researchers. From Fig. 3.1 of the NodeXL book [HDS10, p. 33].

but one enduring technique in particular models relationships using a node-link visualization, where nodes in the network represent entities and the links or edges indicate ties connecting them [BMK96]. An example node-link visualization is shown in Fig. 1.1, which displays relationships among Twitter users at the WIN09 conference and how they separate into two distinct communities of researchers. This network of interactions between people is called a **social network** and the

resulting visualization is called a sociogram by sociologists [Mor53], graph drawing by graph theorists, and a **node-link visualization** by other network researchers including myself.

Node-link visualizations have a long history, but only in the last few decades have we seen their frequent application as a network exploration tool. For example, Fisher, Smith, and Welser [FSW06] and Welser et al. [Wel+07] successfully used node-link visualizations to detect common social roles in online discussion newsgroups such as answer person and discussion person. Node-link visualizations have also been applied to the study of relationships between political blogs during the 2004 U.S. Presidential Election, showing the division between liberal and conservative communities as well as their internal interactions [AG05]. A similar large application is to map the entire Internet [CBB00]. These techniques have made inroads into many other domains as well. Urban planners have used node-link visualizations to understand networks of innovation (Section 5.5.2, [Dem12]), and, similarly, scientometricians use them for measuring and analyzing scientific publishing (Section 3.3.3, [Hen+07]). In biology and medicine, node-link visualizations are used to help explore protein-protein interaction networks [Kel+03] and to visualize patient conditions and treatments (Section 4.3.5, Section 5.5.3). Even archeology now uses node-link visualizations for looking at the relationships between dig sites and artifacts (Section 3.3.2, [Bru12]).

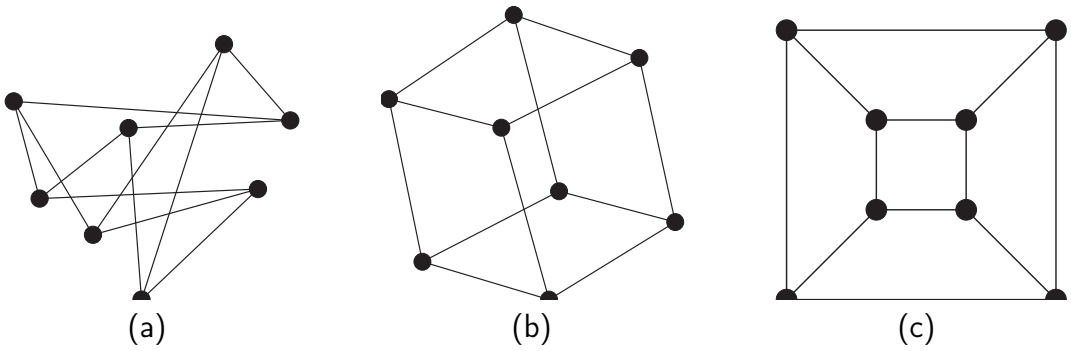


Figure 1.2: Different visualizations of the same network, with (a) obscuring the topology while (b) and (c) are more understandable with less edge crossings.

However, there are a huge array of possible layouts of the nodes and edges in any given network, many of which can create misleading or incomprehensible visualizations [Bra+99]. Even eight nodes can be laid out in a way that obscures the network topology, as displayed in Fig. 1.2. In this case, edge crossings caused by the layout make paths difficult to follow, but other problems can be caused by nodes overlapping or edges tunneling underneath nodes without connecting to them, to name only a few of many potential readability issues. Visualizations of relational structures like networks are only useful to the degree they "effectively convey information to the people that use them" [Bat+98]. What's more, there is no "best" layout for a network as different layouts can highlight different features of the network being studied [BMK96]. In fact, the spatial layout of nodes in the node-link visualization can have a profound impact on the detection of communities in the network and the perceived importance of individual actors [MBK97].

Hence, significant thought must be given to properly laying out networks so that network analysts will be able to understand and effectively communicate data such as clusters in the network, the paths between them, and the importance of individual actors.

As manual layout of nodes in the node-link visualization is incredibly time consuming to do well, a lot of effort has been put into developing automated network layout algorithms. There are many layout algorithms that can be used, including variants of the spring embedder [Ead84] such as the popular Fruchterman-Reingold force-directed algorithm [FR91] (used in Fig. 1.1), the Prefuse gravitational N-Body approach [HCL05], the Harel-Koren fast multi-scale (FMS) algorithm [HK02a], the high-dimensional embedding (HDE) approach of Harel and Koren [HK02c], the algebraic multigrid method (ACE) of Koren, Carmel, and Harel [KCH03], and FM^3 by Hachul and Jünger [HJ05]. These force-directed algorithms are used frequently in practice. A 2006 census of the layout algorithms used for the first 100 examples on visualcomplexity.com showed that over a third used force-directed algorithms, with another third using geographic placements [SA06].

Even with these layout techniques, many existing node-link visualizations of networks are not easily readable, or at least difficult to extract meaning from. Several factors contribute to this problem, including that the inherently complex relationships in large, dense networks are often difficult to perceive even with mas-

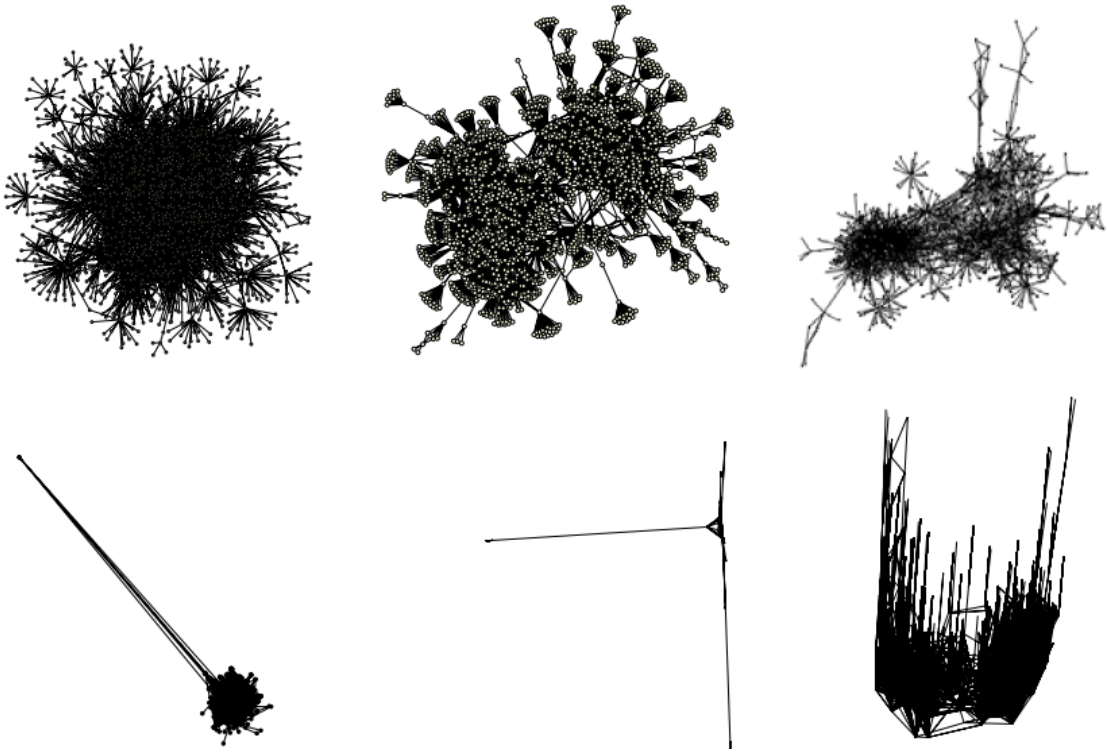


Figure 1.3: An experimental comparison of six layout algorithms on the same social network produced widely different layouts. The top row layouts performed well, though bottom row layouts are difficult to extract meaning from. From [HJ06].

sive displays. Also, as shown in Fig. 1.3, layout algorithms can produce vastly different results for the same network depending on the heuristics they use. The spatial layout of a network visualization is critical to what we perceive from it, meaning that for every network and user task there are many potential unintelligible or even misleading visualizations. Moreover, end users are completely unwilling to experiment with layout parameters to improve the layout after the initial view [Bar+08]. Even expert analysts who have the experience required to tweak the layout algorithm and further optimize the layout manually can have difficulties

with large networks. Many researchers, including myself, have shown that there can be vast improvements in network visualizations by using alternate approaches to layout [HK02c; HJ05], aggregation [Wat06; Dun+12a], and filtering [SD12]. However, many challenges remain.

My dissertation work contributes to this space and focuses on three complimentary approaches for helping users explore network datasets. First, I introduce a technique called **motif simplification** which helps users reduce visual complexity by replacing repeating patterns with representative glyphs (Section 1.1). These glyphs require less screen space, better present the core interesting parts of the network, and improve user task performance. Second, I present new **Group-in-a-Box layouts** to segment dense networks using attribute- or topology-based groupings (Section 1.2). These group-aware layouts can better display the relationships within groups as well as between them. Finally, I develop a set of what I term **readability metrics** to measure the effectiveness of node-link visualizations, both to analyze the utility of layout algorithms but also to interactively guide user improvement of the layout (Section 1.3). I implemented each of these techniques in the free and open source NodeXL [Smi+10] network analyst tool, so that they could be easily used by novice network analysts (Section 1.4). These three techniques and their associated NodeXL implementations are discussed in the following sections, which provide an overview of the chapters in this dissertation.

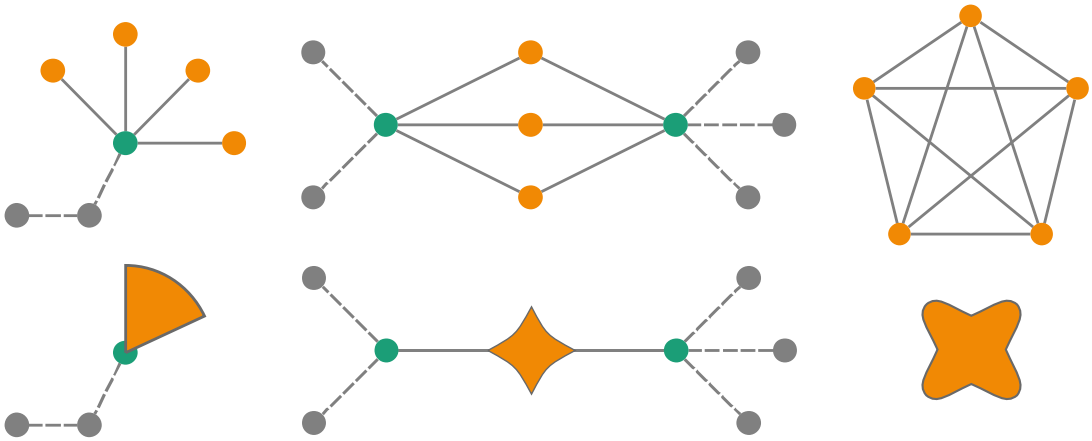


Figure 1.4: Fan, connector, and clique motifs (top) and their glyphs (bottom).

1.1 Motif Simplification to Reduce Complexity

Many complex networks are littered with recurring topologic patterns or **motifs**, either because of the network structure or data collection methods. Three of these motifs are shown in the top row of Fig. 1.4. Regardless of their cause, some frequently expressed motifs contain little information compared to the space they occupy in the visualization. My dissertation helps address this problem with a new technique called **motif simplification**, in which common repeating motifs are replaced with compact yet meaningful glyphs. I focus on the three frequently occurring and high-payoff motifs shown in Fig. 1.4: **fans** of nodes with a single neighbor, **connectors** that link a set of anchor nodes, and **cliques** of completely connected nodes. My research contributes efficient algorithms for motif detection, the design of representative and combineable glyphs, as well as guidelines and an iterative process for creating glyphs for additional motifs.

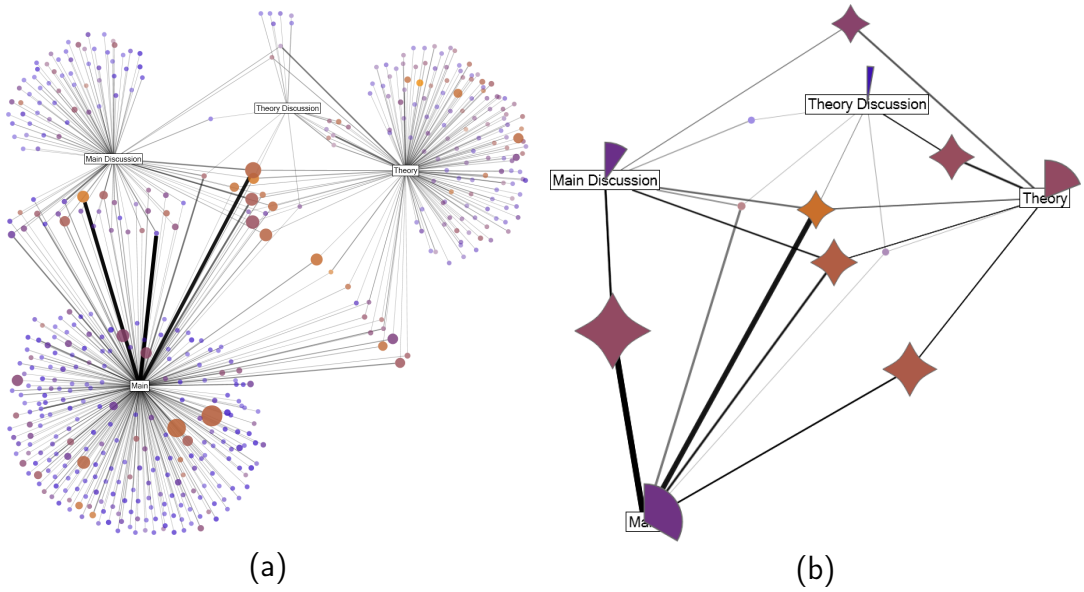


Figure 1.5: A bipartite network of Lostpedia of wiki edits (a) and a simplified version using glyphs for fan and connector motifs (b).

I evaluated motif simplification first with several domain experts in sociology, political science, medical informatics, and the U.S. Department of the Treasury to understand the effectiveness of the technique for real-world analyses. I followed this with a task-based controlled study of 36 participants analyzing networks up to 3958 nodes, which determined the magnitude of any performance differences between using plain and simplified views. One example network from this study is shown in Fig. 1.5, in which a network of wiki editors connected to the pages they edit is shown in a node-link visualization with 513 nodes (left) and the simplified view with only 17 nodes and glyphs (right). These studies showed that motif simplification (1) reduces screen space used and layout effort, (2) can reveal hidden relationships,

and (3) is quite beneficial for many network analysis tasks both in the time users took and their accuracy/error. Unlike other approaches, motif simplification is able to achieve these benefits while maintaining user awareness of the underlying topology. Please see Chapter 4 for more details on motif simplification.

1.2 Meta-Layouts for Subdividing Networks

In contrast to motif simplification, in which functionally equivalent nodes and edges are replaced by representative glyphs, I have also explored the use of **meta-layouts** that highlight more general topology- or attribute-based groupings of the network. These groups can be difficult to understand using the standard tools of color, shape, or convex hulls – as evidenced by the dense, intermingled topologic clusters shown in Fig. 1.6. In this visualization, it is difficult to understand the size of each group, its internal structure, and its ties to other groups. My meta-layouts are designed to make all these group features easier to discern. First, the **Midichlorian-Directed Layout** is a modified force-directed layout algorithm that reduces spring forces between nodes in separate groups. This causes groups to spread apart and be more clearly analyzed, but at the expense of substantial screen space required. Next, I present several **Group-in-a-Box layouts** that display groups individually to more clearly show membership, topology, and inter-group relationships. We have one such layout in NodeXL [Smi+10] that segments

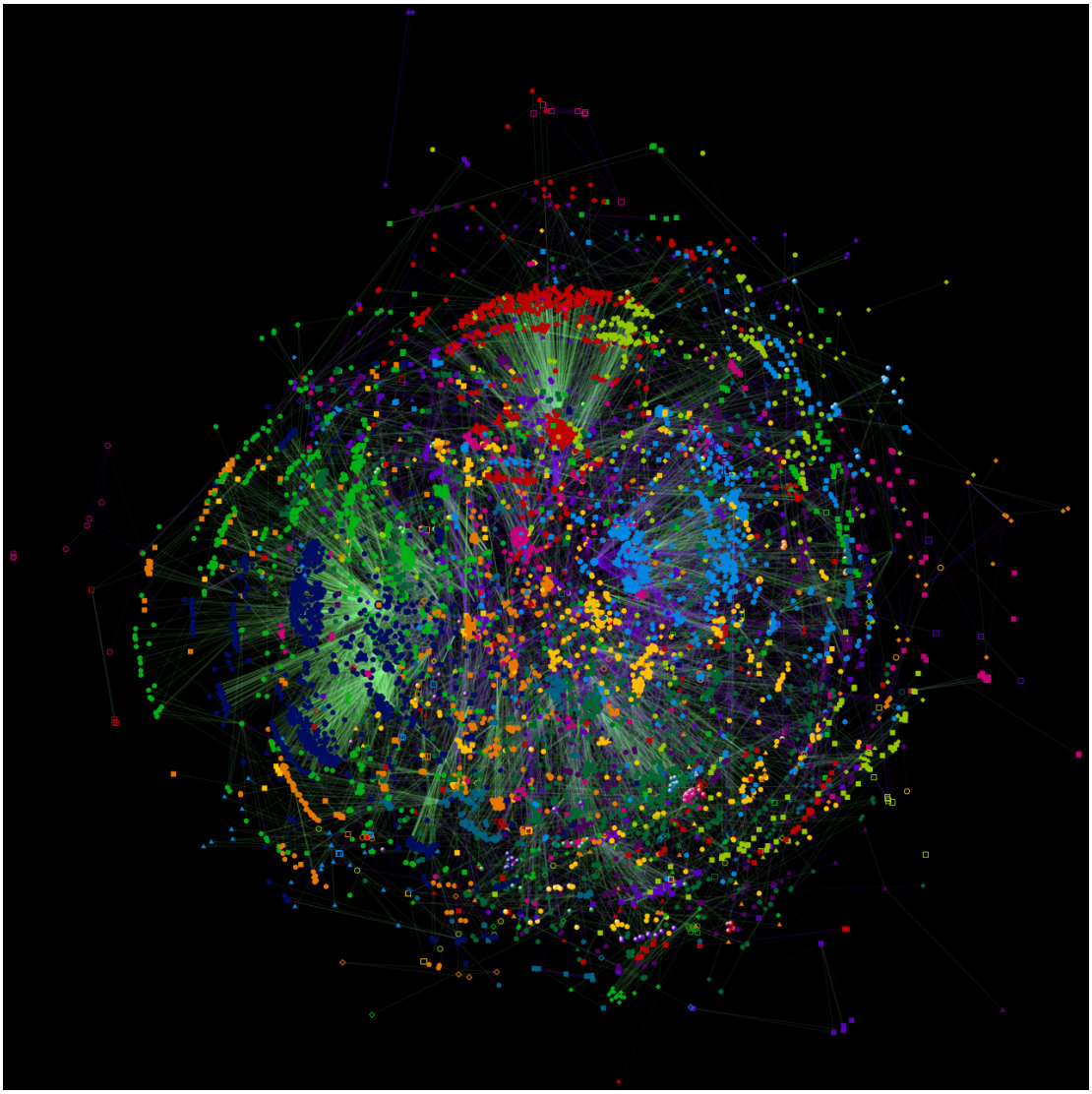
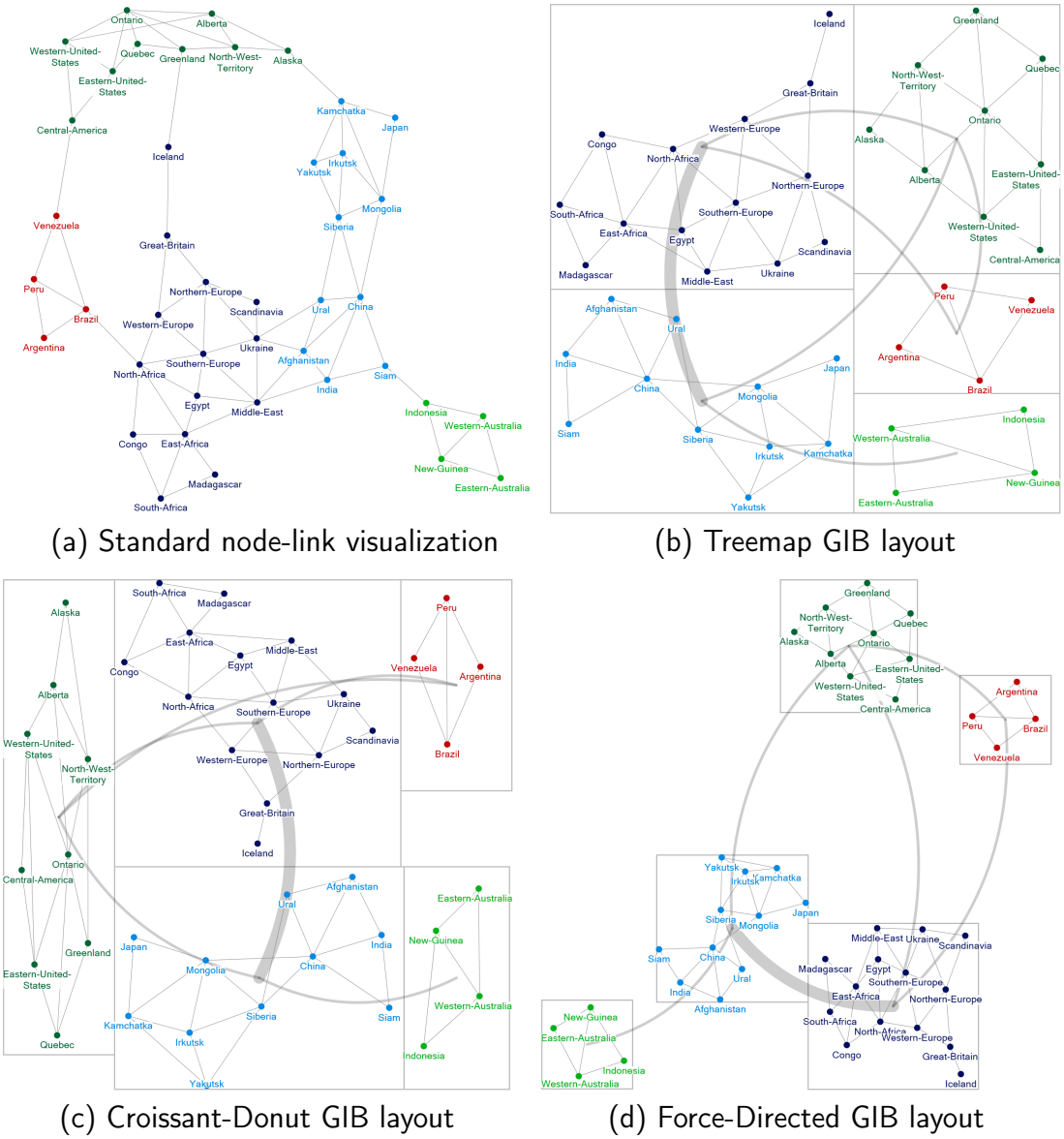


Figure 1.6: Pennsylvania innovation relationships during 1990 (main component) collected by Christopher Scott Dempwolf. Nodes are laid out using the Harel-Koren FMS layout [HK02a] and topologic clusters found using the Clauset-Newman-Moore algorithm [CNM04] are shown using node color and shape. See Section 5.5.2 for more details and analyses of this network.



(a) Standard node-link visualization

(b) Treemap GIB layout

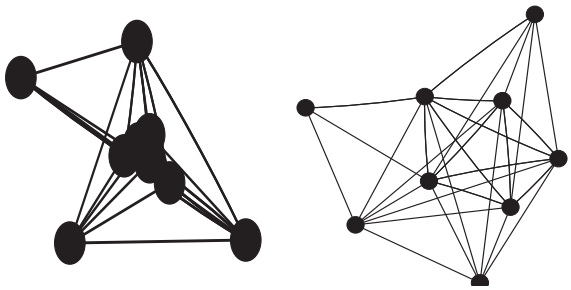
(c) Croissant-Donut GIB layout

(d) Force-Directed GIB layout

Figure 1.7: The network for the board game Risk, where nodes are countries and edges indicate legal movements. Nodes are laid out using Harel-Koren FMS [HK02a], clustered and colored using the Clauset-Newman-Moore topologic clustering algorithm [CNM04]. Inter-group edges are combined into thick meta-edges. (a) shows the initial visualization, while the others show the three Group-in-a-Box (GIB) layout variants. See Section 5.5.1 for more details and analysis.

groups using a **Treemap** [Rod+11; SD12], which is space-filling but often separates related groups, drawing long edges which overlap other groups unnecessarily. This is visible in Fig. 1.7b as the crossing and overlapping meta-edges that represent the combined inter-group edges.

I present several variants to more clearly show group relationships, each best suited to a range of topologies. The **Croissant Group-in-a-Box layout**, shown in Fig. 1.7c, puts the largest group at the top and wraps the remainder around three sides based on their connectivity. This effectively displays large groups, though more smaller groups are better shown using the **Donut Group-in-a-Box layout** (not shown here) which places the largest group in the center and arranges others around the perimeter. Finally, the **Force-Directed Group-in-a-Box layout** (Fig. 1.7d) arranges groups based on their aggregate ties and eliminates any overlap of their boxes. The NodeXL [Smi+10] implementation automatically picks the best approach for the given data to better show disconnected components, few groups, or different distributions of group sizes and connectedness. Several case studies and experiments demonstrate that Group-in-a-Box layouts more clearly show (1) topology within groups, (2) group membership and size, and (3) aggregate relationships between groups. Group-in-a-Box layouts are particularly effective for large networks, where high density and finite screen space limit effective network visualizations. I cover my work on meta-layouts extensively in Chapter 5.



(a) Tight layout (b) Relaxed layout

Figure 1.8: We can eliminate the node occlusion and edge tunnels that make the central overlapping group in Fig. 1.8a so hard to understand by zooming out and increasing the the spring lengths of the layout algorithm (Fig. 1.8b).

1.3 Measuring Network Visualization Readability

My user studies, case studies, and experiments demonstrate the utility of motif simplification and Group-in-a-Box layouts for network visualization, but I am also interested in improving the effectiveness of general node-link visualizations. By quantifying the readability of a layout, we can guide analysts in making improvements and feed the results in automatic layout algorithms. Past work by Purchase and Leonard [PL96; Pur02] as well as Ware et al. [War+02] provides definitions for several of what I call **global readability metrics** (also called aesthetic criteria), which measure detrimental features like edge crossings (see Fig. 1.2) and rate the layout as a whole. However, a single value is not enough to direct users to problem areas of the layout, which part of my dissertation addresses by introducing **local readability metrics** for individual nodes and edges. Moreover, I introduce sev-

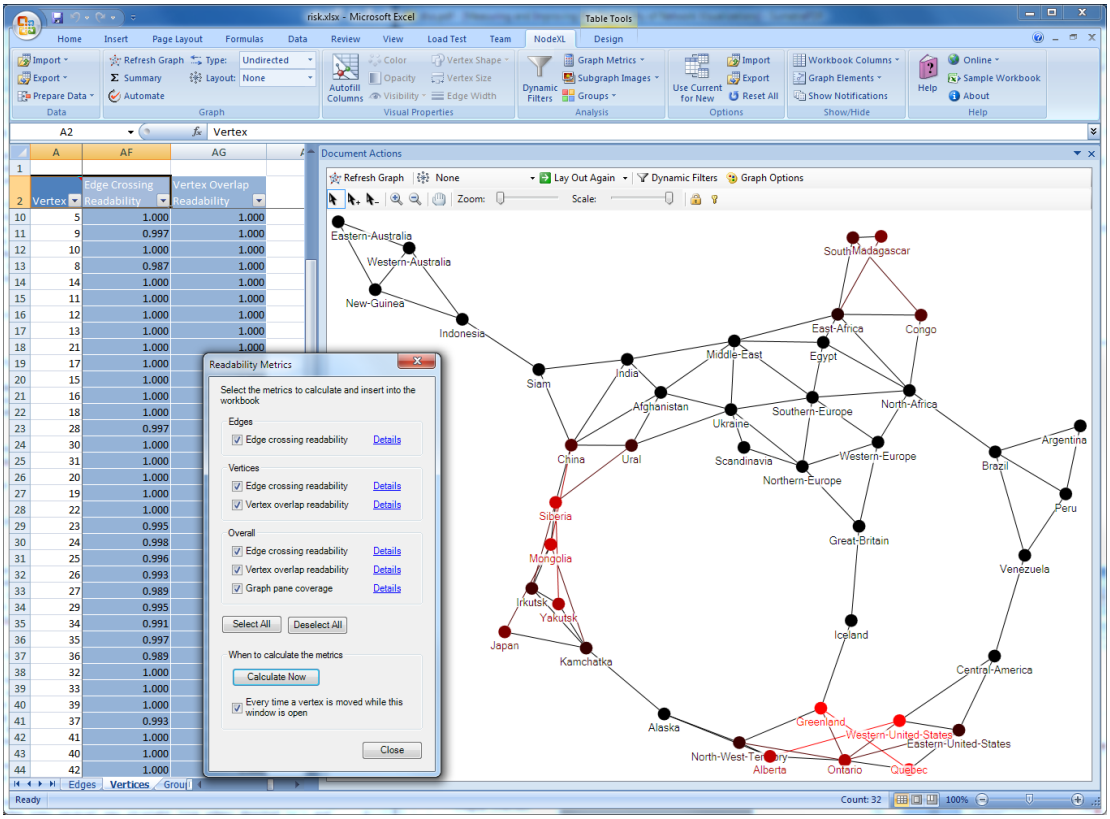


Figure 1.9: NodeXL showing the readability metrics dialog (foreground), the nodes in the worksheet with edge crossing and node overlap metric columns, and visualization where nodes and edges are colored red-to-black by the edge crossing metric. The worst offenders are shown in red. The network shown represents the legal moves in the board game Risk from Fig. 1.7a.

eral new global metrics to detect readability problems like node overlap and edges tunneling under nodes. These readability issues are visible on the left of Fig. 1.8.

I leverage these metrics in a new method for user-assisted layout improvement, which is shown in Fig. 1.9. My approach is to incrementally update the readability metrics in real-time as users manipulate the layout, and provide immediate visual feedback to users showing how they are affecting readability. As there are trade-offs

when optimizing specific readability metrics, I include a survey of the related literature studying each of these metrics and their effect on user task performance. My evaluations indicate that these readability metrics help users create more effective node-link visualizations, and I plan to release both the metrics and layout improvement tool as part of NodeXL [Smi+10]. This work aims to raise user awareness of network visualization readability issues, and applying my optimization technique will guide users in creating more effective network visualizations.

1.4 Exploration Environment

I implemented each of these three approaches in a scalable environment for network exploration and improvement, made publicly available as part of the free and open source NodeXL network analysis tool [Smi+10]. NodeXL is popular and actively developed, has over 184,000 downloads, and has been taught in over 25 introductory courses on network analysis and visualization. I have been involved with the project for five years, first running user studies and then as an advisor and developer. By releasing my work in NodeXL, it immediately becomes available to help the novice users who need it the most. Motif simplification is now available and visible in the publicly shipping tool, and my Group-in-a-Box layouts will be shortly. The readability metrics and associated interactive layout improvement technique are implemented but hidden as they are not yet ready for public use.

1.5 Specific Contributions

The specific contributions of this dissertation are as follows:

- Motif Simplification
  - A technique for simplifying node-link visualizations by replacing common network motifs with representative glyphs,
  - A set of design guidelines for these glyphs to show the motif contents and underlying attributes,
  - The design of glyphs for fans, connectors, and cliques,
  - Algorithms for detecting these three motifs,
  - A supporting task-based study with 36 participants, and
  - A free and open source implementation as part of NodeXL.
- Meta-Layouts
  - A meta-layout called the Midichlorian-Directed Layout which spreads groups apart in a standard node-link visualization;
  - A Croissant-Donut Group-in-a-Box layout that places subnetworks in boxes arranged using a Donut or Croissant pattern, and balances space-filling properties with showing group relationships;

  - A Force-Directed Group-in-a-Box layout that places subnetworks in boxes arranged by their connectivity, and shows group relationships well at the expense of additional screen space;
  - A set of automatic choices that are made for the user to better show disconnected components, few groups, or different distributions of group sizes and connectedness;
  - Supporting case studies and an experiment on Twitter networks; and
  - A free and open source implementation as part of NodeXL.

- Readability Metrics
  - New global readability metrics to help understand different aspects of network visualization readability,
  - Local readability metrics for individual nodes and edges to help users identify problem areas and fix them,
  - A method for user-assisted layout improvement that provides real-time metric feedback to users in a ranked list and with a color scale,
  - Implementations of readability metrics and the layout improvement technique in SocialAction and NodeXL, and
  - A survey of work on readability metrics and evaluations of their effectiveness on various network analysis tasks.

This dissertation is aimed at helping researchers, tool designers, and network analysts. For researchers, my work demonstrates that progress is possible in improving node-link visualization readability and contributes to the literature an improved understanding of why some network visualizations are difficult to read. For designers of network analysis tools, I detail specific techniques they can implement and give guidance as to what measures of readability could help users create more effective visualizations. For analysts, I hope to raise awareness that the images they share or publish could be of higher quality, so that readers could extract relevant information. Furthermore, I provide an implementation of my techniques analysts can apply, so as to improve the utility of their network visualizations through layout changes and meaningful aggregations. My three strategies are complementary and applicable to many types of networks and user explorations. The techniques can be applied separately or in combinations based on the type of network and tasks involved, with different methods better for highlighting certain characteristics.

1.6 Dissertation Roadmap

The remainder of this dissertation is broken into several parts. First, in Chapter 2 I discuss prior work done on network exploration, measuring readability, analyzing motifs, meta-layouts, and visualization evaluation. Next, in Chapter 3 I detail

the NodeXL network analysis tool [Smi+10] in which many of my dissertation contributions are implemented, as well as several applications of network analysis to problems in diverse domains. These applications helped guide my dissertation research. Then, Chapter 4 covers the motif simplification approach for reducing complexity by combining functionally equivalent nodes and edges. Moving on, Chapter 5 describes the meta-layout and Group-in-a-Box approaches for subdividing complex networks into manageable yet meaningful pieces. Chapter 6 then discusses techniques for understanding and improving the readability of a standard node-link network visualization. Finally, I conclude and discuss future directions in Chapter 7. Parts of this work have already been published [DS13; SD12] or are currently under submission [Cha+13], in addition to the many domain-specific publications discussed in Chapter 3.

Chapter 2

Related work

2.1 Introduction

The field of network analyses and visualization is extensive. In this chapter I provide an overview of general network visualization principles, as well as detailed discussion of the techniques most relevant to my dissertation contributions. First, in Section 2.2 I detail general techniques for network visualization and analysis, including alternatives to the standard node-link visualization that have various tradeoffs. I have chosen to focus my work on improving node-link visualizations as they are the best visualization for understanding the overall structure of a network and for many important path-based tasks [HF07]. Moreover, they are incredibly widely used [Ari08; SA06] and the only network visualization available in common analysis tools like NodeXL [Smi+09] (Section 3.2), Gephi [BHJ09], Cytoscape [Sha+03] (Fig. 2.2), Pajek [BM98], and GUESS [Ada06].

Next, I describe the current techniques for measuring the readability of node-link visualizations in Section 2.3. These techniques form the basis for my work on

readability metrics, which I use to help users both understand and improve the readability of their node-link visualizations. Third, in Section 2.4, I cover work with similar goals as my motif simplification technique. This includes approaches for aggregating, clustering, or filtering networks based on topology or attributes, in addition to detecting frequently occurring motifs in networks. Moving on, I detail techniques for taking groups or subnetworks into account when computing layouts in Section 2.5 and contrast these with my Group-in-a-Box meta-layouts. Some of my techniques I can evaluate empirically using simulations, but in many cases it is important to put them in front of real users to determine real-world utility. I relate common evaluation techniques for these kinds of studies in Section 2.6. Finally, I summarize the novelty of my approaches in Section 2.7.

2.2 Network Visualization & Analysis

The area of network analysis is currently of great interest to the community, and many systems have been developed to visualize and analyze networks. There are several general visualization frameworks that can be extended programmatically to create arbitrary visualizations of networks or other datasets, such as the Info-Vis Toolkit [Fek04], Prefuse [HCL05], and JUNG [OM+03]. Traditionally, dedicated network analysis tools have focused on two specific kinds of visualizations: node-link and matrix representations. Node-link visualizations excel at showing

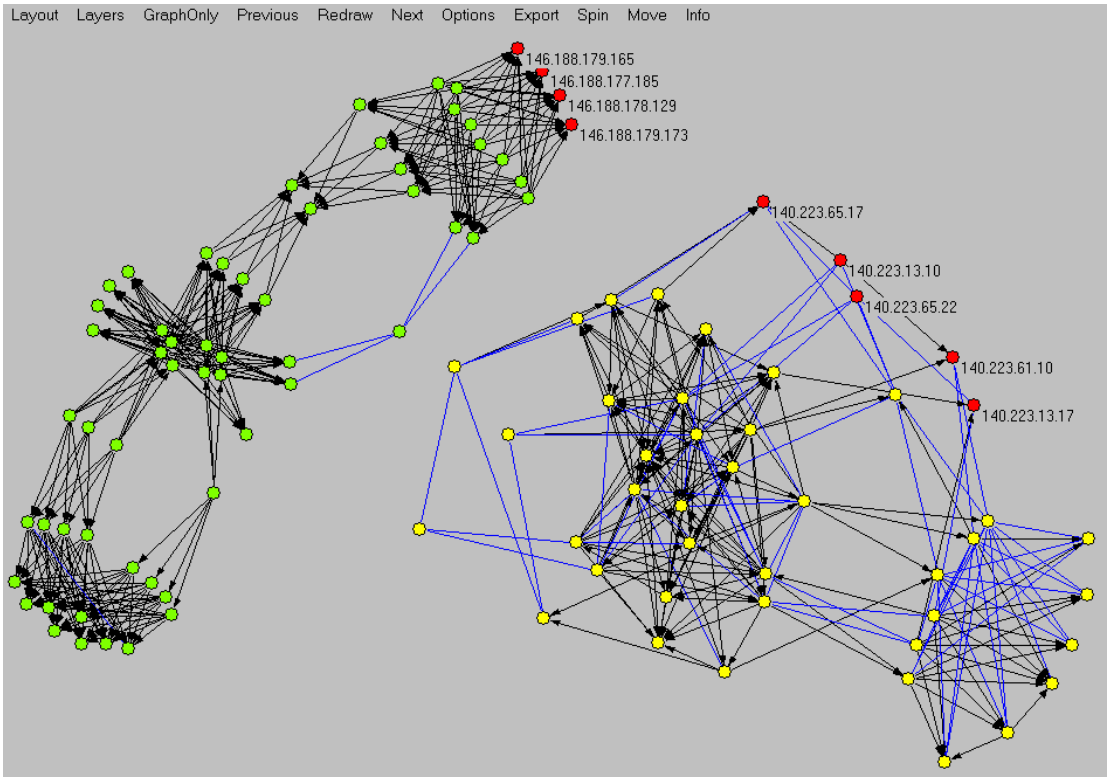


Figure 2.1: The Pajek social network analysis tool [BM98] showing the main core subgraph extracted from Internet routing data.

network topology, especially in sparse social networks. Most general-purpose and domain-specific network analysis tools incorporate node-link visualizations, including NodeXL [Smi+09] (Section 3.2), Gephi [BHJ09], Cytoscape [Sha+03] (Fig. 2.2), Pajek [BM98] (Fig. 2.1), GUESS [Ada06], and SocialAction [PS06]. I focus my efforts on improving the utility of node-link visualizations both because of their effectiveness at showing overall network topology, as well as their wide usage.

Matrix representations are less frequently used, but are better suited to especially dense networks. MatrixExplorer [HF06], TimeMatrix [YEL10], and Matrix

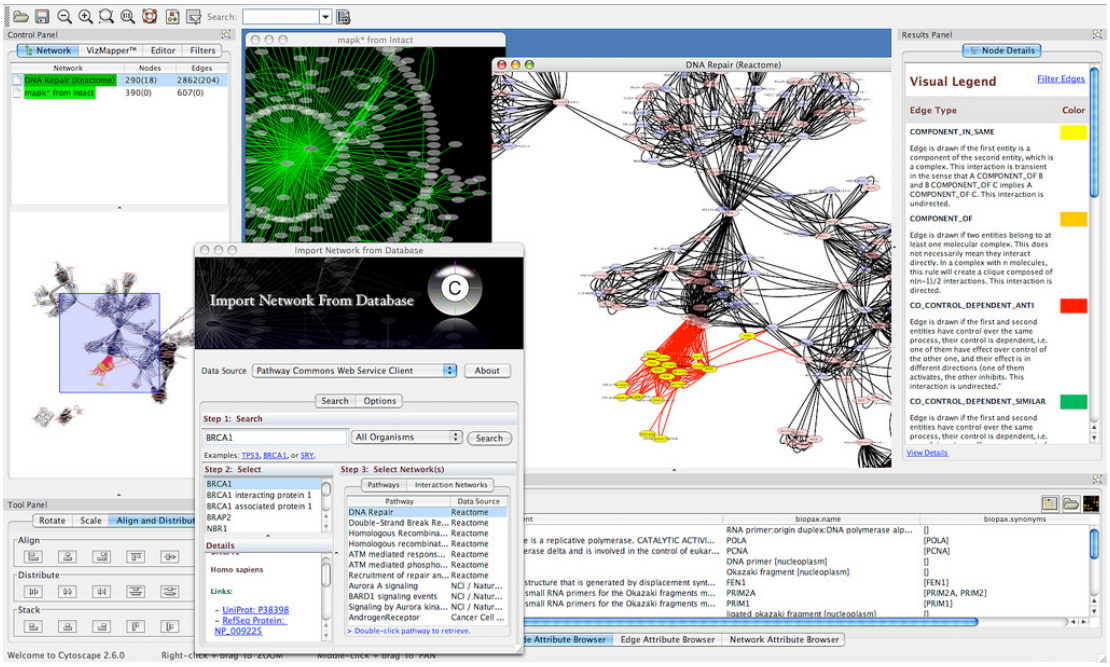


Figure 2.2: The Cytoscape biologic network analysis tool [Sha+03].

Zoom [AH04] are prime examples of matrix visualizations. Whether a matrix or node-link representation is better suited for a specific network depends substantially on the size and characteristics of that network. Node-link visualizations are favored in all cases for path-finding tasks [GFC04] and both show the overall topology of small networks quite well, but readability becomes an issue when confronted with more than a few thousand nodes. Several recent tools like MatLink [HF07] and NodeTrix [HFM07] (Fig. 2.3) have worked to integrate the matrix and node-link representations to combine their strengths. However, the node-link visualizations I focus on remain the most widely used as well as the most effective network overview visualization.

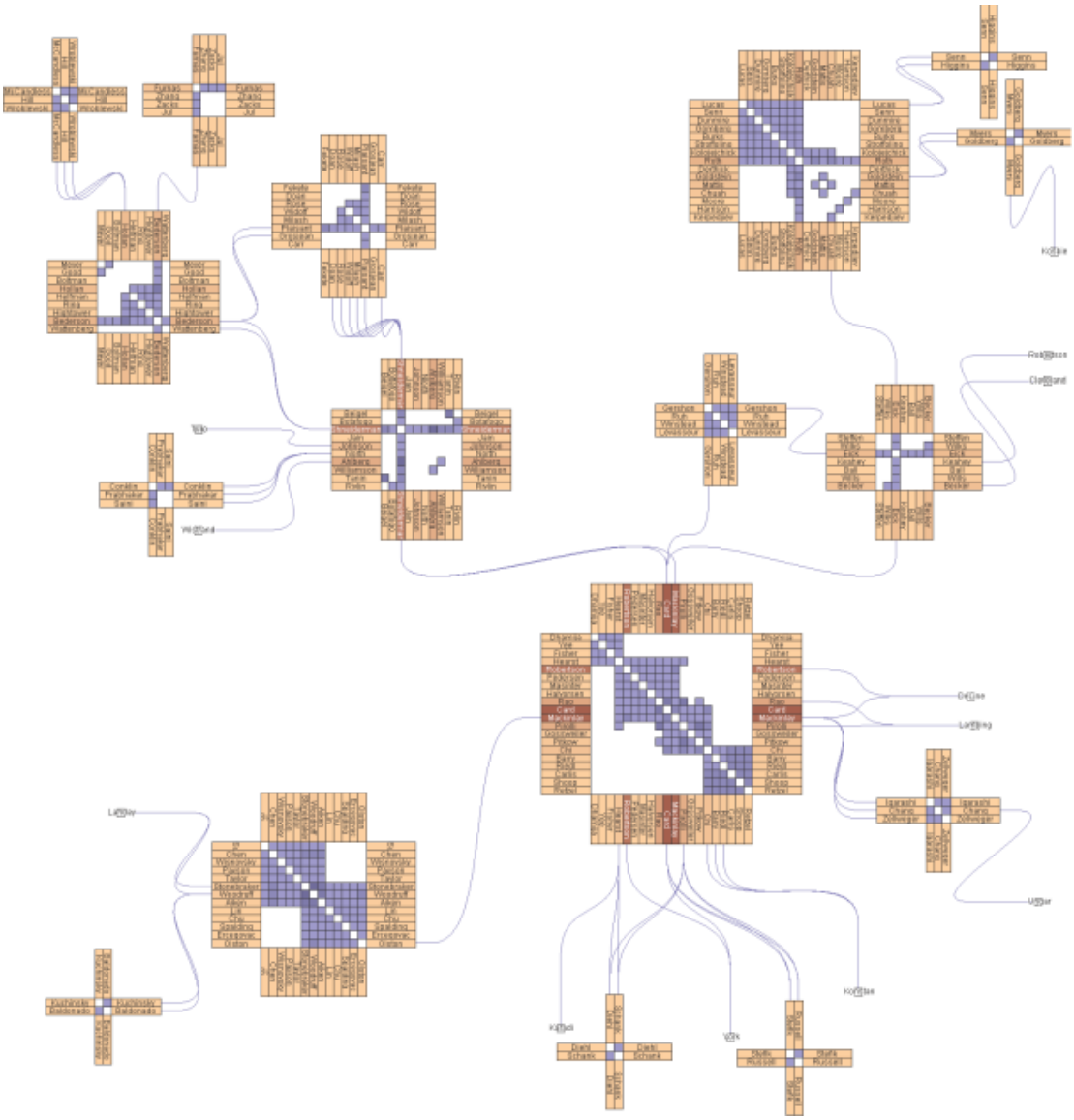


Figure 2.3: NodeTrix [HFM07] showing an overview of research in information visualization from the InfoVis '04 contest.

Social network datasets such as scientific collaboration networks or friendship networks often contain multiple types of nodes and edges (i.e., heterogeneous), and multiple attributes on nodes or edges (i.e., multivariate). In node-link visualizations, multiple attributes can be encoded using size, color, shape, opacity, etc [Mac86]. In particular, [Bla+09] recently attempted to represent multiple types of edges in node-link visualizations using texture and animation. However, it remains challenging to identify patterns and extract trends by solely relying on these visual encodings. My implementations in NodeXL [Smi+10] provide all these state-of-the-art attribute encodings for the node-link visualization, excluding animation. The motif simplification approach even shows underlying color or size information in the representative glyph for a motif. Unfortunately, effective attribute exploration requires alternate visualizations, like those I discuss in the following paragraphs and Section 3.3.2.

Various hybrid network visualizations attempt to combine topology and multivariate data more effectively into a single visualization such as the scatter plots of nodes connected by edges in Semantic Substrates [SA06] (Fig. 2.4) or GraphDice [Bez+10] (Fig. 2.5). Other hybrid approaches provide a visualization of topology on top of node aggregates, such as overlaying edges on Treemaps [Fek+03], combining Treemaps with node-link visualizations [ZCM05] or matrix representations for dense clusters within an aggregate node-link visualization [HFM07] (Fig. 2.3).

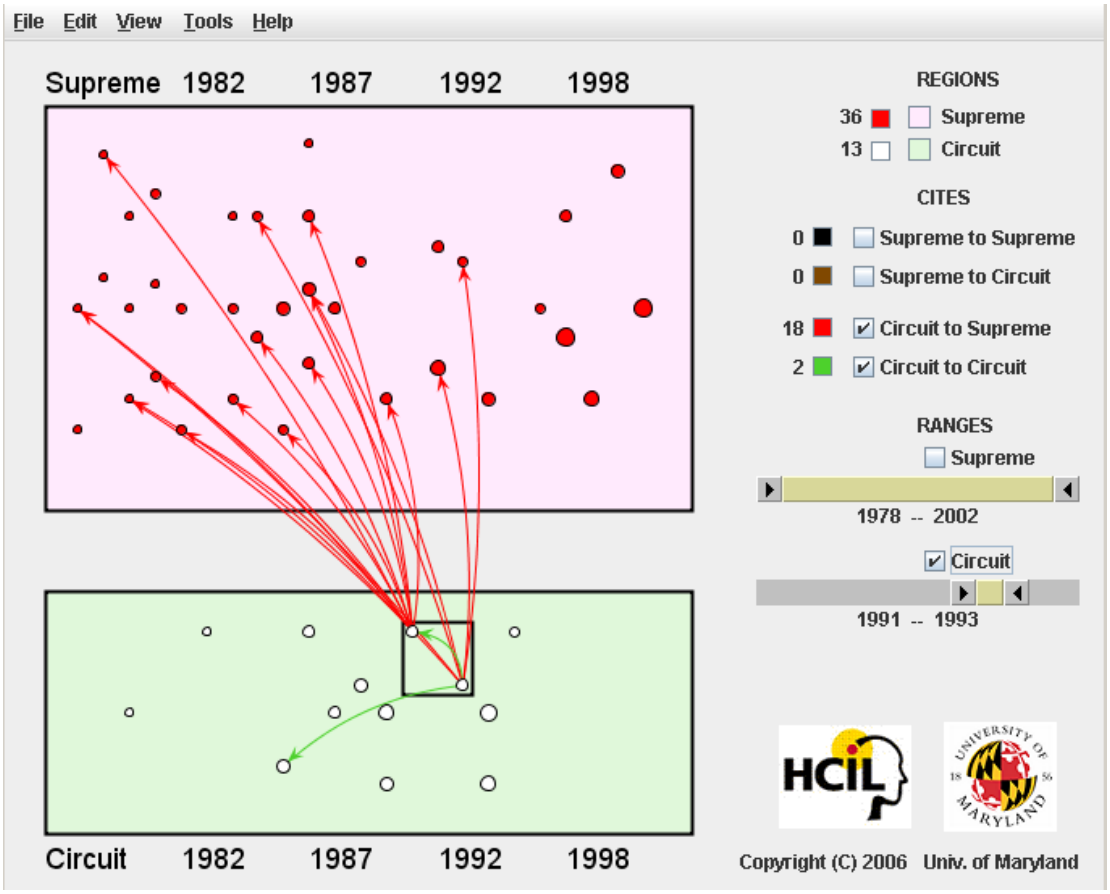


Figure 2.4: NVSS [SA06] showing citations from two Circuit Court cases in 1991-1993 to 19 Supreme Court cases and two other Circuit Court cases.

However, performing analysis of networks with many attributes remains a challenge with these representations, not to mention the difficulty for network overview and topology-based tasks.

There have been some recent attempts to specifically handle the attributes in multivariate networks. For example, ManyNets [Fre+10] (Fig. 2.6) allows users to partition networks according to attributes or topological properties, supporting fast comparison of the partition statistics, though it is difficult to extract patterns and

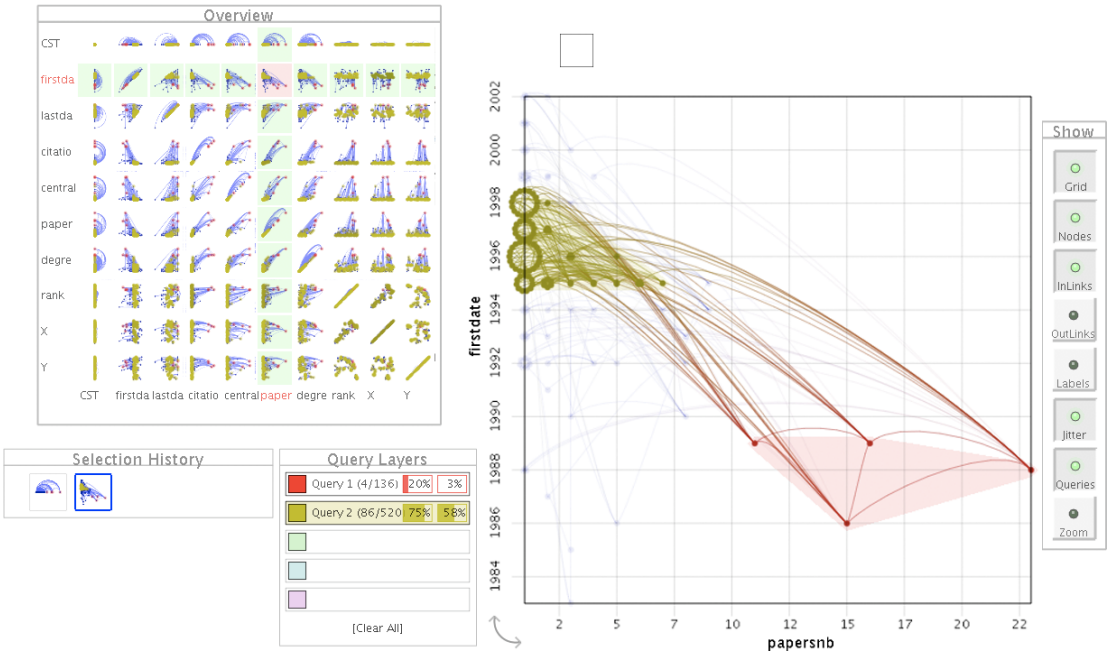


Figure 2.5: GraphDice [Bez+10] showing the InfoVis 2004 contest bibliographic network. The left shows the plot matrix window and the right shows the selected plot. The right view animates between selected plots.

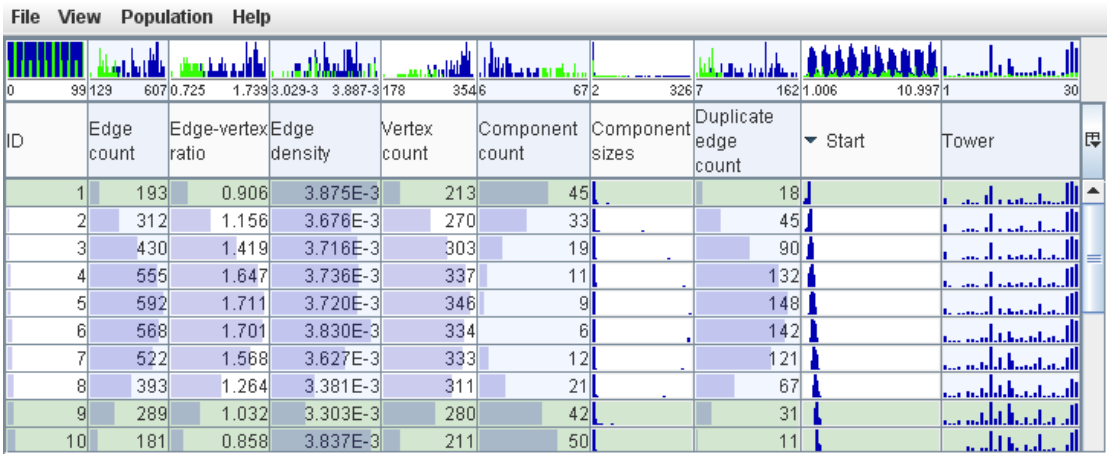


Figure 2.6: ManyNets [Fre+10] displaying the distributions of various statistics across subgraphs (rows).

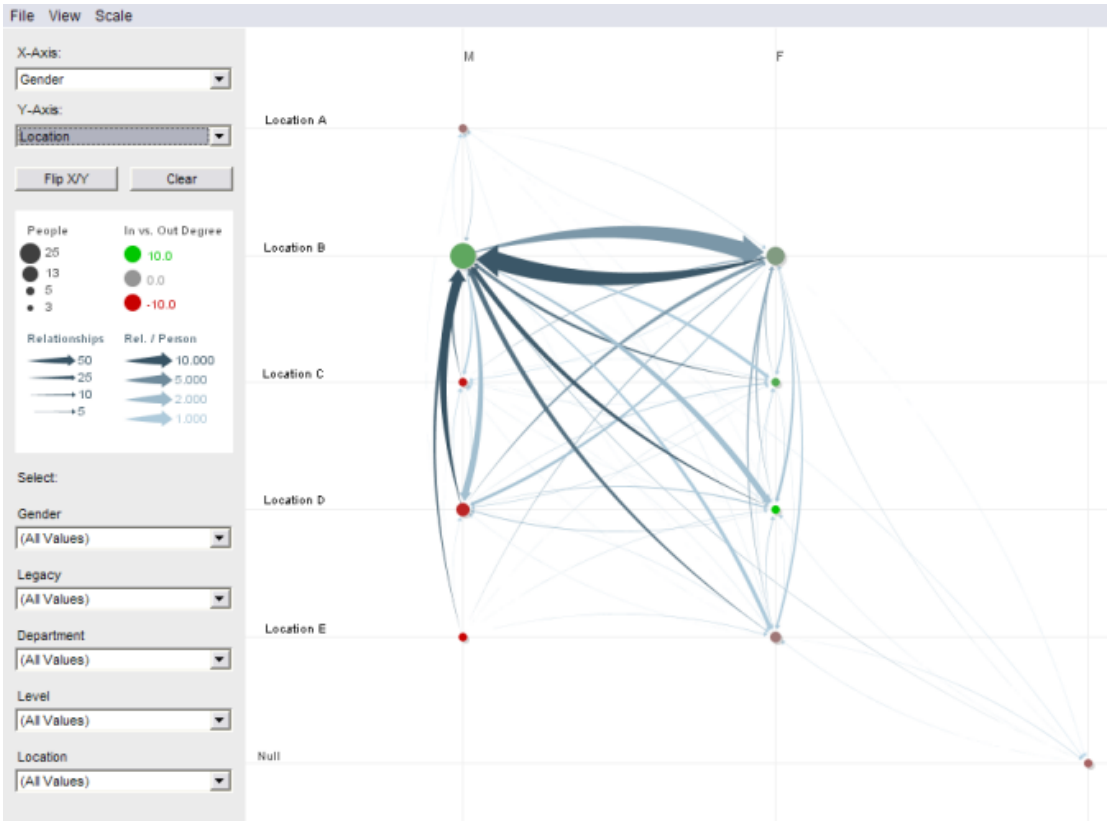


Figure 2.7: PivotGraph [Wat06] showing communication between aggregations of men and women (columns) and various locations (rows).

to identify relationships between the attributes. In contrast, PivotGraph [Wat06] (Fig. 2.7) aggregates nodes by attribute and indicates relationships between the aggregates using edges. However, it does not allow users to drill-down to see the details of the network and does not support comparing more than two attributes. Nor does it allow multiple types of nodes or edges (heterogeneous networks).

There have been many efforts to visualize heterogeneous and multivariate networks. General faceted browsing systems such as FacetLens [Lee+09] can be used on networks with multiple types of nodes and multiple attributes. Nodes are

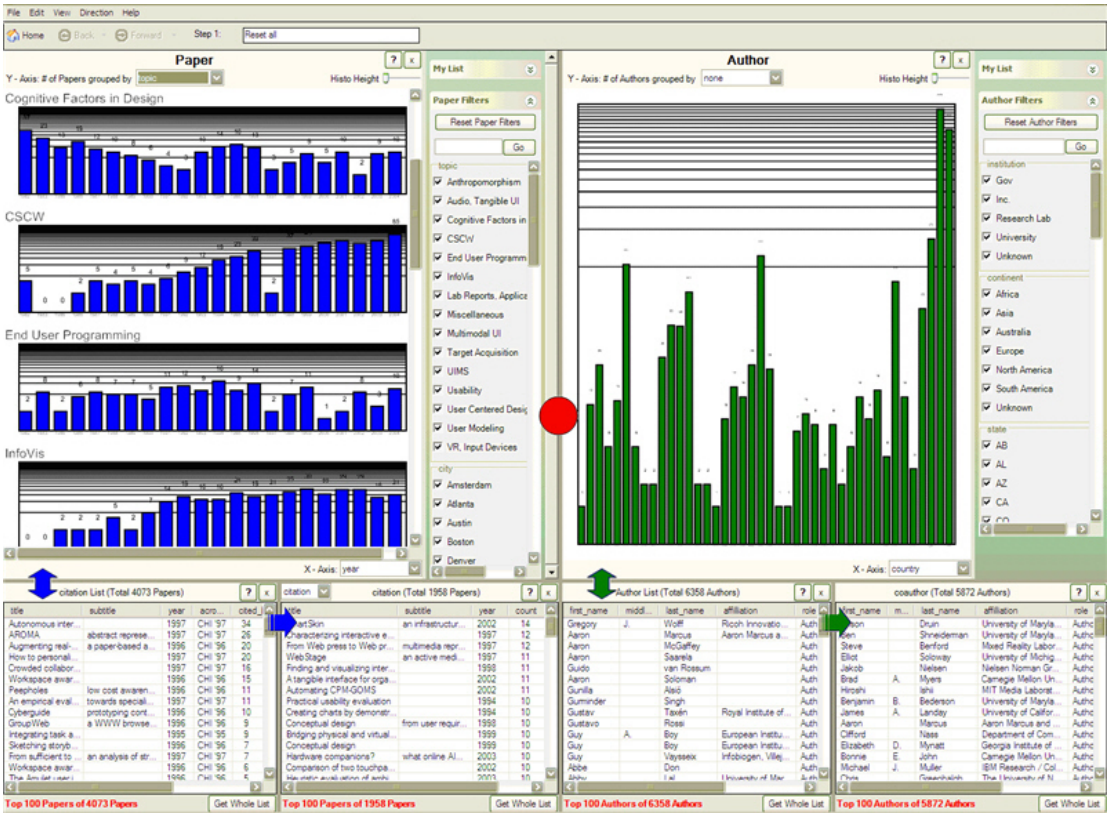


Figure 2.8: The main NetLens [Kan+06] interface here is showing ACM SIGCHI conference papers on the left and authors on the right.

grouped by their attribute values (i.e., facets) and users can pivot between node types, but only from a single node to its connected nodes. FacetLens helps users extract patterns and trends in the node attributes, but it does not explicitly represent the relationships between nodes. NetLens [Kan+06] (Fig. 2.8) is well suited to handle content-actor networks with two node types. It uses two coordinated views, each containing nodes aggregated according to their attributes. Users can explore the network by filtering in one view and pivoting from their filtered subset to connected nodes in the other view. NetLens allows for complex analysis

scenarios and extraction of trends and patterns in multivariate content-actor networks, but is limited to two node types at a time. Alternatively, my GraphTrail approach, which I discuss in Section 3.3.2, supports attribute exploration across many different node and edge types.

All these techniques I have discussed are effective for exploring networks based on their attributes, especially for heterogeneous networks. However, none of them are as effective as standard node-link visualizations for showing the overall topologic structure of a network and for helping users perform path-based tasks. However, these visualizations can be combined with node-link diagrams in a multiple coordinated view system [NS00; BWK00], with brushing and liking to highlight the same data in each view. One example tool is Network Workbench [NWB06], which provides an impressive array of statistics, modeling, scientometric, and visualization algorithms for analyzing bibliometric datasets. Unfortunately these visualizations lack brushing and linking and are weakly integrated into the rest of the exploration process. Examples of systems that do a better job of this include my GraphTrail and Action Science Explorer (Sections 3.3.2 and 3.3.3).

2.3 Measuring Node-Link Visualization Readability

There is a substantial body of work aimed at developing and, more recently, empirically verifying the correctness of a wide variety of readability metrics (RMs),

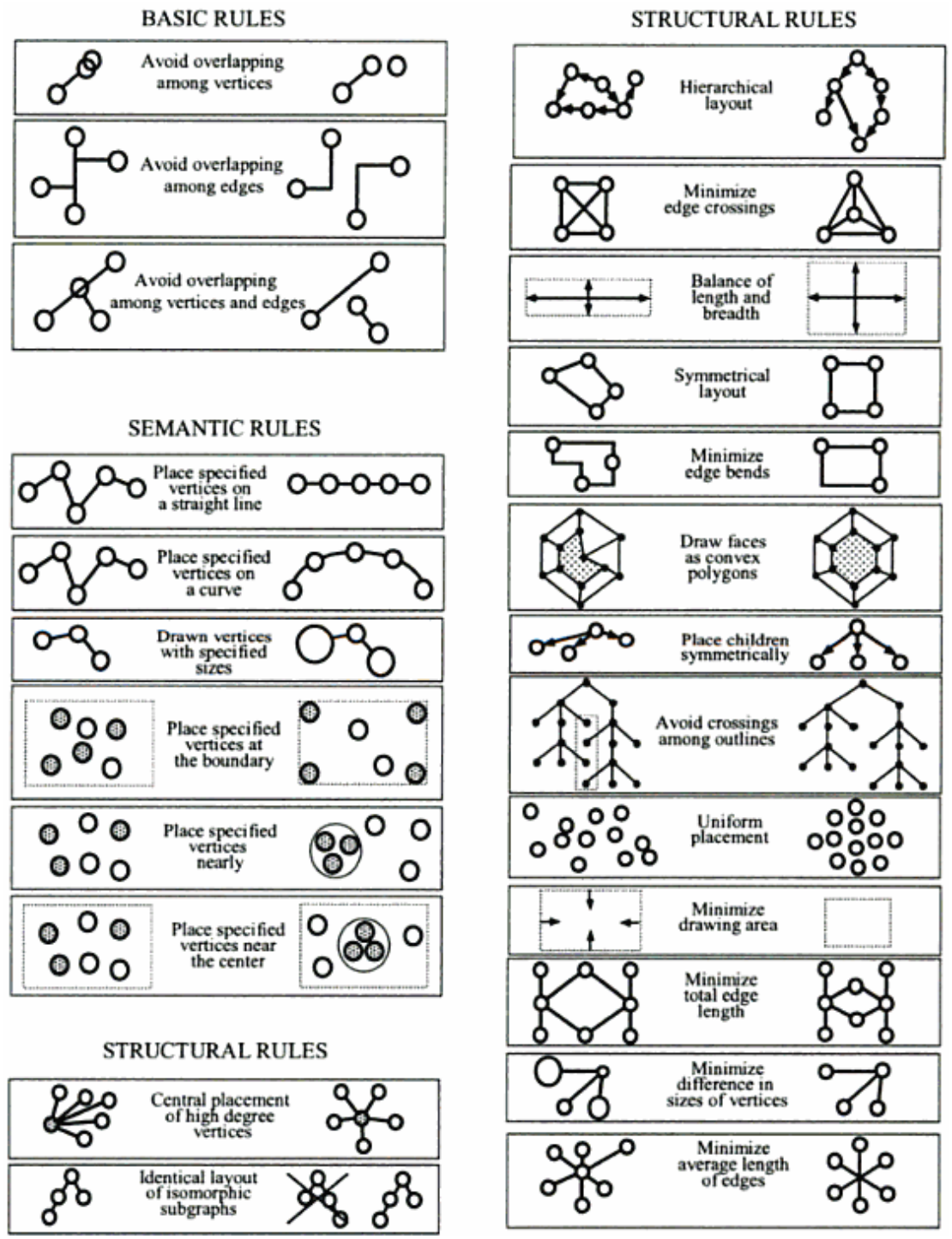


Figure 2.9: Simple rule-based drawing optimizations shown in Figure 2.3.1 of [Sug02, p. 14].

or, as they are often called, aesthetic criteria. Sugiyama's book [Sug02] includes a figure showing several simple rule-based drawing optimizations, replicated here in Fig. 2.9. Excellent overviews of RMs for general graphs can also be found in [Bat+98; War04; Bat+94; BFN85]. RMs specific for trees and UML diagrams are described in [WS79] and [Eic03], respectively. The first standard and numerical definitions of many specific RMs were given by Purchase and Leonard [PL96] and were elaborated on by Purchase [Pur02] who developed seven specific RM formulas. These will form the basis for much of my work.

Previous work in this area primarily deals with RMs for the entire graph drawing, giving, for example, a count of the total number of edge crossings. I name such RMs for the entire drawing as **global readability metrics**, or global RMs, and have developed several that are not included in the literature. Section 6.4 provides a detailed background for several global RMs, including Edge Crossing, Edge Crossing Angle, and my new Node-Node Overlap, Node-Edge Overlap, and Group Overlap metrics. Several other global RMs are discussed there in less detail, though many have citations to prior work in the area. These serve as excellent measures for how understandable the whole graph drawing is, but do not provide the level of specificity needed to direct users to problem areas. To address this problem, I augment several existing and my new global RMs with novel **local readability metrics** for individual nodes and edges.

Several layout algorithms try to directly satisfy readability metrics, such as using simulated annealing to distribute nodes evenly, make edge-lengths uniform, minimize edge-crossings, and keep nodes from coming near edges [DH96]. However, most layout algorithms use simple heuristics instead. Moreover, no sufficiently fast automatic layout techniques exist to leverage these metrics to create better general node-link visualizations. Rather than try to combine these metrics in a computationally expensive layout algorithm, I develop an assistive user feedback technique to help users optimize their layout manually using local RM calculations.

2.4 Motif Simplification

We can reduce the visualization complexity by showing an aggregate version of the network, based on any number of criteria. NetLens [Kan+06] (Fig. 2.8) groups nodes by their attributes and can pivot between connected groups of two different types, while PivotGraph [Wat06] (Fig. 2.7) uses attribute groupings but shows ties between aggregates using arcs. One of my techniques, GraphTrail (Section 3.3.2), combines these approaches with familiar charts, arc diagrams, and a many-to-many pivot between several node types. However, these approaches focus on attribute comparisons at the expense of showing topology, as I discussed in Section 2.2. Alternatively, my motif simplification approach retains all topology information in the overview visualization by using glyphs for specific motifs.

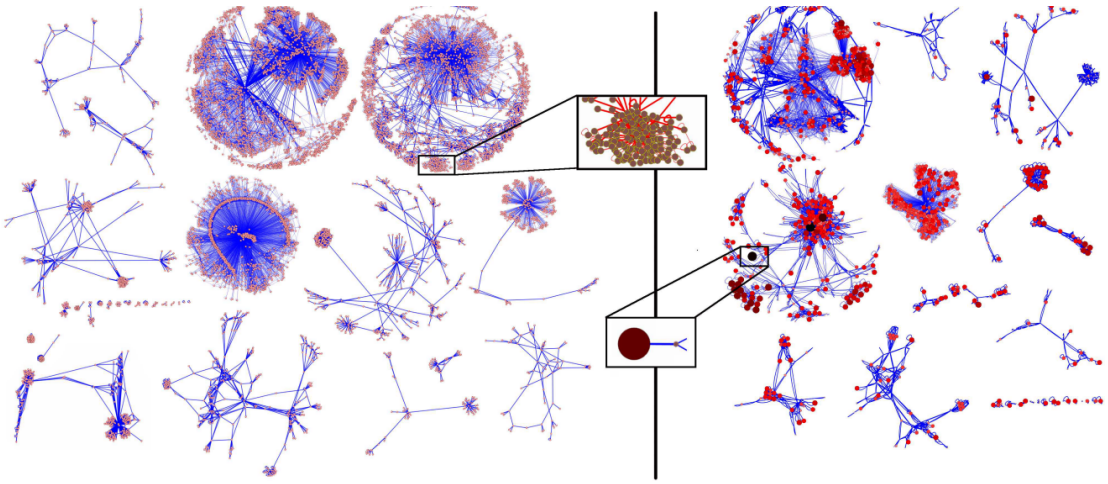


Figure 2.10: Greedy graph summarization technique applied to the CRN-10k graph. From [NRS08].

Instead of attribute aggregation, we can use a hierarchical topologic clustering to show a topologic overview in a network of meta-nodes like ASK-GraphView [AHK06] or van Ham & van Wijk [HW04]. Rather than letting meta-nodes overlap, van Ham & van Wijk used semantic fisheye views to show clusters as merging spheres. Other approaches to creating overview networks include graph summarization [NRS08] (Fig. 2.10) and aggregating nodes by shared neighbor sets [LSS12]. Liao, Shi, and Sun [LSS12] also provide a topologic clustering tool, and a level of detail option to split meta-nodes apart to better see the underlying topology. ManyNets [Fre+10] (Fig. 2.6) takes a different approach, showing statistical comparisons of a network partitioned by topology, attributes, or time. These techniques can show the aggregated topology of networks with hundreds of thousands of nodes, but not the underlying topology which is important for users to under-

stand the network structure. Often this is because of the ambiguous nature of clustering algorithms, in contrast to the exact motif detection algorithms I developed for motif simplification. Moreover, these tools do not present aggregate attribute information on nodes, unlike my motif glyphs.

Alternatively, we can filter to an important subset using a metric for node importance. Skeletal images [Her+99] highlights high-metric nodes, and replaces filtered trees with triangles that take the same space. Motif simplification, instead, aims to reduce the space required by the network in the visualization and allow additional layouts. Tsigkas, Thonnard, and Tzovaras [TTT12] similarly filtered a security network of events and features on a domain-specific metric, while including a way to aggregate the events joining a subset of features into meta-edges. However, the aggregation is limited to ties between two feature types and obscures the number of connecting nodes and edges.

My approach is to instead aggregate the network by the frequently occurring motifs it contains. While the fan, connector and clique motifs I target are quite prominent in social network datasets, there are many other motifs of interest, especially for biologists. Motif census (counting the kinds of motifs) and analysis is used extensively to analyze the behavior of complex biologic networks, looking for repeated patterns that indicate underlying processes. For example, Milo et al. [Mil+02] used an approach that finds motifs that appeared more frequently than

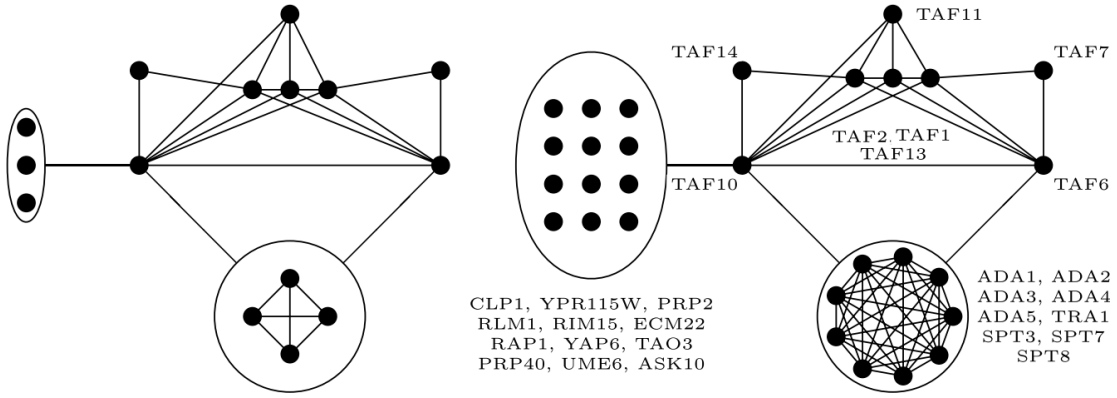


Figure 2.11: An interesting motif found in the protein-protein interaction network of *S. cerevisae*, a species of yeast. It appears 27,720 times, though these motifs all overlap and share the same set of 29 nodes. From [GK07].

expected in suitably random networks. They provide an extensive chart of motifs of three or four nodes, and describe their frequency in various biologic networks. Also, Zhu, Gerstein, and Snyder [ZGS07] provides an overview of the use of network motifs for analyzing biologic networks. Luscombe et al. [Lus+04] and Ye et al. [Ye+05] both demonstrate the applications of motif analysis for understanding biologic processes. To look for motifs larger than three or four nodes, Grochow and Kellis [GK07] developed a technique called symmetry-breaking that quickly finds motifs of various sizes. In applying their algorithm to the protein-protein interaction network of *S. cerevisae*, a species of yeast, they discovered one motif that appeared 27,720 times but does not appear at all in suitably created random ensembles. This motif, shown in Fig. 2.11, is composed of various overlapping combinations of 29 nodes that represent cellular transcription machinery. For my three motifs, I had to develop my own algorithms to scale well to large motifs.

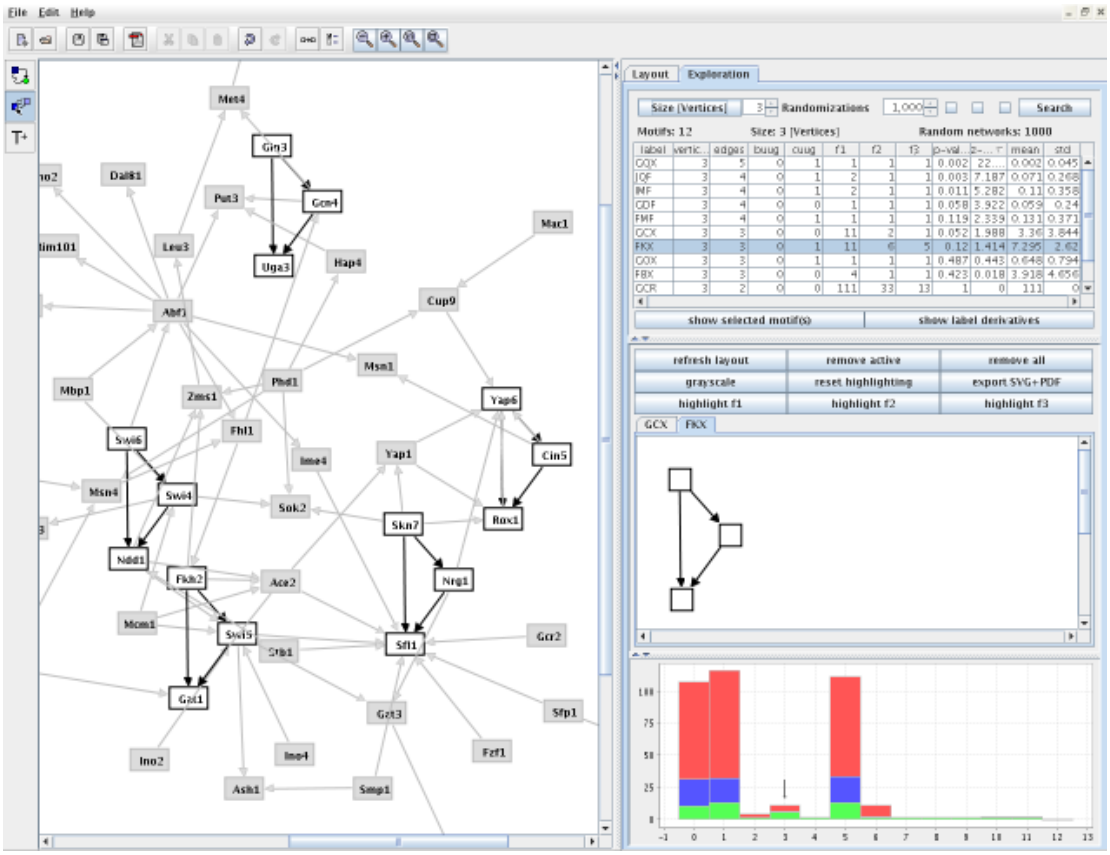


Figure 2.12: In MAVIsto [KSS06], matches for a particular motif like the feed-forward loop are laid out aligned the same direction and highlighted. The bar chart shows how frequently particular motifs occur above expected levels.

Knowledge of the motifs present in a network can help predict behavior and the "structural signatures" of individual entities [Wel+07], but visualizing these motifs effectively is challenging. Huang et al. [Hua+05] detect motifs with fewer than five nodes and draw transparent convex hulls to highlight them. Similarly, Klukas, Schreiber, and Schwöbbermeyer [KSS06] take the matches to a chosen 3–5 node motif and color them within the overall visualization and draw them identically to be easily spotted (Fig. 2.12). While highlighting the motifs can help biologists

spot the locations of particular processes, it does little to reduce the clutter of a complex network drawing and can even reduce the readability. Instead, my motif simplification work directly tries to reduce this clutter by replacing motifs with representative glyphs.

In contrast to motif simplification, current approaches to reducing complexity aggregate nodes based on their attributes, topology, or metrics but do not provide visible indications on the meta-nodes showing the underlying topology. Moreover, these algorithms usually pay little attention to the motifs present and create a grouping with ambiguous topology. While current tools can highlight small detected motifs, there are few techniques for providing a graphical overview or summary of them. More importantly, I know of no approaches other than motif simplification that leverage the motifs present to reduce the visual complexity of the network visualization.

2.5 Meta-Layout

Much of the work on meta-layouts has focused on so-called multiscale layouts, which attempt to take more structure of the graph into account for the layout than plain force-directed techniques. For example, Large Graph Layout [Ada+04] iteratively moves down a minimum spanning tree placing children on spheres around parents. This results in beautiful static images such as the map of the Internet in

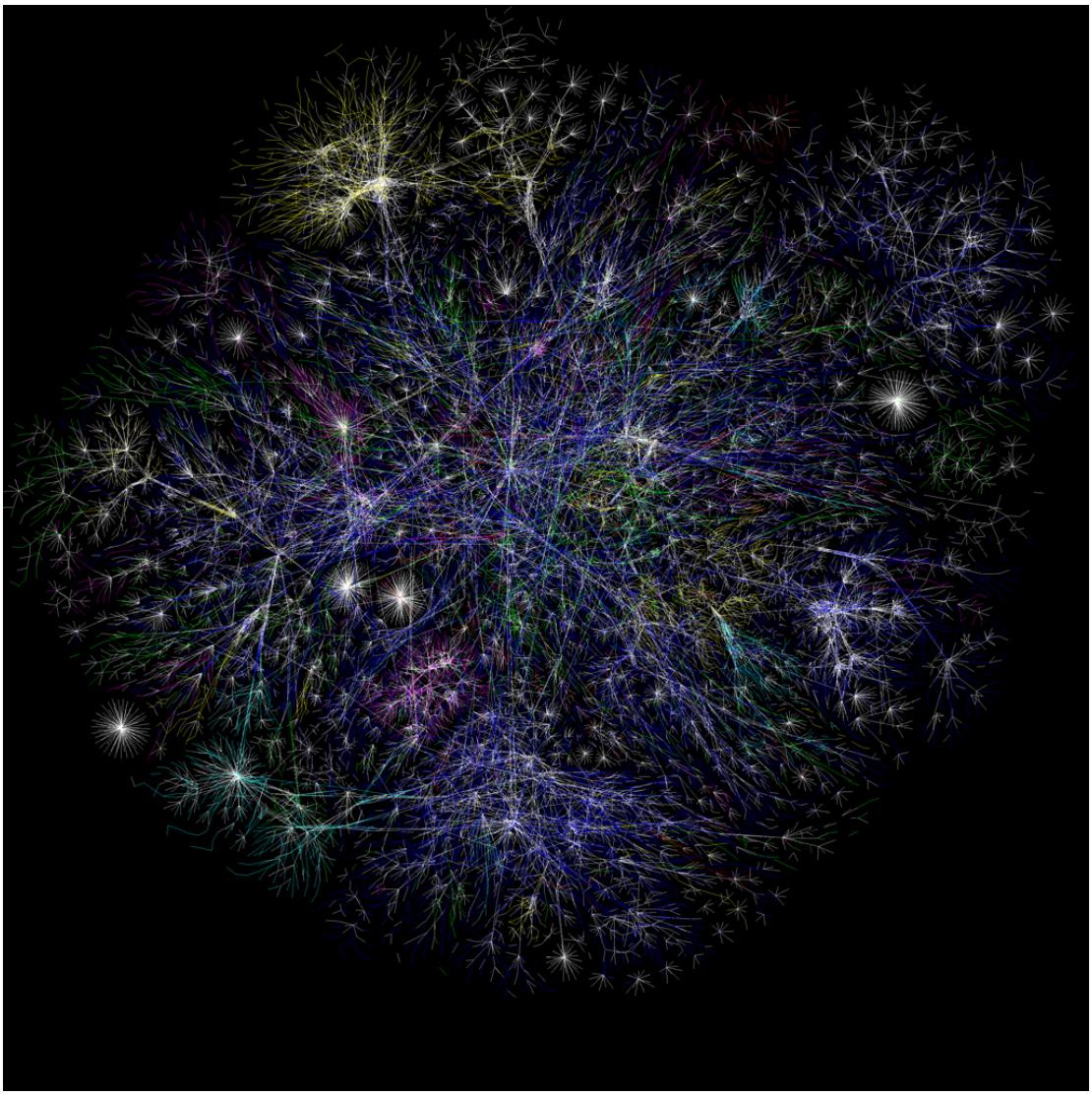
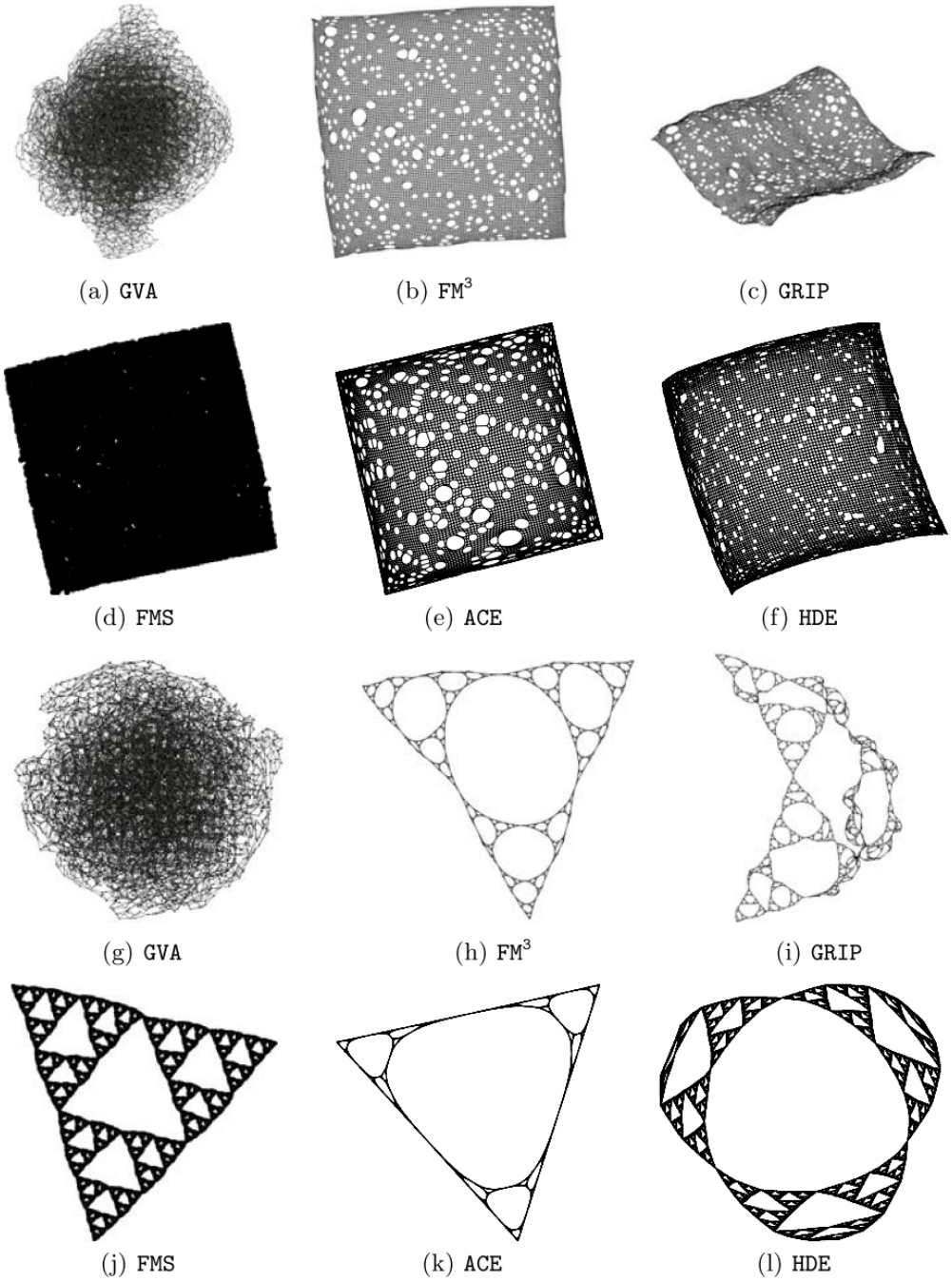


Figure 2.13: Large Graph Layout [Ada+04] rendering of the internal structure of the Internet (as of 2005). From opte.org/maps

Fig. 2.13, though it is hard to see topology and near impossible to see attributes at the scale of networks they tackle.

Other examples of multiscale layouts include a Cytoscape plugin by Salmela, Nevalainen, and Aittokallio [SNA08], the Harel-Koren FMS layout [HK02a] used in NodeXL [Smi+10], and many others (e.g., [HJ05; Wal01; Won+08; GGK04]). One effective approach, the Lin-Log layout [Noa04], takes explicit cluster membership into account when computing the node positions. Hachul and Jünger [HJ06] provide an experimental comparison of six multiscale layouts on various toy datasets, such as the random grid and Sierpinski triangle shown in Fig. 2.14, as well as some real-world ones like the social network in Fig. 2.15. These multi-scale layouts can show the overall topology of the network well if they use enough screen space, but this "zooming out" prevents them from displaying internal group ties clearly. None of them, including the Lin-Log layout [Noa04] which takes clusters into account, highlight group sizes and internal structures as well as my Group-in-a-Box layouts.

One interesting meta-layout is a modification to Treemaps that attempts to map the boxes to known geographic locations [WD08]. These spatially ordered Treemaps can be effective for visualizing geographic data like the London tube network (Fig. 2.16). This could be potentially modified to use the the relative relationships of the groups rather than the geography. However, I chose to allow some screen space to be "wasted" to show the ties between groups more clearly



(a) GVA (b) FM^3 (c) GRIP

(d) FMS (e) ACE (f) HDE

(g) GVA (h) FM^3 (i) GRIP

(j) FMS (k) ACE (l) HDE

Figure 2.14: An experimental comparison of six layout algorithms on a random grid and Sierpinski triangle dataset, discussed in [HJ06].

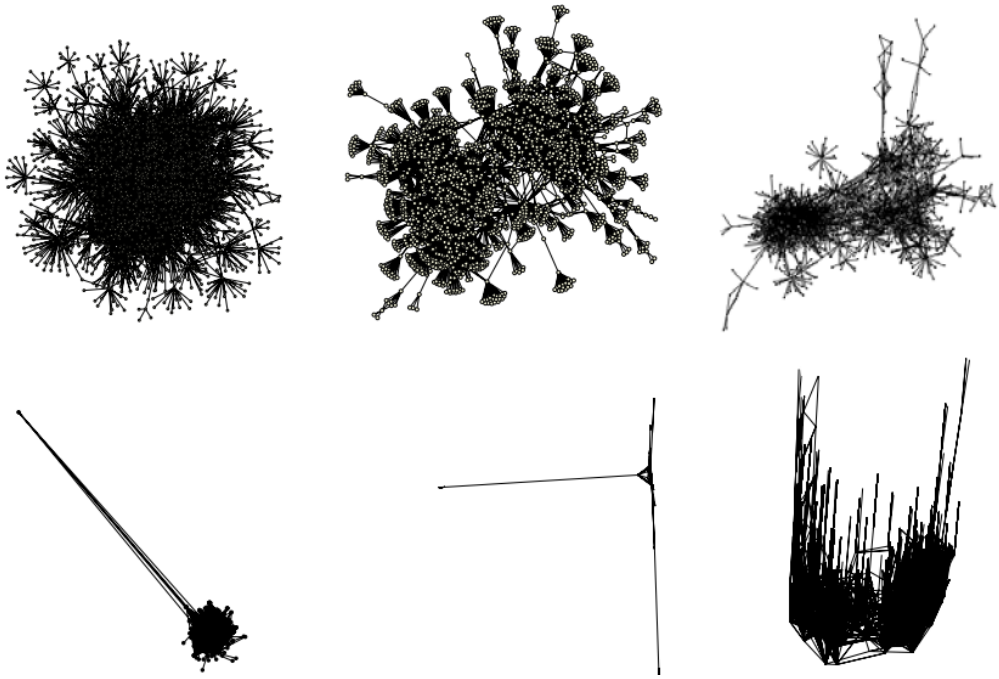


Figure 2.15: An experimental comparison of six layout algorithms on a social network dataset produced widely different layouts. From [HJ06].

instead of using a Treemap algorithm. Another meta-layout is DICON [Cao+11] (Fig. 2.17), which uses Treemap-like icons to represent clusters. In addition, it uses a layout algorithm for the icons that generate similar icons for similar clusters. This approach would potentially do well with hierarchically clustered networks instead of a one-level hierarchy like the Group-in-a-Box layouts use, but does not display the internal group structure nearly as well as the Group-in-a-Box layouts.

One option is to use edge bundling rather than aggregating the underlying edges like I commonly do with my Group-in-a-Box layouts (e.g., Fig. 1.7). Since large numbers of links that span a graph drawing can undermine readability, there has been a strong attraction to edge bundling to reduce clutter [Hol06; Pup+11].

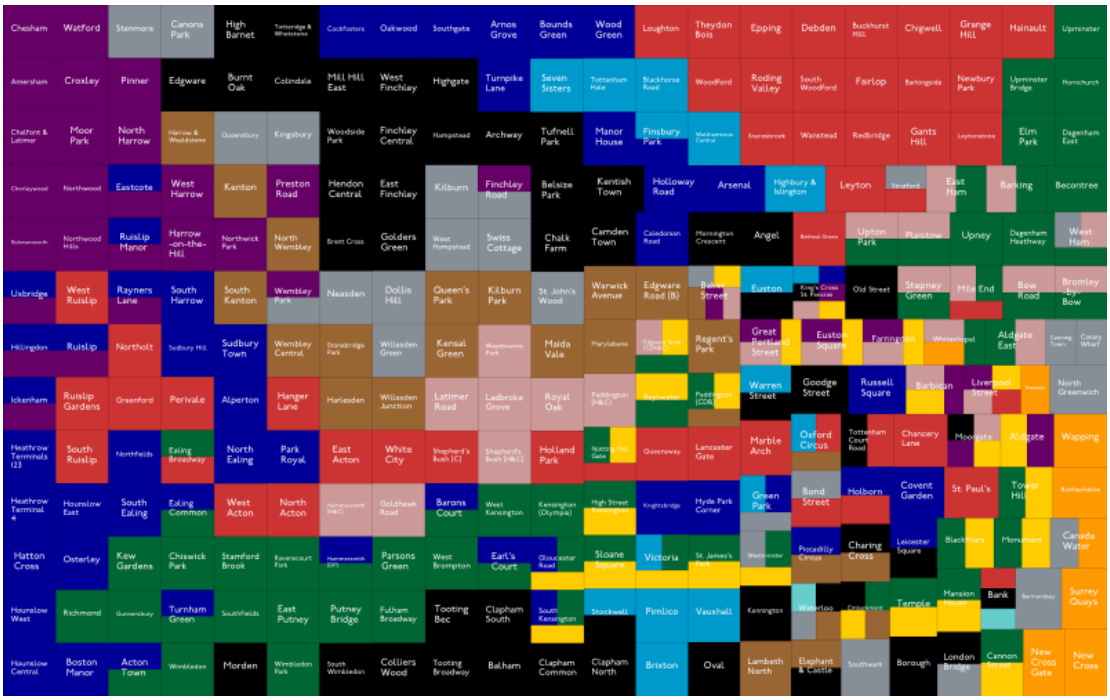


Figure 2.16: Spatially ordered Treemap [WD08] of the London tube network. Stations (squares) are colored by the lines they serve.

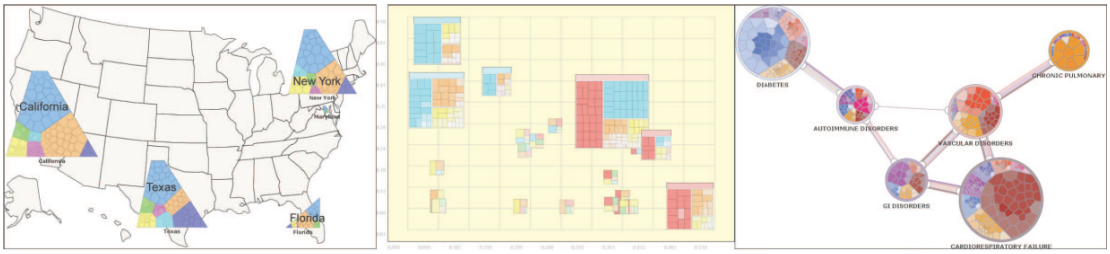


Figure 2.17: DICON [Cao+11] showing Treemap-like icons for clusters.

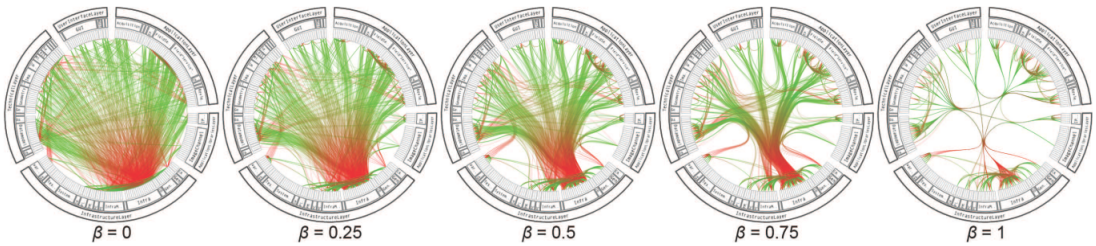


Figure 2.18: Increasing strength of edge bundling going left to right. From [Hol06].

NodeXL [Smi+10] currently supports several levels of edge bundling, and an example of these increasing levels is shown in Fig. 2.18. The initial view is attractive, but the bundles seem to obscure rather than highlight the strength of relationships among the clusters. However, the option is available to users.

2.6 Evaluation

Evaluating the effectiveness of complex creativity and exploration tools can be challenging. Simple usability issues can be collected as participants express confusion or difficulties, and can even be iteratively used to improve the system throughout the user study [Med+02; Med+05]. I applied these techniques in the development of my three network visualization improvement approaches. However, the scope of the features used and the intellectual effort required for exploration render quantitative laboratory techniques infeasible for capturing many important aspects of the tool usage [CC00]. For a recent overview of these techniques, see [Lam+11; PFG08].

One way that individual tools can be analyzed and compared with others is based on the insights into the data users find with them, where what constitutes an insight is rigorously defined [Nor06; SND05; Sar+06]. Alternatively, Shneiderman and Plaisant [SP06] make the argument that qualitative evaluation methods are becoming common, accepted, and effective techniques for analyzing visual analytics

tools. Excellent examples of these qualitative evaluation techniques for longitudinal studies are demonstrated by [PS09; PS08a; SS06].

For my work, I predominantly use more conventional task-based studies and experimental evaluations, as the approaches I am suggesting are more directly comparable to the current state of the art node-link visualizations. Lee et al. [Lee+06] provide a task taxonomy for network visualization, which I leverage in my studies (e.g., see my evaluation of motif simplification in Section 4.5). The tasks I chose are also used in many recent papers evaluating network visualizations [HF07; SA06; GFC04]. Also, there is a substantial amount of work on user perception for experimental metric-based studies, including [Pur02; War+02; Hua07b; BMK96]. However, this is beyond the scope of my work.

2.7 Summary

There are many approaches for visualizing networks, the most common being node-link visualizations which are very effective for visualizing the overall network topology. Unfortunately the effectiveness and perceived meaning of a node-link visualization is highly dependent on the layout of nodes and edges. Readability metrics exist to quantify the effectiveness of a static drawing, but do not identify specific problem locations. While several layout algorithms try to directly or indirectly optimize for these metrics, they are often marginally effective or only

useful for specific tasks for which they are optimized. Moreover, there are no user-controllable layout algorithms or assisted layout techniques based on the metrics. My work contributes new global readability metrics, as well as local readability metrics to direct users to problem areas. I leverage these local readability metrics to create an interactive layout improvement technique that guides users using visual metric feedback.

It is challenging to use node-link visualizations to analyze large, multivariate, and/or heterogeneous networks, and one of the most effective approaches is to use aggregation by topology or node and edge attributes. Effective aggregation is difficult to do well while preserving the underlying aggregate topology. Aggregating by network motifs has not been explored yet, nor has using representative glyphs for the resulting meta-nodes. While aggregation by toplologic and attribute clustering has been done in node-link visualizations, the resulting groups have only been used to improve the layout of inter-group relationships. My Group-in-a-Box layouts can show inter-group relationships, but also group size by their bounding regions as well as internal group structure.

Chapter 3

Applied Network Visualization

3.1 Introduction

This chapter serves two purposes. First, it describes in detail NodeXL [Smi+10], which is a free and open source network analysis tool that drops into Microsoft Excel. I cover why I chose to implement many of my dissertation techniques as part of NodeXL, as well as my many contributions to NodeXL's design, development, and evaluation (Section 3.2). Second, this chapter provides an overview of some of my work on applying network analysis principles to various domains and real-world problems (Section 3.3). It is from these applications that I gained an understanding of what approaches are effective for displaying networks visually, which interaction techniques are useful for exploring them, and what major challenges remained. Moreover, I learned about the necessity for designing exploration tools for end user tasks, as well as how to leverage powerful Computer Science and statistics techniques and present the algorithmic results to users. These lessons guided my dissertation work, and will continue to assist me in my future design challenges.

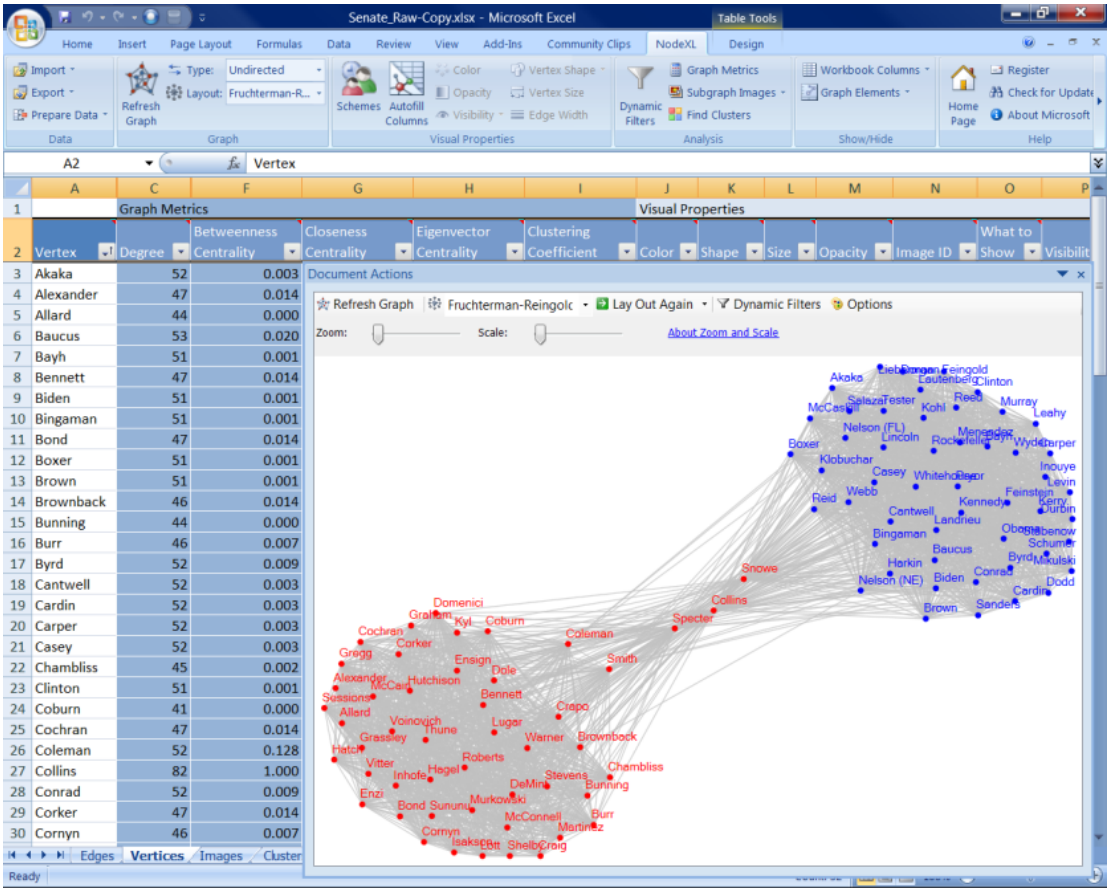


Figure 3.1: The NodeXL [Smi+10] workspace. The dual pane view of network data and metrics (left pane) with node-link visualization (right pane) provide an integrated snapshot of statistics and visualization, along with built-in functions and controls that support exploration and discovery. Individual worksheets separate network analysis tasks into separate categories, closely aligned with topology and attribute-based tasks, such as "Edges", "Vertices" (nodes), and "Groups." The social network shown reflects voting patterns of U.S. senators, analyses of which are detailed in [PS08a; PS09], as well as Sections 3.3.1 and 4.3.1.

3.2 NodeXL

NodeXL [Smi+09; HSS11; Smi+10], shown in Fig. 3.1, is a free and open source network analysis add-in for Excel 2007/2010/2013. NodeXL is tailored to provide

powerful features while still being easy to learn. The Excel integration allows rapid data processing using standard formulas and macros, but NodeXL also provides calculators for network statistics, automatic layout algorithms, visual attribute encodings, dynamic filters, direct manipulation, coordinated views, and importers from online social networks and common network file formats like GraphML, Pajek, and UCINET. These importers are especially important for helping novice users collect datasets that are of interest to them like Twitter keyword searches, their Facebook network, or their personal email collection.

NodeXL is widely used in many disciplines and has a full-time developer as well as a team of volunteer advisors and developers. Over 25 introductory courses on network analysis have used NodeXL and its companion book [HSS11] as part of their curriculum,[1] due mainly to its ease of use, open source nature, and design focus on novice users. I myself have taught several tutorials on using NodeXL for network collection and analysis.

3.2.1 Contributions to NodeXL

I have been involved with the NodeXL project since 2008 as an advisor, developer, and by running exploratory user studies that show that novice network analysts can effectively explore datasets with NodeXL [Bon+09]. Moreover, many of the tech-

[1] nodexl.codeplex.com/wikipage?title=NodeXL%20Teaching%20Resources

niques I present in this dissertation are implemented and made publicly available in NodeXL. My motif simplification approach detailed in Chapter 4 is currently shipping in NodeXL for anyone to use and build upon. Of the Group-in-a-Box layouts I have worked on (Chapter 5), the Treemap GIB layout is already available in NodeXL. The Croissant-Donut and Force-Directed variants have been implemented and I will push them to the trunk shortly. Finally, some of my readability metrics and the assistive layout improvement tool (Chapter 6) are implemented as a hidden feature and may be released in the future when we can devote additional time to readying them for public consumption.

I chose to develop my techniques within NodeXL for several reasons. First, NodeXL is a high quality network analysis tool with a large, active, and expanding user base. It has over 184,000 downloads and is on an increasing trajectory. Moreover, there are about 660 query results for "NodeXL" on Google Scholar, many of which are papers applying NodeXL to network analysis challenges in various domains. Second, given its role as a teaching tool, many NodeXL users generally have little prior knowledge about network visualization readability. I believe that these novice users will particularly benefit from my readability-improving techniques. Moreover, the NodeXL codebase is separated into the classes necessary for the interactive Excel template and a disjoint set of generally applicable code that is packaged as a separate C# network analysis library. Users of this library

have access to many of the algorithms behind my techniques without having to do the implementations themselves. Finally, NodeXL's free availability and open source license encourages collaboration and provides a reference implementation for future users interested in applying or evaluating my techniques.

3.2.2 NodeXL Interface

The basic interface of NodeXL is shown in Fig. 3.1. The left side provides several worksheets in an Excel workbook that represents the network: one each for the nodes, edges, groups, group members, and overall metrics. Each worksheet has several columns, including basic information about the network like the nodes and edges between them. Additionally, there are places to insert columns for node or edge attributes and calculated metrics, as well as columns that control the visual display of each network item. These include color, shape, size, label, tooltip, display position, and the like. Any of these visual properties can be automatically filled based on the metric or attribute columns using a special autofill dialog. Moreover, standard Excel formulas or macros can be used for arbitrary calculations and scales within the tool. The Excel ribbon is customized with a new tab for many of the common operations users perform on networks, including the autofill feature.

The visualization pane shown in the right of Fig. 3.1 displays a node-link visualization based on the network in the workbook. Whenever the contents of

the workbook is updated, the visualization pane can be refreshed using a button. The pane also provides users with several automatic layout algorithms to arrange the network, and any automatic or manual adjustments to the node positions are stored in the workbook as well. Moreover, the contents of the visualization can be filtered using a dynamic filters dialog. Additional windows can be opened for filtering the visible network, autofilling visual property columns based on metrics or attributes, and running automated analyses of several networks sequentially.

The worksheet view and the visualization pane are connected using brushing, where any selection in one is reflected in the other. Clicking a node in the visualization or dragging a box around several causes the associated rows to be selected in the nodes worksheet. Likewise, any incident edges are selected in the edges worksheet. The reverse is also true. Any nodes or edges selected in the worksheets are highlighted in the visualization pane as well.

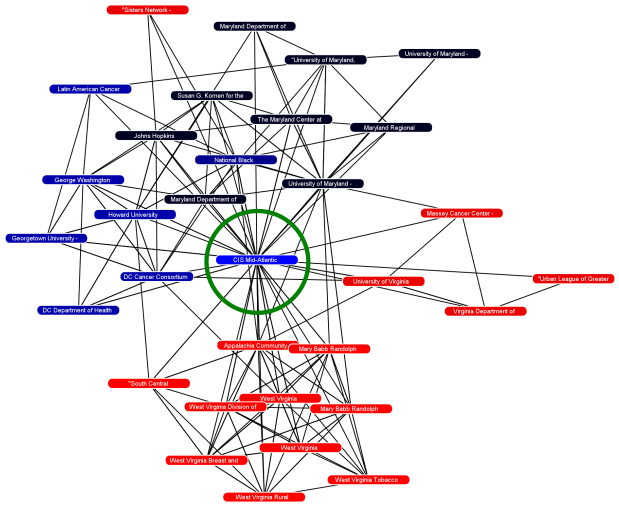
3.3 Applying Network Visualization to Real Problems

While much of my work has been on NodeXL [Smi+10], I have worked extensively with target users from several domains on visualizing and analyzing their real-world networks. I have been involved in network analysis projects for six years, and I strive to solve real problems by initiating contact with domain experts across many disciplines. I design and build visual analytics tools that have helped

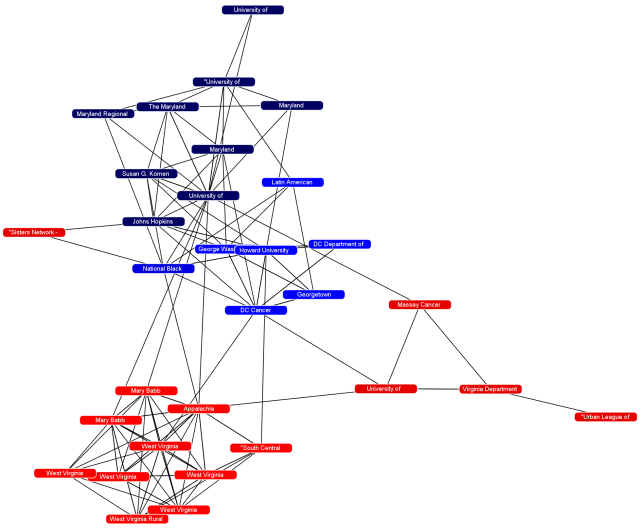
urban planners [SD12], political scientists [DS13], health care professionals, the U.S. Treasury, and many others described below. This work has helped me gain an understanding of the effectiveness of various visualization and interaction approaches, as well as what major research challenges remained. Moreover, it helped me to realize the importance of keeping the end users and the tasks they wish to accomplish in mind throughout the design process. The tasks end users wish to perform drastically impacts the effectiveness of any chosen visualization and interaction techniques. Often, some of the best breakthroughs for the end users came when I could integrate powerful Computer Science and statistical algorithms and present the results within the visualization or a coordinated view in the tool.

3.3.1 The Importance of Network Topology and Filtering

For some network analyses users are only interested in the topology of the relationships and not any additional attributes. For one such exploration, I visualized the relationships between 750 organizations that are engaged in cancer research, awareness, and outreach. The data used to create the network was collected through a survey of these organizations by a central agency, the Cancer Information Service (CIS) of the National Cancer Institute (NCI). Due to this selection method, the CIS played a central role in each of the networks, connected to each of the surveyed organizations. Many network datasets suffer from similar selection mechanisms,



(a)



(b)

Figure 3.2: Relationships between cancer research, awareness, and outreach in DC, MD, VA, and WV. The different colors represent each of the states in the region. (a) shows the network with the CIS ego node circled in green, while (b) shows the same network after removing the CIS node and laying it out again. The resulting visualization shows the remaining group structure and connections more clearly.

only showing the ego network of a person's Facebook friends, related replies or mentions on Twitter, or a set of connected web sites in a web crawl. In these sorts of ego-centric datasets, simple filters like removing the ego of the network can substantially improve the resulting visualization. For example, Fig. 3.2 demonstrates how removing the completely connected CIS ego node from the network for one region can substantially improve the layout and readability of the remaining nodes, with no loss of information.

Some networks have large numbers of nodes and edges which can obscure meaningful groups or network items with interesting attribute values. Filtering can be applied to node values to remove incidental nodes of specific types or with low metric values, leaving only key actors. User-controlled dynamic query filters [AWS92; WS92] have demonstrated their value in successful commercial products that deal with multivariate data, such as Spotfire [Spo] and Tableau [Tab]. Dynamic query filters are even more valuable in network visualizations, where the clutter of nodes and links can severely inhibit readability. NodeXL, discussed in detail in Section 3.2, supports filters on node values, link values, graph metrics, layout positions, and many other attributes.

Filtering is a well-established technique for multivariate data, as shown in scattergrams, but the variety of filters in many networks means careful thought is needed to produce effective results. Furthermore, scattergram filtering typically

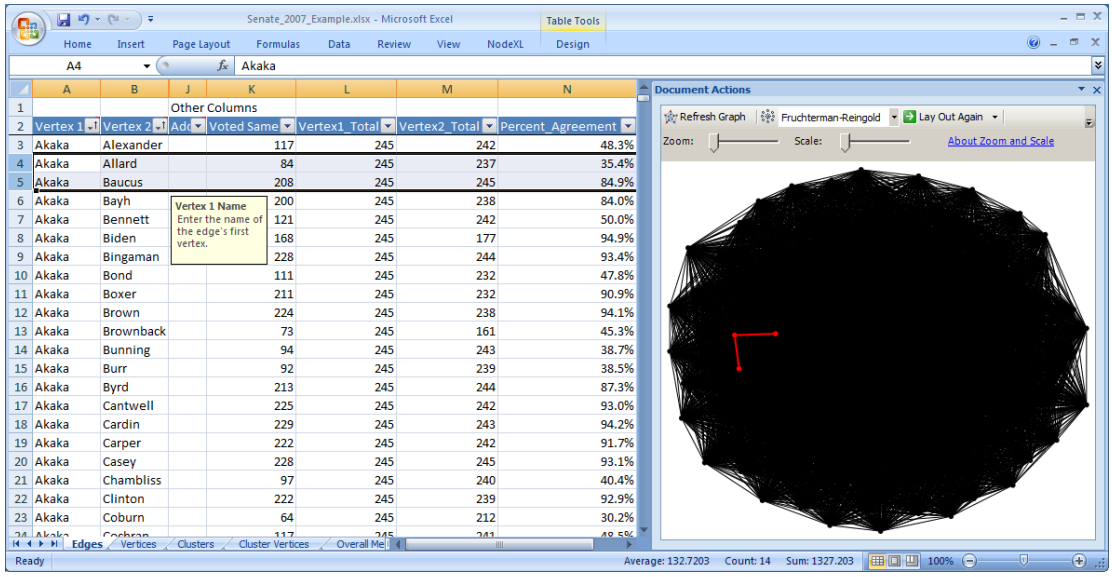


Figure 3.3: 2007 U.S. Senate voting network, showing all 4950 links. The network is visualized inside the NodeXL network analysis tool as part of Excel. The highlighted red edges show the Akaka–Allard and Akaka–Baucus ties.

leaves the remaining markers in place, but in networks, layout methods interact with filtering, so thoughtful exploration is needed.

The power of attribute or metric filtering is shown in an example network of U.S. Senate voting patterns from 2007.[2] The similarity in voting patterns (from 0.0 to 1.0) is an attribute of each one of the 4950 links connecting the 100 Senator nodes. The naive visualization produces a thickly connected graph (Fig. 3.3), but filtering the similarity values to show only those with values above 0.65 produces a revealing portrait (Fig. 3.1). The force-directed layout shows the willingness of the three Republican Senators Snowe, Collins, and Specter (center, in red) to vote

[2]Data provided by Chris Wilson of Slate magazine available in the NodeXL template format at nodexl.codeplex.com/wikipage?title=NodeXL%20Teaching%20Resources

in support of their Democrat colleagues (top-right, in blue). One of these, Arlen Specter, later switched his affiliation to the Democrats in 2009. However, apart from the party groups and these moderates, not much of the network structure is visible inside the dense party clusters. This data is further explored in Section 4.3.1.

As these filtering operations omit information from the visualization, it becomes important to keep track of what was omitted. While my GraphTrail approach detailed in Section 3.3.2 was designed to present the history of exploration automatically, most network analysis tools do not give you any indication that data has been removed. This prompted me to think about ways that nodes in larger datasets could be automatically filtered, but displayed in such a way as to notify the user what filtering has taken place and display the underlying node distributions. This is especially important for ego-centric datasets like social network crawls or web crawls like discussed in Section 4.3.4, where there can be an enormous amount of peripheral data that can obscure the core relationships. This line of thought helped guide me in the creation of the fan and connector motif simplification approaches described in Chapter 4. Similarly, in the Senate example above the importance of edge filtering was highlighted to me. The clique motifs simplification technique I develop in Chapter 4 is based on this kind of edge filtering, and I even apply it to the same Senate dataset in Section 4.3.1.

3.3.2 The Importance of Node & Edge Attributes

Some network datasets and analysis tasks require less focus on the topology and more on the node and edge attributes. One of my studies focused on the network of relationships formed by IP traffic on a local area network (LAN) [Blu+08]. The visualization tool we designed, NetGrok, is targeted at system administrators monitoring the status of their LAN. While the LAN topology was important for users to view, the topology of the connections with remote machines was less likely to be observed in the packet capture or relevant to the users. We focused instead on showing changes in communication patterns that could indicate malicious or erroneous behavior on the LAN.

The approach we developed for this challenge focused on presenting aggregations of the connection attributes over time such as the bandwidth used and total number of connections. Two of the views of NetGrok are shown in Figs. 3.4 and 3.5. In the node-link view (Fig. 3.4), the relationships between computers on the LAN are shown using a force-directed layout in an inner circle, while remote computers are arranged in a hash layout based on one of their attributes: their IP address. Connections to external computers were hidden by default due to their number and relatively low meaning, but shown on demand. An alternate view replaced the node-link visualization with a treemap as in Fig. 3.5, where each relationships are similarly shown on demand.

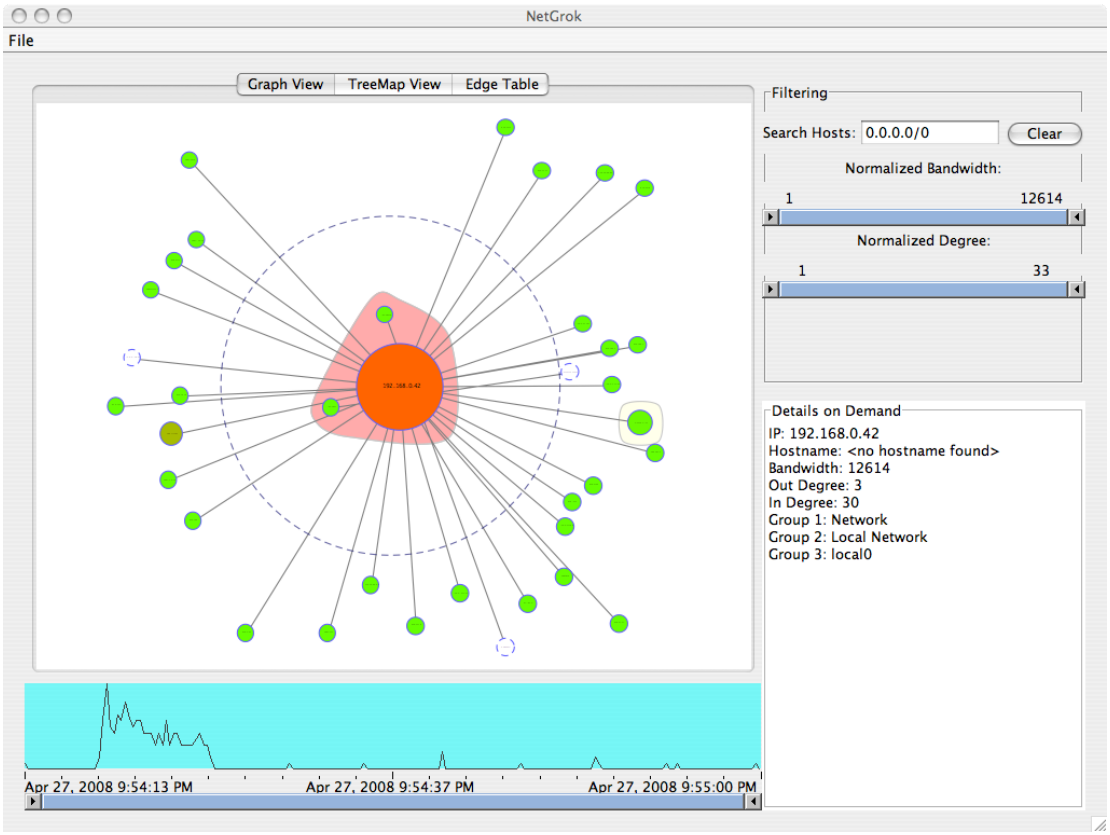


Figure 3.4: NetGrok's [Blu+08] elements include a node-link visualization (upper left), a time-line histogram (lower left), a filter panel (upper right), and details on demand (lower right).

While individual nodes and their relationships can be of interest, in many cases it is the groups of nodes and their aggregate relationships that are more useful to study. One of my previous projects as an intern at Microsoft Research, called GraphTrail [Dun+12a; RLD] (Fig. 3.6), was targeted at more general networks and aimed to explore networks by aggregating node and edge attributes in standard charts. For example, the bars in the bar chart in Fig. 3.6 each represent an aggregate of nodes and the arcs along the bottom show the aggregate relation-

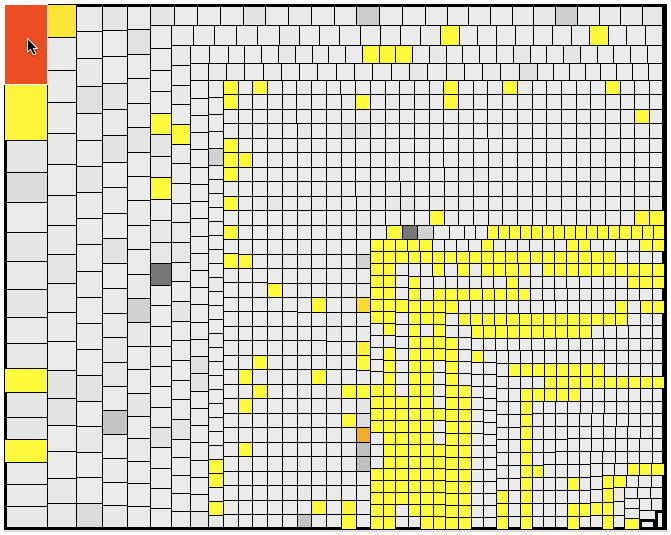


Figure 3.5: NetGrok's [Blu+08] treemap layout arranges computers by the number of connections they have and colors them by the bandwidth used. Communications between computers are shown using highlighting on mouseover.

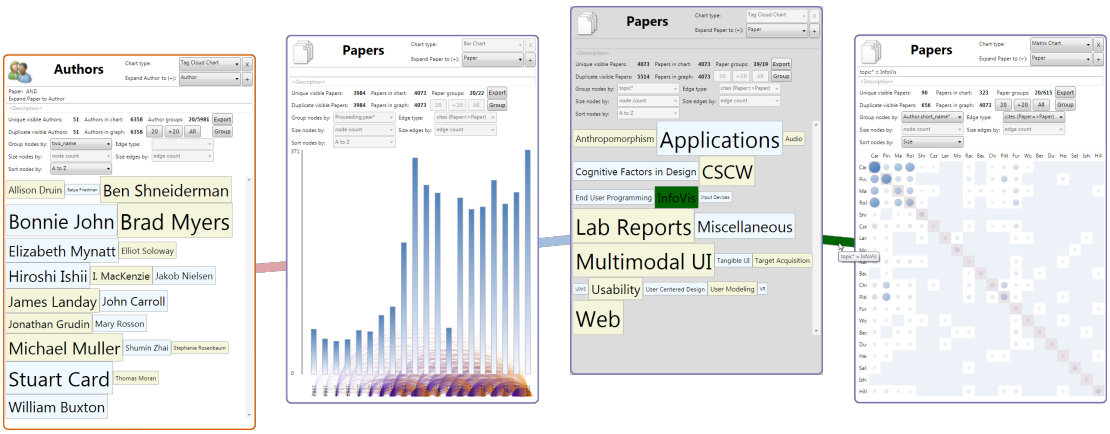


Figure 3.6: GraphTrail [Dun+12a] showing three views of ACM SIGCHI conference publications, based on both the authors and their connected papers.

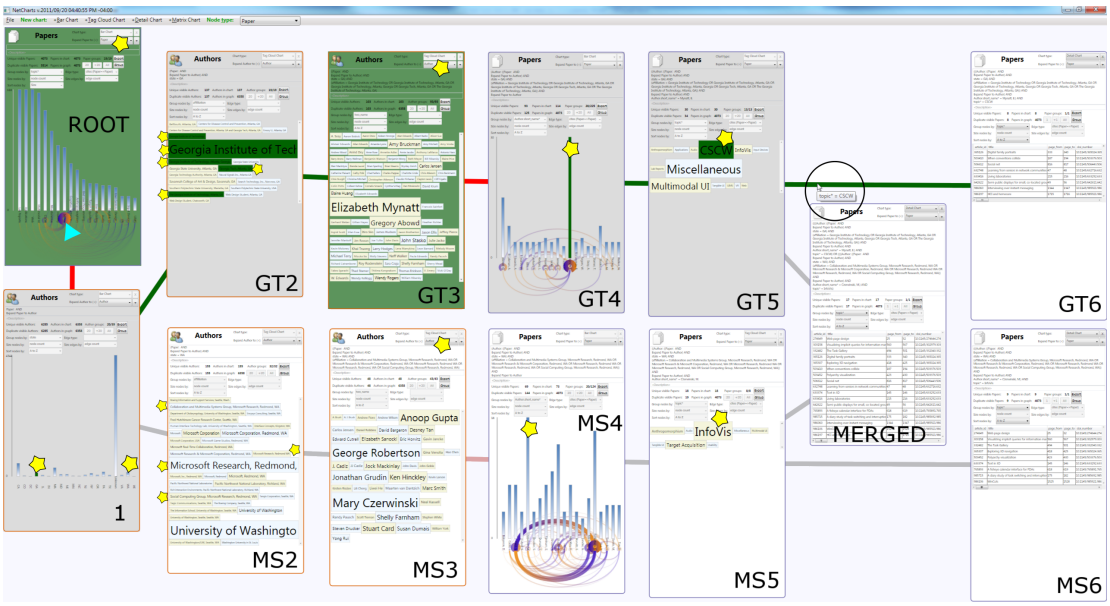


Figure 3.7: A GraphTrail [Dun+12a] analysis showing two parallel exploration paths, the top examining Georgia Tech (GT) publication and citation patterns and the bottom comparing Microsoft Research (MS). They start at the ROOT chart that contains all the papers in the dataset. Charts in each path are numbered in order of creation (e.g., 1, GT2, GT3, etc.), and the user interactions are shown with stars. The MERGED chart is the union of both branches' results. The user moved the mouse over the final parent link in the GT path (circled), highlighting the chain of actions up to the root.

ships between them. Similarly, the matrix chart on the far right show aggregate citations between authors and even to themselves along the diagonal. In addition, GraphTrail provides a pivoting mechanism to explore connected aggregates of the network across node types.

One of the main benefits of GraphTrail is an infinite canvas that aggregates can be dragged to and dropped to create new charts for filtered subsets. Moreover, data can be dragged from several charts into one target, creating the union of those sub-networks. This intuitive data filtering is augmented with parent links, which

indicate the source(s) of the data for each chart. On mouseover, the parent links highlight the entire provenance of that specific data all the way back to the root chart, in addition to a text tooltip indicating the operation performed. An example of this exploration history view is shown in Fig. 3.7. Exposing the analysis process in this way enables users to utilize their spatial memory while visual and textual feedback helps them track their interactions.

I compared GraphTrail with three tools with similar goals: NetLens [Kan+06] (Fig. 2.8), PaperLens [Lee+05], and FacetLens [Lee+09]. From this I determined that GraphTrail could make all the findings reported for the other tools, as well as several additional ones that were not discoverable in the others. Moreover, a three-month field study with a team of archeologists and a lab study demonstrated that GraphTrail improves insight discovery, analysis comprehension, exploration recall, and sharing analyses with others. Prior to using GraphTrail, the archaeologists had been using Cytoscape [Sha+03] to explore slices of the network with one or two node types, and GraphTrail greatly assisted their explorations by allowing more interactive exploration and exposing the exploration history. This approach may be a first step on the way to asynchronous collaboration for network analysis.

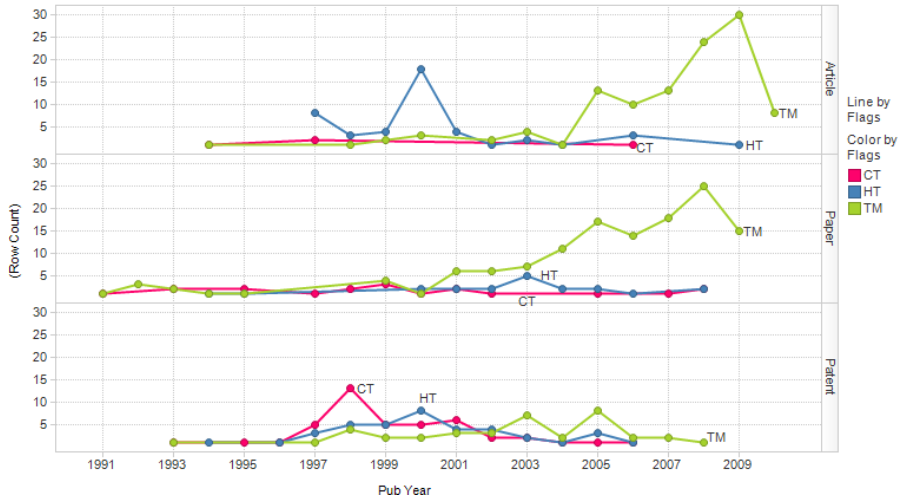
Both NetGrok and GraphTrail were designed to primarily display attribute information, with the underlying topology available on demand or in aggregate. These two approaches are highly effective for certain tasks, such as monitoring a

computer network (NetGrok) or exploring the attributes of a network while preserving the data provenance (GraphTrail). However, neither are particularly good at showing the overall topology and path information that would be available in a node-link visualization. Through these projects I began to understand the breadth of visualization techniques for networks, and that it is often difficult to build general tools for all kinds of analysis tasks. My dissertation work has primarily focused on helping users perform topology-based tasks, though my increased awareness of the importance of attribute values guided the design of the motif simplification glyphs (Chapter 4) and Group-in-a-Box aggregation techniques (Chapter 5).

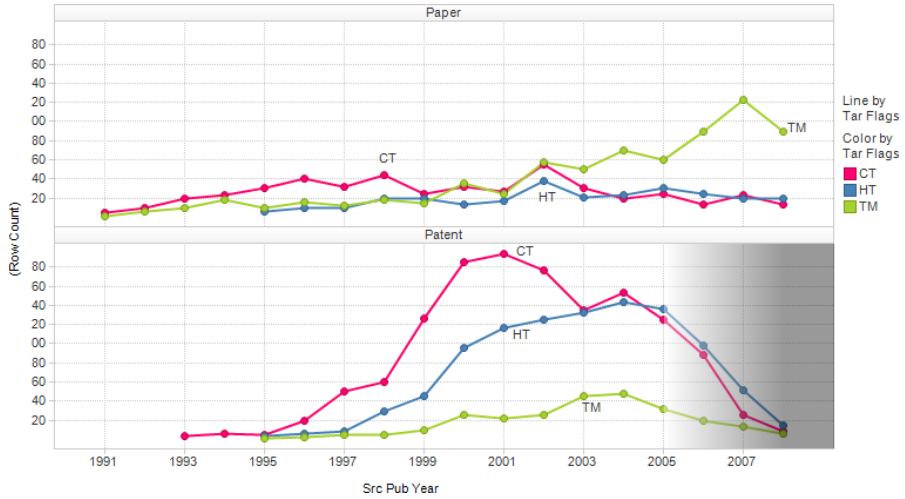
3.3.3 The Importance of Statistics and Algorithms

Much of my applied work in network analysis has been in text analytics and **scientometrics**, the science of measuring and analyzing science. My work on scientometrics focuses on measuring the impact of scientific publications, patents, and trade press articles and how they affect innovation.

One example is a study I did comparing the trajectory of three information visualization innovations: treemaps, cone trees, and hyperbolic trees [Shn+12]. I collected and analyzed academic publications, patents, and trade press articles over the almost two decades after the techniques were proposed. While node-link visualizations were useful, I found that for this task line charts were a more effective



(a)



(b)

Figure 3.8: These line charts show the impact of treemaps (TM/green), cone trees (CT/red), and hyperbolic trees (HT/blue) in terms of trade press articles, academic papers, and patents. (a) shows the number of publications per year by type of publication for each innovation and (b) shows the number of citations to papers and patents by year for each innovation. Note that the sharp fall in patent figures in the faded area may be due to the average 32-month USPTO processing time in 2005-2008. From [Shn+12].

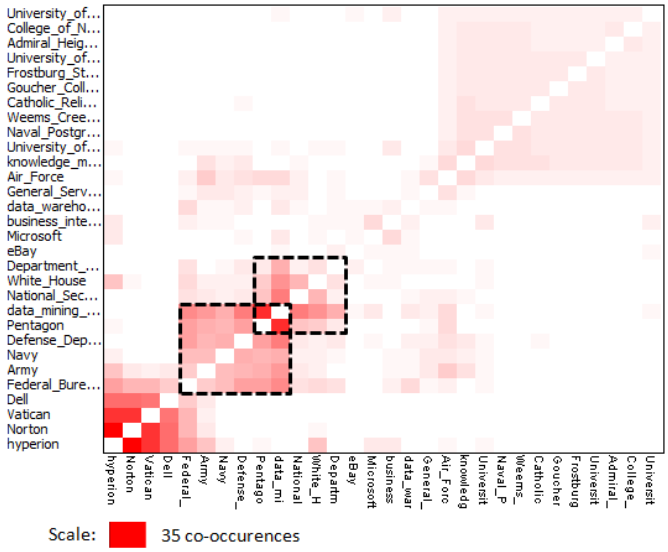


Figure 3.9: NetVisia [Gov+11b] visualization of the clustered heat map of the degree values for the STICK business intelligence term co-occurrence data from 2005, filtered to show only nodes with degrees between 45 and 491.

network representation of what we wanted to see: changes in statistics over time. Two examples are shown in Fig. 3.8, where the citation network is displayed as several line charts that show aggregates of nodes and edges over time. Our paper [Shn+12] shows additional examples using scatterplots.

I expanded these techniques to use clustered matrix diagrams for NetVisia [Gov+11b], including clustering nodes by metrics and by topology. An example of this is shown in Fig. 3.9 for business intelligence terms and their co-occurrences. In this case it was both statistics and hierarchical clusters of related terms that were of interest. These tasks were much more easily performed with line and matrix visualizations, and reinforced my belief that tasks and statistics of interest should guide tool and visualization design.

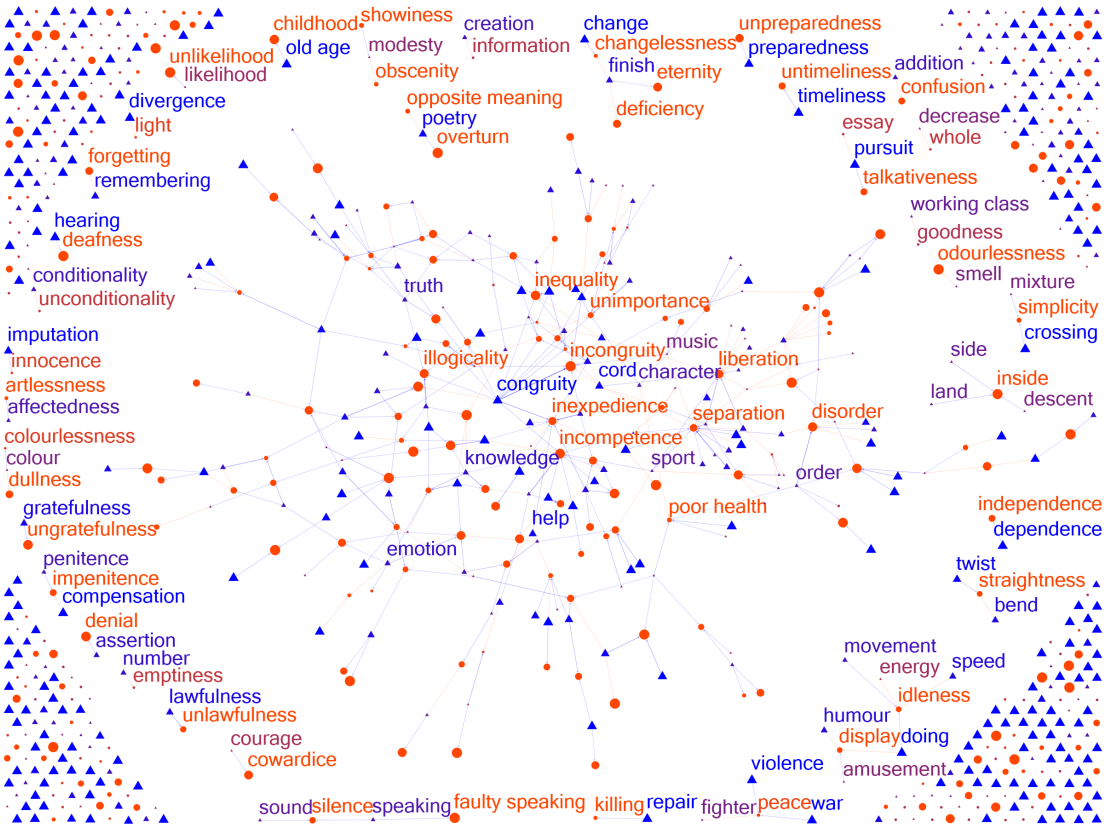


Figure 3.10: After removing edges with low weight we can see the structure the network backbone. Isolate category pairs are drawn in a ring around the main connected component and singletons are staggered in the corners. Each node is colored by its semantic orientation (red for negative, blue for positive) and edges are colored by their weight, from red to blue. Node shape also codes semantic orientation, with triangles positive and circles negative. Size codes the magnitude the semantic orientation, with the largest nodes representing the extremes. Node labels are shown for nodes in isolates and those in the top 20 for betweenness centrality. From [MDD09].

I also investigated using networks to model relationships between words or word categories. As a way to understand the behavior of a new sentiment analysis technique, I developed node-link visualizations of the semantic relationships between thesaurus categories [MDD09]. After algorithmically determining antonym relationships between categories of the Macquarie Thesaurus, I was able to show the relationships between categories of words as well as the semantic orientation of individual categories using color Fig. 3.10. The density of this network was quite high, with 812 nodes connected by 27,155 antonym edges, and thus necessitated substantial filtering and labeling only the most significant nodes. An interesting aspect of Fig. 3.10 is the large number of disconnected components in a ring around the center, representing small groups of related thesaurus categories. Moreover, there are many completely disconnected categories laid out in the corners of the visualization. At the time, NodeXL [Smi+10] had no way of handling these disconnected nodes and this layout took an enormous amount of my time to hand-tune. This kind of rote, manual correction helped me understand the necessity of techniques for handling disconnected components, such as the Group-in-a-Box layout algorithms I describe in Chapter 5 which would make this task automatic today.

Another project I was involved with focused on creating a literature exploration and analysis tool called Action Science Explorer (ASE) [Dun+12b; Gov+11a]. ASE was designed to support exploring a collection of papers so as to aid users

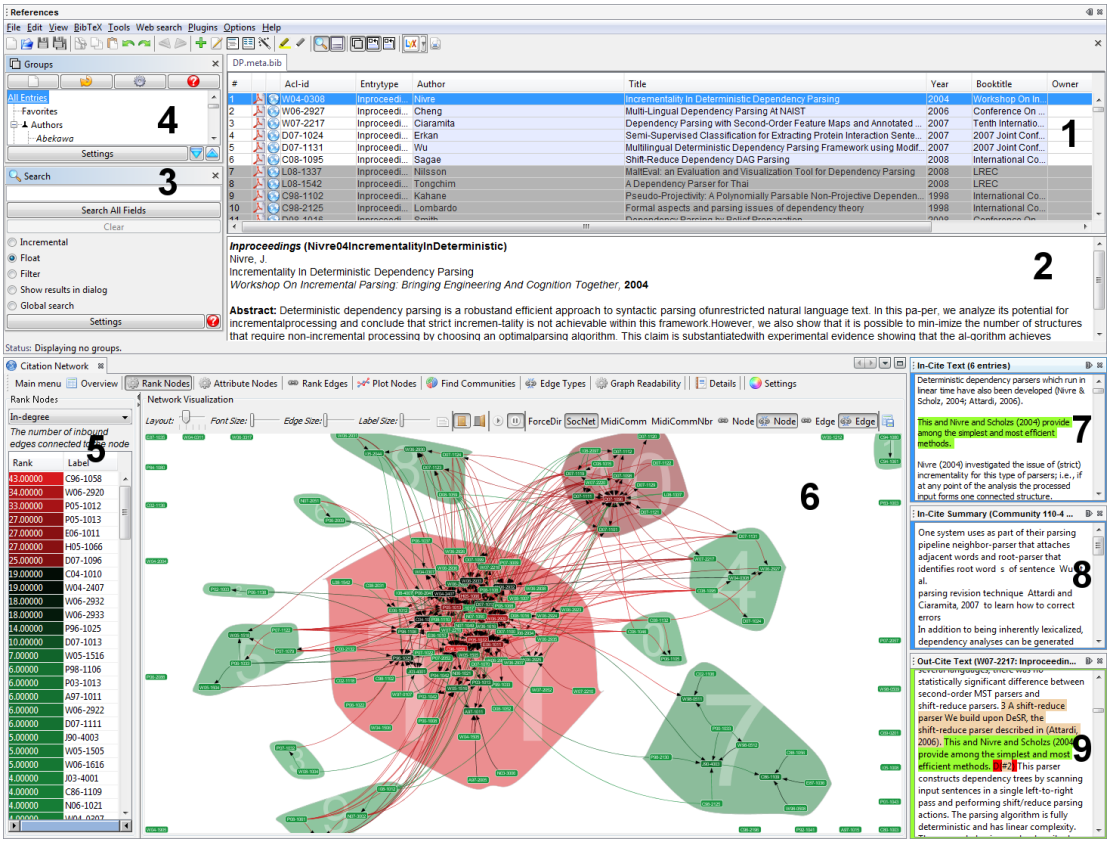


Figure 3.11: The main views of ASE [Dun+12b] are displayed and labeled here: Reference Management (1–4), Citation Network Statistics & Visualization (5–6), Citation Context (7), Multi-Document Summaries (8), and Full Text with hyperlinked citations.

in rapidly creating summaries of unfamiliar research domains. It incorporated (1) bibliometric lexical link mining to create a citation network for a field and context for each citation, (2) automatic summarization techniques to extract key points from papers, and (3) potent network analysis and visualization tools to aid in the exploration relationships. ASE, shown in Fig. 3.11, presents the academic literature for a field using many different modalities: tables of papers, full texts, text

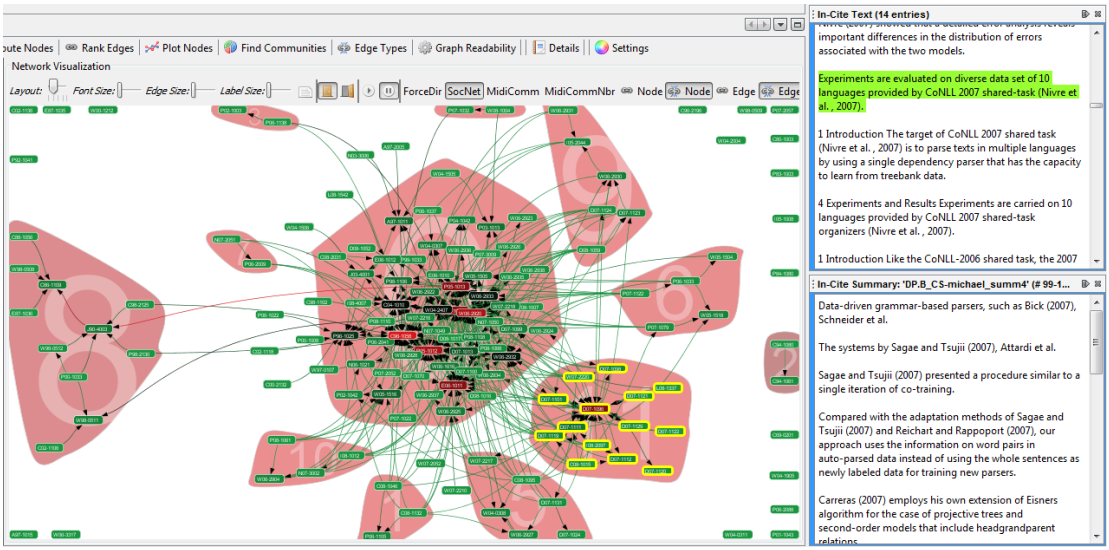


Figure 3.12: Algorithmically found communities in ASE [Dun+12b] are shown using convex hulls in the node-link visualization. When selected, all the citation context is shown in the top-right, along with an automatically generated summary of the overall context (bottom-right).

summaries, and visualizations of the citation network and the groups it contains. Each view of the underlying data is coordinated such that papers selected in one view are highlighted in the others, providing additional metadata, text summaries, and statistical measure rankings about them. Users can filter by rankings or via search queries, highlighting the matching results in all views.

ASE represented a major collaboration with several experts in Natural Language Processing, who were interested in (1) understanding the effectiveness of their link mining and multi-document summarization approaches and (2) being able to apply these algorithms to real tasks and present the results to users. An example of the multi-document summaries ASE can compute is shown in Fig. 3.12

for a selected topologic cluster of papers. Our collaborations helped them improve the effectiveness of the summarization algorithm, as well as develop a prototype tool that will guide developers of literature exploration systems to integrate such Natural Language Processing techniques.

From all these collaborations I have gained an improved understanding of how algorithms and statistics can be brought to bear on network analysis tasks. The various attribute- and topology-based clustering algorithms especially can be used to create the groups for my Group-in-a-Box layouts (Chapter 5). Moreover, if there are any text associated with nodes or edges like the Tweets in a Twitter keyword network, this text can be analyzed to present additional information to the user as part of the group box labels or in additional coordinated views. The results of a statistics algorithm can be shown using color coding or the like, and then displayed in aggregate within my motif glyphs (Chapter 4).

3.4 Summary

NodeXL [Smi+10] is a free and open source Excel template for network analysis. It provides powerful features while still being easy to learn, and avoids the pre-processing and programming steps required by many existing tools. The Excel integration brings standard formulas and macros, but we also include calculators for network statistics, layout algorithms, visual attribute encodings, dynamic filters,

direct manipulation, coordinated views, and much more. NodeXL is widely used in many disciplines and taught in over 25 introductory courses on network analysis. I have been involved in the design, evaluation, and development of NodeXL, and have integrated the techniques presented in this dissertation as part of the shipping product. As my research focuses on improving network visualization readability, it is especially beneficial for the introductory users NodeXL targets.

In addition to my work on NodeXL [Smi+10], I have been involved in the application of network analysis and visualization techniques to problems across several domains. In the various domains I have worked in, several different network properties have been important to display. Working with real users helps inform the design process, as the tasks, statistics, and algorithms relevant to them dramatically affect the choice of visualization and interaction techniques. In domains where network topology was most important to show, filtering the network by attributes or statistics was critical. The limitations of filtering techniques helped guide my development of motif simplification (Chapter 4). Moreover, when showing topology there is a major challenge in finding an effective and simple layout for the nodes that avoids readability problems, while at the same time highlighting the necessary structures for the task at hand. This provided me with motivation to investigate the use of readability metrics for understanding these issues and improving the layout (Chapter 6). In other cases, the attributes of the nodes or

edges in the network were more important, and I developed specialized visualizations depending on the user tasks. While my dissertation work is focused primarily on topology-based tasks, I gained an understanding of the importance of showing attribute or statistics information. This informed the design of my motif simplification glyphs and Group-in-a-Box aggregation techniques (Chapter 5). Other forays into domains such as Natural Language Processing helped me to understand the necessity of Group-in-a-Box layouts, even for handling simple disconnected components. These explorations helped shape the rest of my dissertation work, as well as my future design challenges.

Chapter 4

Motif Simplification to Reduce Complexity

4.1 Introduction

One way to reduce the complexity of node-link network visualizations is the use of aggregation, specifically by aggregating common network structures or subnetworks called **motifs**. Large, complex network visualizations often have large motifs repeated throughout because of either the network structure or how the data was collected. Regardless of their cause, some frequently occurring motifs contain little information compared to the space they occupy in the visualization. Existing tools may highlight certain motifs, allow users to filter them out, or replace groups of similar nodes with meta-nodes (e.g., see Section 5.2.2 and Fig. 5.4) – but each of these approaches has the serious limitation of obscuring the underlying topology.

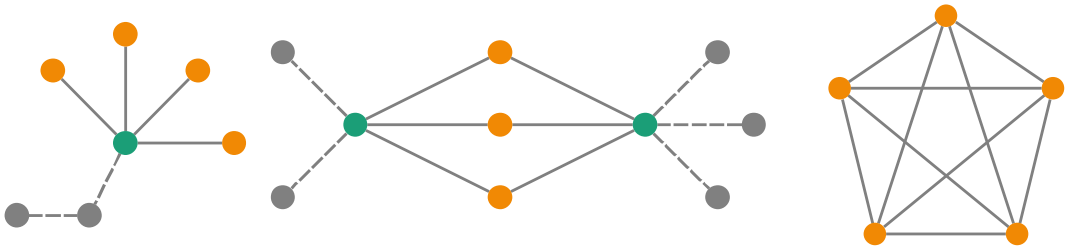


Figure 4.1: From left to right: fan, connector, and clique motifs.

I improve on these approaches with **motif simplification**, in which network motifs are automatically replaced with compact, representative glyphs. Well-designed glyphs have several benefits: they (1) require less screen space and layout effort, (2) are easier to understand in the context of the network, (3) can reveal otherwise hidden relationships, and (4) preserve as much underlying information as possible. In this chapter I discuss three high-payoff motifs that plague network analysts, shown in Fig. 4.1: **fans**, **connectors**, and **cliques**. I contribute the design of representative and combinable glyphs for these motifs, algorithms for detecting them, and a supporting task-based controlled study with 36 participants. These techniques are all implemented and made publicly available as part of the free and open source NodeXL network analysis tool [Smi+10].

4.1.1 Chapter Overview

Specifically, the contributions of this chapter are:

- A technique for simplifying node-link visualizations by replacing common network motifs with representative glyphs,
- A set of design guidelines for these glyphs to show the motif contents and underlying attributes,
- The design of glyphs for fans, connectors, and cliques,
- Algorithms for detecting these three motifs,

- A supporting task-based study with 36 participants,
- A free and open source implementation as part of NodeXL.

Parts of this chapter have been published [DS13] as well as featured in an overview paper on novel network analysis techniques in NodeXL [SD12]. I first describe the basics of Motif Simplification (Section 4.2), including glyph design (Section 4.2.1), motif detection algorithms (Section 4.2.2), and details about the NodeXL implementation (Section 4.2.3). I next demonstrate the utility of motif simplification in several case studies (Section 4.3), a usability study (Section 4.4), and a controlled experiment (Section 4.5). I end with a summary in Section 4.6.

4.2 Network Motif Simplification

Many common network motifs present little meaningful information, yet can dominate much of the display space and obscure interesting topology. I believe that replacing these motifs with representative glyphs will create more effective visualizations as there will be far fewer nodes and edges for layout algorithms and users to consider. I have chosen three motifs for my foray into motif simplification:

- A **fan motif** consists of a **head node** connected to **leaf nodes** with no other neighbors. As there may be hundreds of leaves, replacing all the leaves and their links to the head with a **fan glyph** can dramatically reduce the network size.

- A **D-connector motif** consists of functionally equivalent **span nodes** that solely link a set of D **anchor nodes**. Replacing span nodes and their links with a **connector glyph** can aid in connectivity comparisons.
- A **D-clique motif** consists of a set of D **member nodes** in which each pair is connected by at least one link. Cliques are common in biologic or similarity networks, where swapping for a **clique glyph** can highlight subgroup ties.

These motifs are prime simplification candidates for several reasons. For one, these motifs are quite common in the network datasets I have encountered in several disciplines. While simple to understand on their own, these motifs can account for much of the visual complexity of a node-link visualization. The fan motifs especially can dominate the diagram. While connector motifs usually occupy less space than the fans, they are hard to detect and can contribute substantial complexity. In the densest networks, such as similarity scores, overall relationships can be hidden in a tangled hairball of overlapping clique motifs as in Fig. 4.9a.

4.2.1 Glyph Design

For each motif, careful thought must be given to how to represent the simplified version. Arbitrary motifs can be shown as a simple meta-node (e.g., $\bigoplus$), possibly with embedded images that show a small node-link visualization of the underlying subnetwork. However, a specially designed representative glyph for a motif can

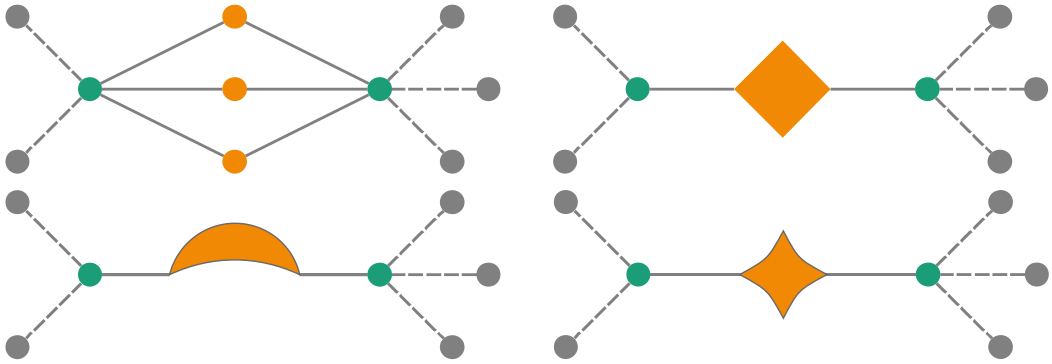


Figure 4.2: A 2-connector motif with three simplified glyph variants: diamond, crescent, and tapered diamond.

make it easier to understand aggregate topology and attributes with only minimal additional visual clutter. I went through several designs for each of my motif glyphs, some of which are discussed below.

4.2.1.1 Motif Topology

Foremost each glyph must be representative of the underlying subnetwork topology so that the aggregate relationships in the network can still be understood. As I aim to reduce visual clutter, I must use a small, easily-distinguishable glyph rather than heavy-weight visualizations. An effective way to differentiate the glyphs is to use unique shapes to identify each type, ideally that correspond to the underlying topology.

Several example shapes for a connector motif are shown in Fig. 4.2. The diamond is a straightforward representation of the outline made by the motif topology, is discernible at scale, and has geometric properties that allow easy area scaling

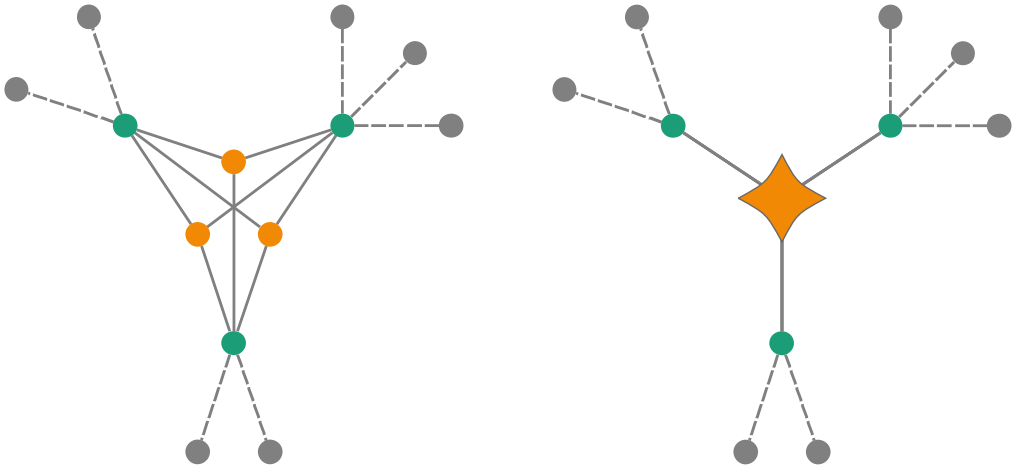


Figure 4.3: A 3-connector motif and its glyph.

and subdivision. However, they are often used with other shapes for categorical attribute coding. The crescent is not, but my user study indicated that its asymmetry was visually jarring and that it had poor edge connector properties (Section 4.4). I finally chose a symmetric tapered diamond: unique enough to be distinguishable and representative yet symmetric and connectable. I use the same shape regardless of the number of anchor nodes so as to reduce the shape corpus required (Fig. 4.3). The clique motifs were originally represented with a tapered square to indicate the link density, but it was easily confused with the connector motif and has since been replaced with a rounded X (Fig. 4.6). Like the connector motif, the same shape is used for any number of clique members. For the fan motifs, I chose a sector of a circle (Fig. 4.4), as it represented the fan of leaf nodes commonly seen in node-link visualizations.

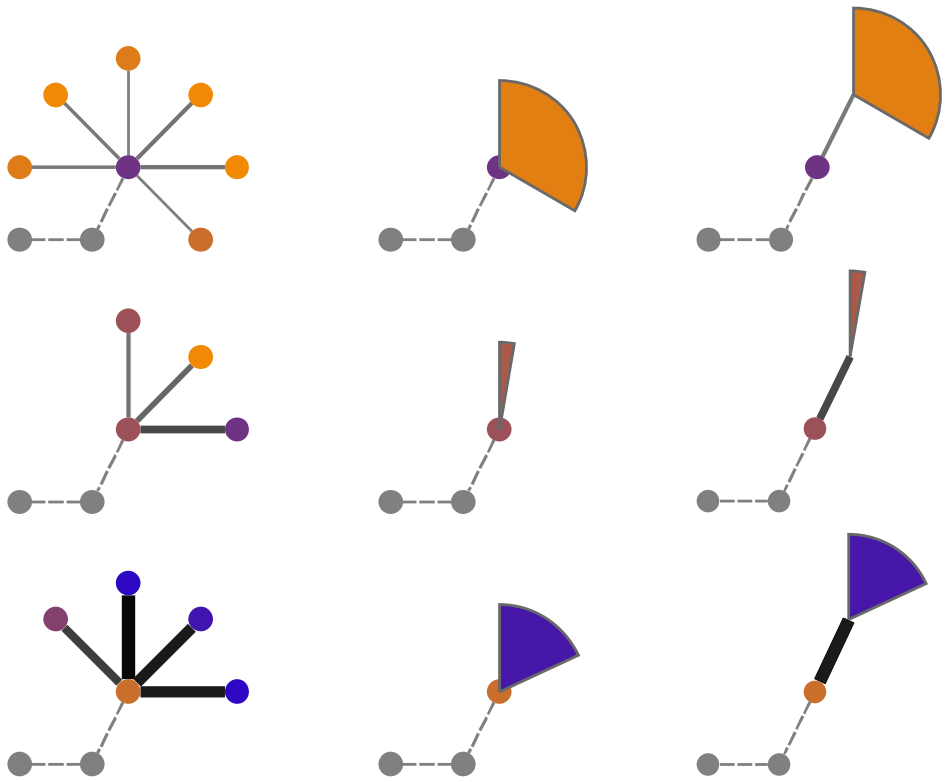


Figure 4.4: Three fan motifs and two glyph variants of each.

4.2.1.2 Contained Nodes

In addition to the topology, it is helpful to show information about the nodes contained in the motif. What information we want to show impacts the display mechanism we choose for it. Most useful would be a count of the nodes in the motif. This quantitative value is best expressed by position [Mac86], though in node-link visualizations this is reserved for showing ties. The next best choices would be length, angle, or area [Mac86]. For the fan motif, I scale the angle of the sector linearly between 10–120° by the number of contained nodes, which also linearly

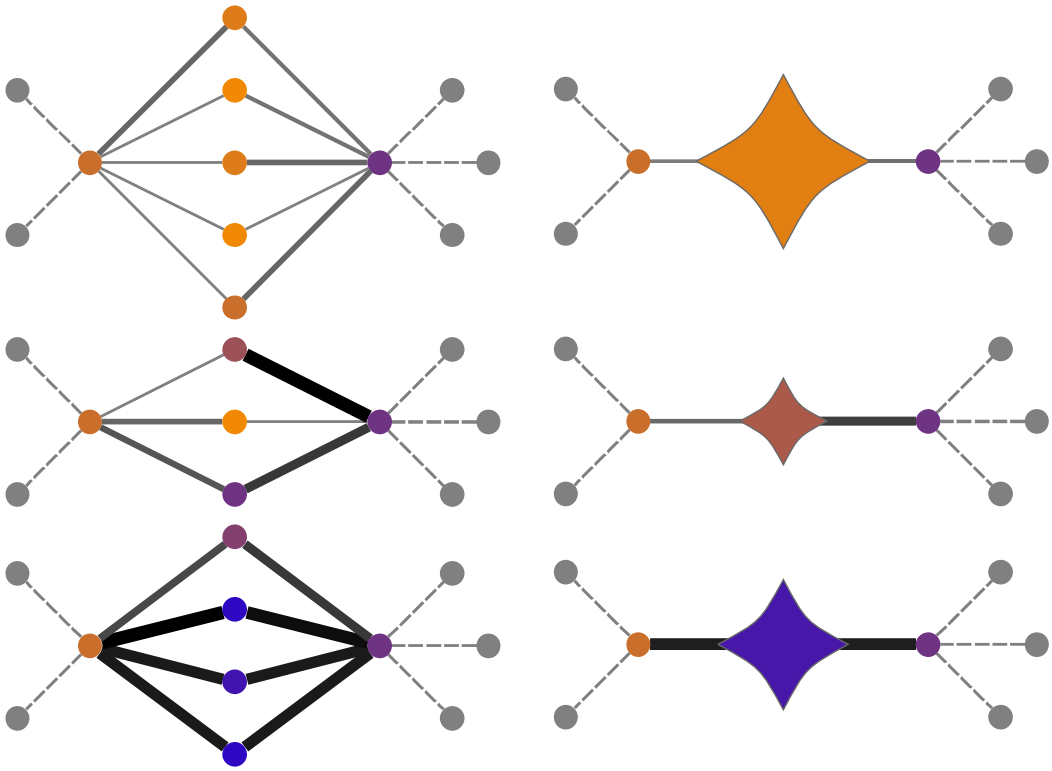


Figure 4.5: Three 2-connector motifs and their glyphs.

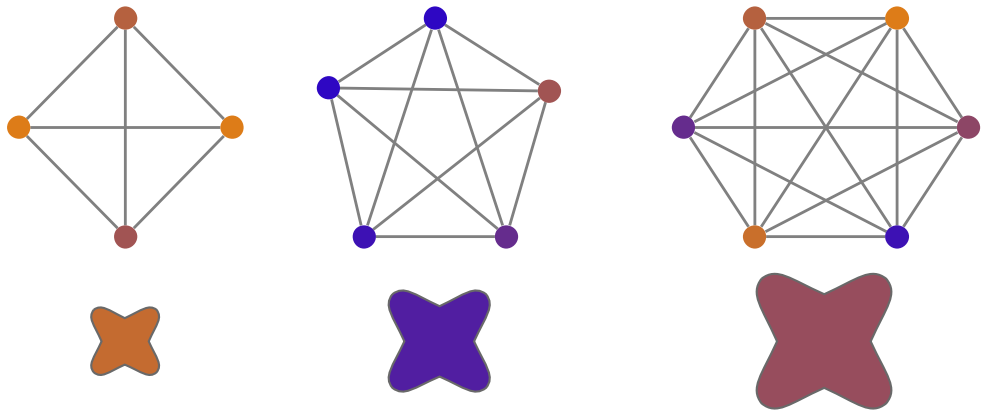


Figure 4.6: 4-, 5-, and 6-clique motifs and their glyphs.

scales its area (Fig. 4.4). I chose this range after tests using smaller ranges (20–90°) did not reveal enough size variation. The vertical alignment eases area comparisons and eases glyph subdivision to show edge directionality or attributes. I also scale the area of the other motifs linearly by the number of nodes (Figs. 4.5 and 4.6). Designers of future motif glyphs should ensure the shape is still discernible at its minimum size while not so large at its maximum to occlude edges unnecessarily.

We may also wish to show quantitative attributes or statistics of the underlying nodes. Showing all the values or their distribution would require complex embedded charts or focusable tooltips. Instead, I show a function of the values such as mean (used for these examples), sum, min, or variance. As size is reserved for node count, we are left with the less effective color saturation, color hue, and density/opacity [Mac86]. While these are less effective encodings, the maximum deviation reported for quantitative tasks is only 13% [CM85]. Glyphs demonstrating these quantitative attribute or statistic encodings are shown in Figs. 4.4 to 4.6, using the same color scale as the underlying nodes in the network. Categorical attributes are more challenging to display without subdividing glyphs or embedding visualizations, increasing the visual clutter. Finally, text attributes such as labels would help reveal the contents of the motif. While a glyph can show a small label, it is challenging to compute a representative one. Instead, I discuss later how interactivity can reveal the underlying nodes.

4.2.1.3 Connecting Edges

Nodes contained within a motif may have connecting edges, and when the motif is simplified these edges are re-routed to link to the glyph instead. This can result in duplicate, overlapping edges in straight-line drawings, as with the connector motif in Fig. 4.5. As with nodes, it is useful to show the number of duplicate edges and any attributes they may have. The edges could be drawn independently as curves of varying arcs, stacked in slices with scaled area, or use the edge distribution visualizations from [Mur08]; but again I strive to avoid visual clutter and show aggregate relationships clearly.

I aggregate these duplicate edges into meta-edges, with width and thus area representing a function of the underlying edges such as the number of edges (Figs. 4.9 to 4.11), the average of an attribute value (Figs. 4.4 and 4.5), etc. There are options for showing categorical attributes or labels, but these require cluttered embedded visualizations or interactivity. In some cases there are no attributes on the edges to encode, and showing even edge count would be a redundant. One example is the fan motif, in which the number of edges equals the already-encoded number of leaf nodes (in an undirected network without duplicates). Example fan glyphs without meta-edges are shown in the center column of Fig. 4.4.

Alas, glyph shape impacts how edges connect to them. Ideally, each glyph lies along a straight line with connecting edges so paths can be traced easily. For the

2-connector motif, a crescent would suffice if its corners were aligned along the path (Fig. 4.2). However, for connectors with three or more anchors my users reported that crescents make edges difficult to follow. Symmetric shapes like the tapered diamond and rounded X are better suited for many connecting edges.

4.2.1.4 Motif Overlap

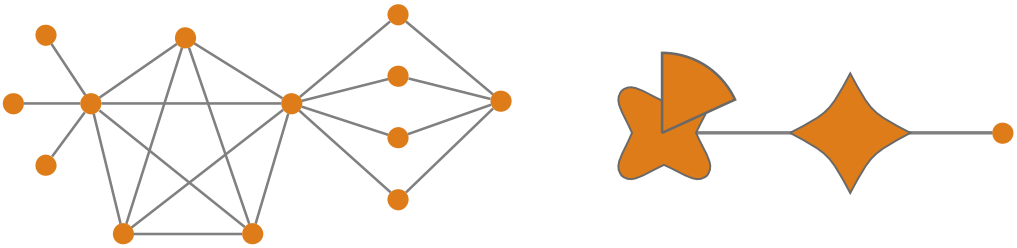


Figure 4.7: Glyphs for fan, clique, and connector motif overlap.

Often motifs are non-overlapping and easily transformed into glyphs, though many motifs do not have this luxury. When detecting motifs I can choose a non-overlapping set to display, but motif glyphs will be more effective at reducing complexity when they can be combined to show overlapping motifs. The design of any motif glyphs must thus take overlaps into account. Among my three motifs, fans are the most immune to overlap. The fan leaves have too few edges to participate in the other motifs, though the fan head can be a connector anchor or clique member. As a clique glyph replaces all the clique members, I must exclude the fan head from the fan glyph to allow this combination. Similarly, a connector anchor can be a clique member, which requires its exclusion from the connector

glyph. Two example overlaps are shown in Fig. 4.7 and more on overlap handling is discussed later in Section 4.2.2.4.

4.2.1.5 Glyph Interactivity

While the motif glyphs I described can be effective for simplifying a network, I would like to make sure that they are easily understandable and investigable. One important aspect of this is to ensure that users can switch between the original and simplified views interactively. Users can simplify the entire network, or only a selected subset of motifs. Likewise, users can expand the entire network to see the original visualization, or only expand a selected glyph they are interested in exploring. I expose the contents of each glyph with tooltips. It would be possible to expand on this and show details for a glyph via a heavyweight focusable tooltip that contains a chart of attribute distributions or a list of node labels.

Direct manipulation of the motif glyphs and underlying nodes is an effective way of exploring the network. Users can adjust node or glyph placement manually, as well as highlight incident edges or adjacent nodes through simple context menus. Additionally, automatic layout algorithms are available for laying out the simplified network. An ideal layout algorithm would take the shape and size of the glyphs into account, in addition to the number of edges in any meta-edges.

4.2.2 Motif Detection Algorithms

General motif detection can be accomplished with approaches like symmetry-breaking [GK07], but custom algorithms are more effective for specific motifs that can vary substantially in size. I have implemented algorithms to detect fan, connector, and clique motifs of all sizes. I refer the interested reader to view and utilize my C# source code.[1] I use the terminology of a network or graph G with a set of nodes $G.nodes$, and each node n has a set of adjacent nodes $n.neighbors$. The size of each of these node sets, say s, is denoted as $|s|$.

4.2.2.1 Fan Motifs

My approach to detecting all the fan motifs in a network is detailed in Algorithm 1, which has a run time complexity of $O(|G.nodes| \times$ average neighbor count). Average neighbor count is usually relatively small and can be considered a bounded constant, so this technique should scale well. However, I recently came upon an alternate, faster algorithm with linear time complexity shown in Algorithm 2, though it has not yet been implemented in NodeXL and is not discussed further here.

The current algorithm (Algorithm 1) first passes through all the nodes in the network, searching for potential fan heads. Each fan head must have two or more neighbors to exclude the degenerate barbell case (Line 3), though this criteria could be increased to find larger fans. For each potential fan head, I then search through

[1] nodexl.codeplex.com/SourceControl/changeset/view/70521#1208172

Algorithm 1 Fan motif detection algorithm.
Time complexity: $O(|\text{G.nodes}| \times \text{average neighbor count})$

```
1: procedure DetectFans
2:     for all n ∈ G.nodes do
3:         if |n.neighbors| ≥ 2 then
4:             leaves ← {∅}
5:             for all nbr ∈ n.neighbors do
6:                 if |nbr.neighbors| = 1 then
7:                     leaves.add(nbr)
8:             if |leaves| ≥ 2 then
9:                 RecordFan(n, leaves)
10: end procedure

11: procedure RecordFan(head, leaves)
12:     ...                                   ▷ Record a given fan motif
13: end procedure
```

Algorithm 2 Alternate fan motif detection algorithm.
Time complexity: $O(|\text{G.nodes}|)$

```
1: procedure DetectFans
2:     fans ← Map⟨Node, List⟨Node⟩⟩
3:     for all n ∈ G.nodes do
4:         if |n.neighbors| = 1 then
5:             head ← n.neighbors[0]
6:             if head ∉ fans then
7:                 fans[head] ← List⟨Node⟩
8:             fans[head].add(n)
9:     for all head, leaves ∈ fans do
10:        if |leaves| ≥ 2 then
11:            RecordFan(head, leaves)
12: end procedure

13: procedure RecordFan(head, leaves)
14:     ...                                   ▷ Record a given fan motif
15: end procedure
```

the set of its neighbors to find any leaf nodes connected only to it (Line 5). Each of these leaf nodes are added to the set of potential leaves. If two or more leaves are found in the neighbor set, the fan motif is acceptable and recorded (Line 8).

The differing neighbor count criteria for head and leaf nodes in Algorithm 1 prohibits any overlapping motifs from being detected. However, please note that I am using $|n.neighbors|$ to show the size of the neighbor set of n, which may differ from n's degree if there are overlapping edges. For example, in a network with directed edges a leaf node may have two overlapping edges connecting it to the head node, one for each direction. Moreover, an undirected network with several edge types may have overlapping edges of differing types. Some algorithms for computing degree would return higher values in these cases than the actual number of neighboring nodes.

4.2.2.2 Connector Motifs

Connectors have an **dimension**, denoted D, that indicates the number of anchors it has. D can be any integer two or greater, though the frequency of the motifs generally decreases proportional to D. My algorithm for detecting connector motifs of all dimensions is shown in Algorithms 3 and 4, and takes parameters *D-min* and *D-max* to indicate the range of dimensions to search for. The run time complexity of this algorithm is also $O(|G.nodes| \times$ average neighbor count). Again, average neighbor count can be considered a bounded constant.

Algorithm 3 Part 1/2 of the D-Connector motif detection algorithm which finds potential motifs and filters out invalid ones. [D-min, D-max] is the range of dimensions of the connector motifs to find (the number of anchors). Time complexity: $O(|\text{G.nodes}| \times \text{average neighbor count})$. See also Algorithm 4.

```
1: procedure DetectConnectors(D-min, D-max)
2:     found ← Map⟨String, Connector⟩
3:     detectLoop:
4:     for all n ∈ G.nodes do
5:         if |n.neighbors| ∈ [D-min, D-max] then
6:             for all nbr ∈ n.neighbors do
7:                 if |nbr.neighbors| < 2 then
8:                     continue detectLoop
9:             AddSpan(n.neighbors.sorted, n, found)
10:    out ← {∅}
11:    used ← Map⟨Node, Connector⟩
12:    filterLoop:
13:    for all c ∈ found.values do
14:        if |c.spanners| ≥ 2 then
15:            for all s ∈ c.spanners do
16:                if s ∈ used.keys then
17:                    c′ ← used[s]
18:                    cTotal ← |c.spanners| + |c.anchors|
19:                    c′Total ← |c′.spanners| + |c′.anchors|
20:                    if |c.spanners| > |c′.spanners| or
                         (|c.spanners| = |c′.spanners| and
                         cTotal ≥ c′total) then
21:                        out.remove(c′)
22:                        used.removeAll(c′.spanners)
23:                        used.removeAll(c′.anchors)
24:                        AddConnector(out, used, c)
25:                    continue filterLoop
26:            AddConnector(out, used, c)
27:    for all c ∈ out do
28:        RecordConnector(c.anchors, c.spanners)
29: end procedure
```

Algorithm 4 Part 2/2 of the D-Connector motif detection algorithm. This part contains procedures and a class needed for Algorithm 3.

```
30: procedure AddSpan(anchors, spanner, found)
31:     key ← string(anchors)
32:     if key ∉ found then
33:         found[key] ← new Connector(anchors)
34:     found[key].spanners.add(spanner)
35: end procedure

36: class Connector
37:     anchors ← {∅}, spanners ← {∅}
38:     procedure Connector(new-anchors)
39:         anchors ← new-anchors
40:     end procedure
41: end class

42: procedure AddConnector(out, used, c)
43:     out.add(c)
44:     for all spanner ∈ c.spanners do
45:         used[spanner] ← c
46:     for all anchor ∈ c.anchors do
47:         used[anchor] ← c
48: end procedure

49: procedure RecordConnector(anchors, spanners)
50:     ···                                  ▷ Record a given connector motif
51: end procedure
```

Connector motifs are not as straightforward to detect as fan motifs, despite the algorithms having the same run time complexity. First, a pass is made through all nodes searching for span nodes with sets of neighbors that could be anchors and creating or adding to a map of keys to possible motifs. An additional pass is required to traverse the potential motifs and remove those with only one span node, as well as remove all but the most desirable of any overlapping motifs. I

choose motifs to keep first by the number of spanners, then by the total number of anchors and spanners, then arbitrarily.

The algorithm is broken into several procedures and a class to store the details for each potential connector motif. The detect loop in the algorithm (Algorithm 3, Line 3) passes through all nodes in the network, searching for potential span nodes. Each span node must have between *D-min* and *D-max* neighbors, which must be anchor nodes. I require a minimum of two span nodes for the connector motif, so each anchor node must have two or more neighbors itself (Line 7). At least two of the neighbors are span nodes, but the remainder can be connections to the main network or other anchor nodes in the motif. If all the anchor nodes check out, the span node is added to a connector motif (Algorithm 3, Line 9) using the AddSpan procedure (Algorithm 4, Line 30). This motif can be new or an existing one with the same set of anchors. All existing motifs are stored in a map (Algorithm 3, Line 2), using a string representation of the anchors as a key and an instance of the **Connector** class (Algorithm 4, Line 36) as the associated value. This allows speedy lookup of each potential motif given a sorted anchor set. Note that the anchor set and its string representation must be sorted so as to avoid having motifs with identical anchor sets but the anchors were found in a different order.

After searching for all potential span nodes, Algorithm 3 requires an additional pass over the detected connector motifs to ensure that (1) they have two or more

span nodes and (2) they do not overlap with other connector motifs. The filter loop on Line 12 goes through each potential **Connector** instance in the map to verify that they pass these two criteria. The first criteria, the minimum number of span nodes, could be increased if only larger higher payoff motifs are of interest (Line 7, 14). An example I have found that matches the second criteria, connector motif overlap, is a ring of four nodes $A - B - C - D - A$ isolated from the rest of the network. In this case it is unclear whether to choose A & C or B & D as the 2-connector motif anchors, as I do not allow overlap.

As there may be other examples of overlap that need to be caught, I chose a general overlap detection approach that compares each span node s in a motif to all span and anchor nodes in already detected motifs (Algorithm 3, Lines 15 – 26). If there is no overlap with existing motifs, the potential **Connector** c is stored (Line 26) using the AddConnector procedure (Algorithm 4, Line 42). However, if one of the span nodes s of a potential **Connector** c is also a span or anchor node of an already found motif c', I then compare their sizes. I choose to keep the motif that has the greatest number of spanners, and if they are equal I choose the one with more total anchors and spanners. If both values are equal I keep the first detected. If the prior motif c' is to be replaced, I must first remove its spanners and anchors from the map (Algorithm 3, Lines 21–23). After passing the minimum span count and overlap ranking checks, the detected

connector motif c is then stored (Algorithm 3, Line 24) using the AddConnector procedure (Algorithm 4, Line 42). As part of this, the spanners and anchors are all added to the map of used nodes and associated with their **Connector**. All this bookkeeping process prevents a potential connector motif from overlapping with more than one that was already found. Finally, I record the remaining non-overlapping and valid connector motifs (Algorithm 3, Line 27).

4.2.2.3 Clique Motifs

To find all cliques in the graph I use the Tomita et al. algorithm [TTT06], which has a run time complexity of $O(3^{|G.nodes|/3})$. However, this algorithm has high memory requirements and for especially large graphs a new linear-storage algorithm by Eppstein and Strash may be faster or required [ES11]. Unfortunately cliques in general can have high amounts of overlap. I use a greedy heuristic that chooses the largest non-overlapping clique motifs to keep that has a time complexity of $O(\text{number of motifs} \times \text{average motif size})$. This works well on the networks I have analyzed, but may be insufficient for studying dense networks.

4.2.2.4 Resolving Motif Overlap

When computing motifs, not only can motifs of a type overlap (e.g., cliques), but in general the various types can overlap with each other as well. While my design for fan and connector motifs prevents ambiguous overlap and allows easy combinations

(Fig. 4.7), the choice of which cliques to simplify can impact user perception of the network. To effectively pick a disjoint set of motifs to keep I would have to rate each motif by desirability and solve the set packing problem, one of Karp's 21 NP-complete problems [Kar72]. Not only is this problem computationally hard to solve exactly, it is also difficult to approximate, hence my use of heuristics.

4.2.3 NodeXL Implementation

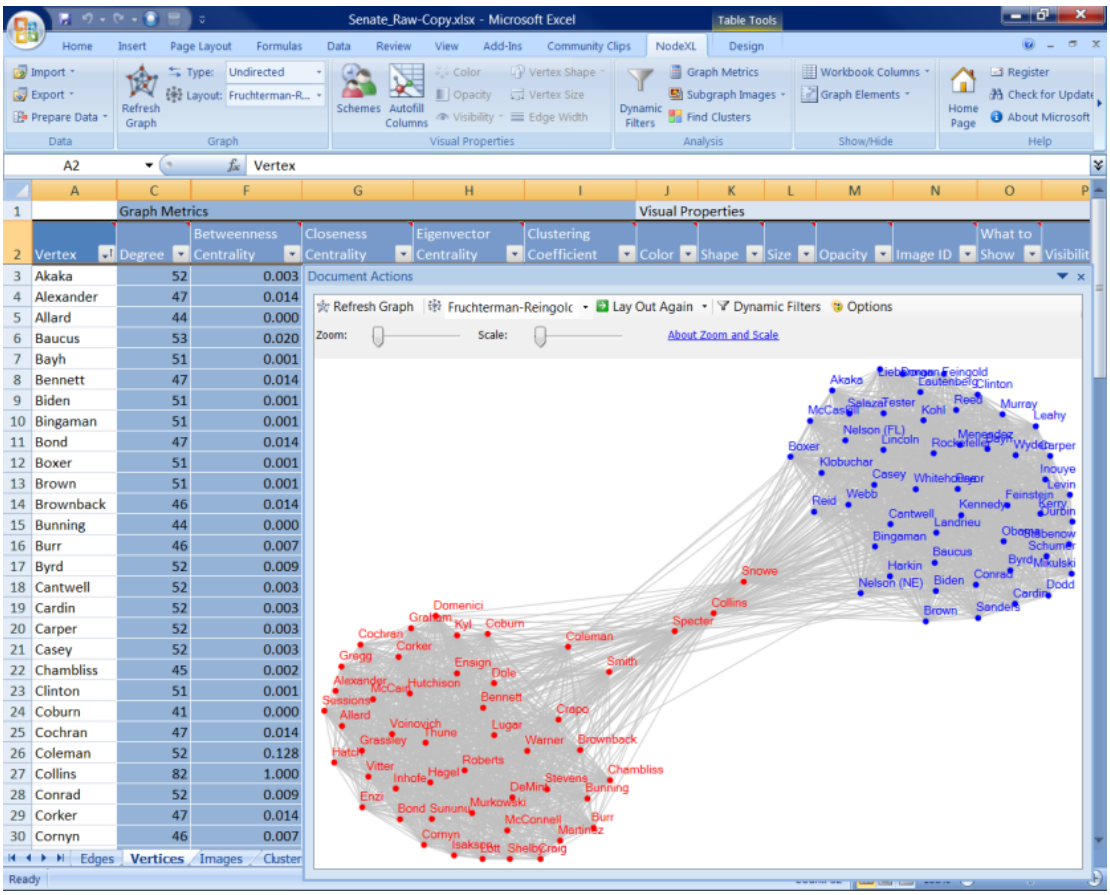


Figure 4.8: The standard NodeXL workspace, showing U.S. Senate voting patterns from 2007. The left view shows the worksheets that store the network and its attributes, while the right pane shows a node-link visualization of the network.

I have implemented a reference implementation of my motif simplification approach and made it publicly available as part of the NodeXL network analysis tool [Smi+10; Smi+09]. Given that many NodeXL users generally have little prior knowledge about network visualization readability, I believe that they will particularly benefit from my interactive motif simplification techniques.

I have integrated my motif simplifications into the standard NodeXL groups infrastructure, which stores groups using two worksheets: (1) **Groups** which contains a row for each group and its attributes, and (2) **Group Vertices** where each row maps an individual grouped node to its associated group. These worksheets can be populated automatically in a variety of manners, including detection of topological clusters, exact-value attribute groupings, connected components, and now my three network motifs. The NodeXL group model allows for nodes that are in no group at all, which is important for motif simplification as not every node in the network is part of a motif. Note however that this group model does not allow overlapping groups, which means that special care must be given to the definition of what members of each motif constitute the group in the worksheets.

In the group worksheets users can interactively edit the labels, attributes, visual encoding, and membership of specific groups; remove groups completely; or even create custom sets of groups by editing the worksheets or visual interaction with the node-link visualization. Moreover, automated statistics can be computed for

each group and added to the Groups worksheet, including node & edge counts, geodesic distances, and graph density; as well as the number of edges between pairs of groups in a special **Group Edges** worksheet.

After the groups have been computed or entered into the worksheets manually, users can display them in the visualization pane. When users select a group in the worksheet, all its member nodes are selected in the visualization. Likewise, for any nodes selected in the visualization users can select any groups in the worksheet that contain them using the ribbon menu. By default, groups are shown in their original expanded form based on the current layout algorithm, with categorical color and shape coding so as to distinguish them from each other. However, users can switch between the original expanded form and an alternate collapsed form for specific selected groups or all groups. This is done using the context menu in the visualization pane or the ribbon groups menu.

The default collapsed form for groups is a meta-node representation of the same categorically coded shape with a plus sign inside to indicate its status (e.g., $\bigoplus$), sized proportional to the number of nodes the group contains and with any associated label next to it. However, the groups for my motifs use their representative glyphs that were described in Section 4.2.1. When a collapsed group is selected in the visualization pane it is also selected in the Groups worksheet, and its position in the visualization can be adjusted with the mouse. These collapsed representa-

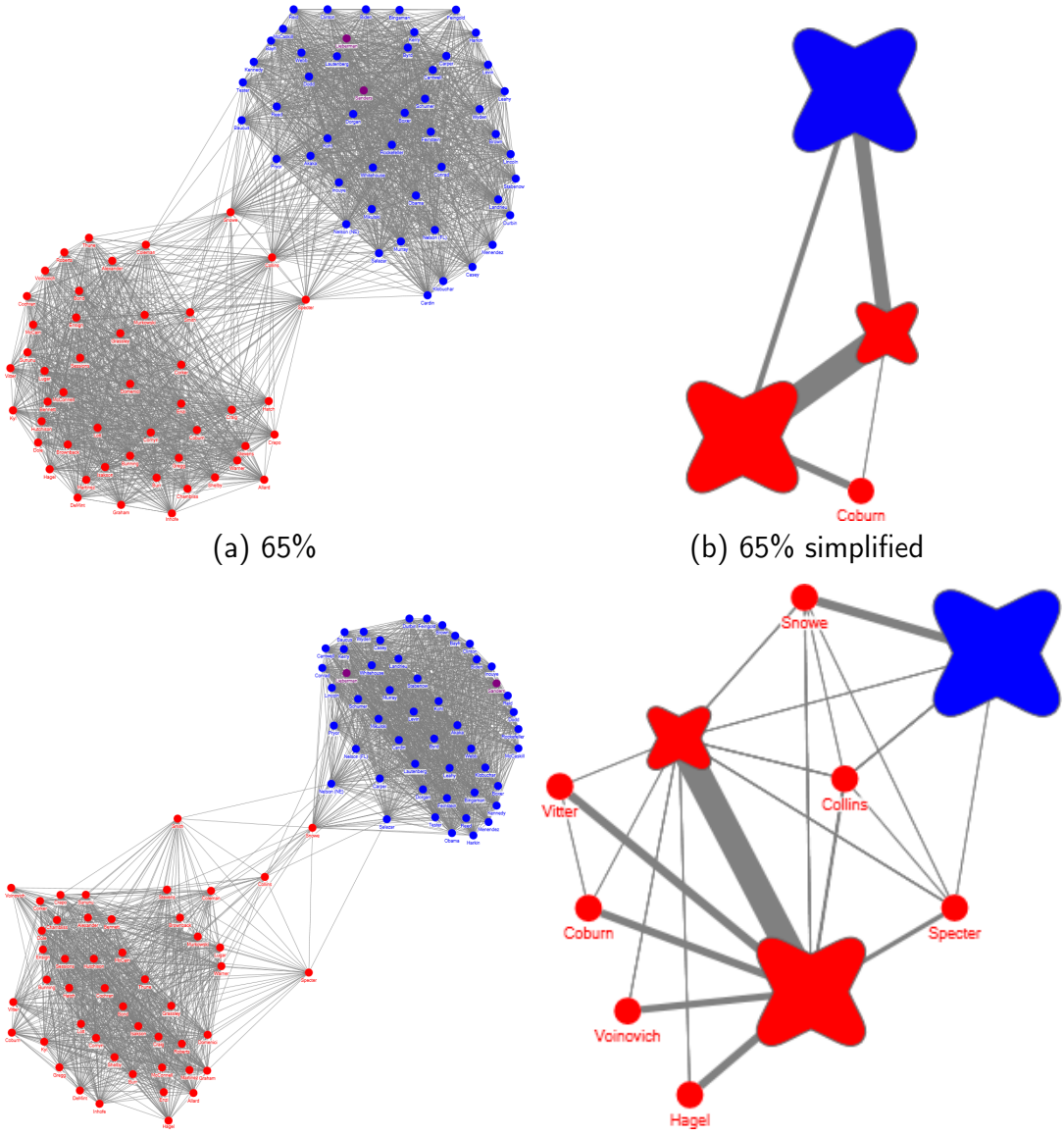
tions are by default colored using the same categorical coloring as for the expanded version so the association between views can be easily identified. Through an option in the groups menu, users can switch from the default categorical colors and shapes to the underlying node attribute encodings the user specified. This updates all collapsed motifs so that they show the aggregate attribute information about the underlying nodes they represent.

4.3 Case Studies

I explored several networks of interest using motif simplification, in several cases while helping domain experts analyze their data. Overall, motif simplification resulted in vastly reduced network size, reducing the visual complexity faced by the user and easing automatic and manual layout tasks.

4.3.1 U.S. Senate Voting Patterns in 2007

The power of clique motif simplification is shown in an example network of U.S. Senate voting patterns from 2007, originally discussed in Section 3.3.1. Fig. 4.9a, like Fig. 4.8, highlights the bridge-building nature of three Republican senators in the middle of the visualization. However, further insights are not readily visible in the tangled hairball of each party except, perhaps, that the two independent senators vote with the Democrats.



(a) 65% (b) 65% simplified

(c) 70% (d) 70% simplified

Figure 4.9: U.S. Senate 2007 co-voting network at 65% and 70% agreement cutoffs, simplified using clique motif glyphs. Key features are visible, such as the moderate Republican clique around McCain with "wildcards" at the periphery.

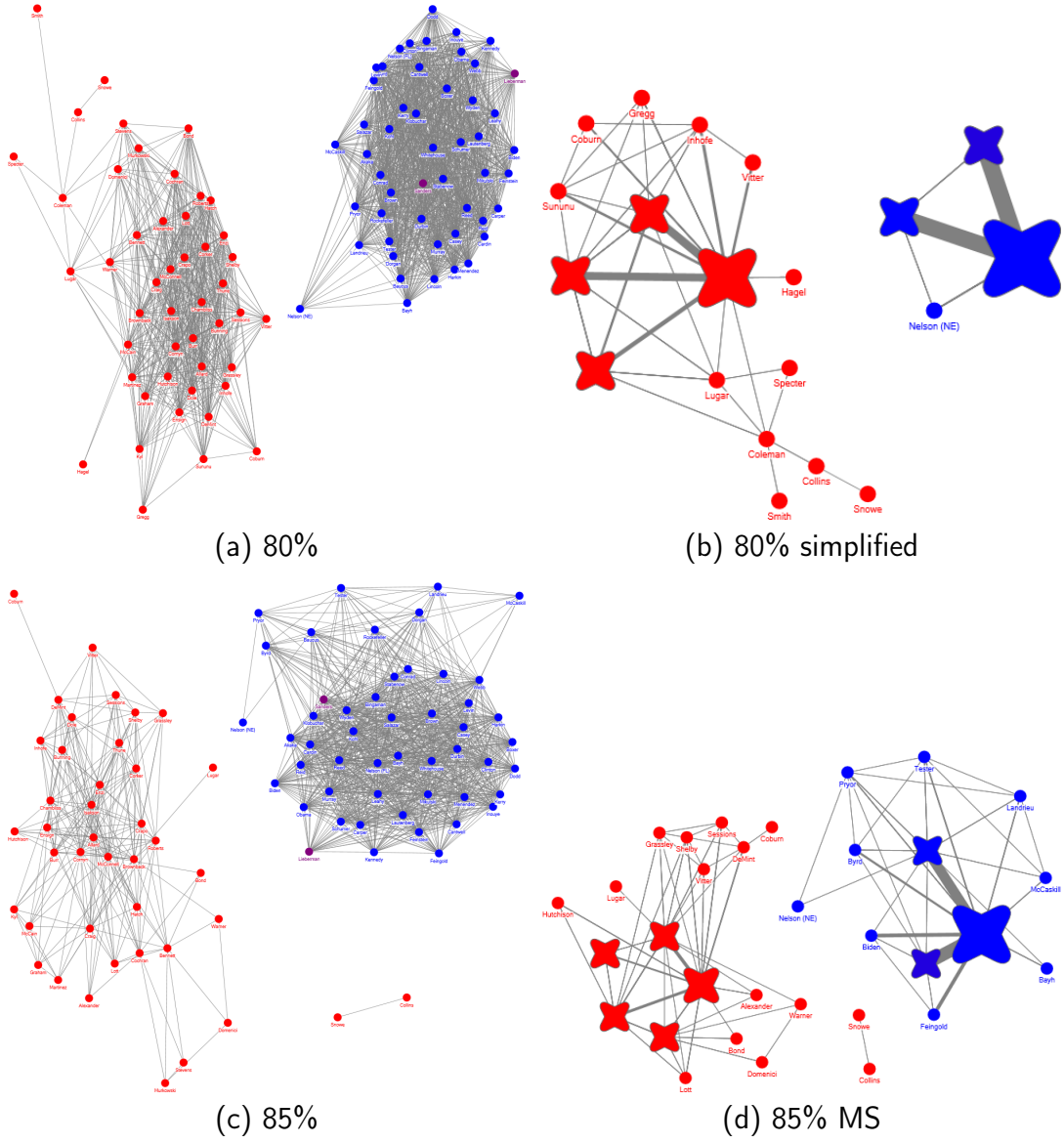


Figure 4.10: U.S. Senate 2007 co-voting network at 80% and 85% agreement cutoffs, simplified using clique motif glyphs. The east-coast liberals and the Blue Dog Democrats separate at 80%. We see the network decompose at higher cutoffs.

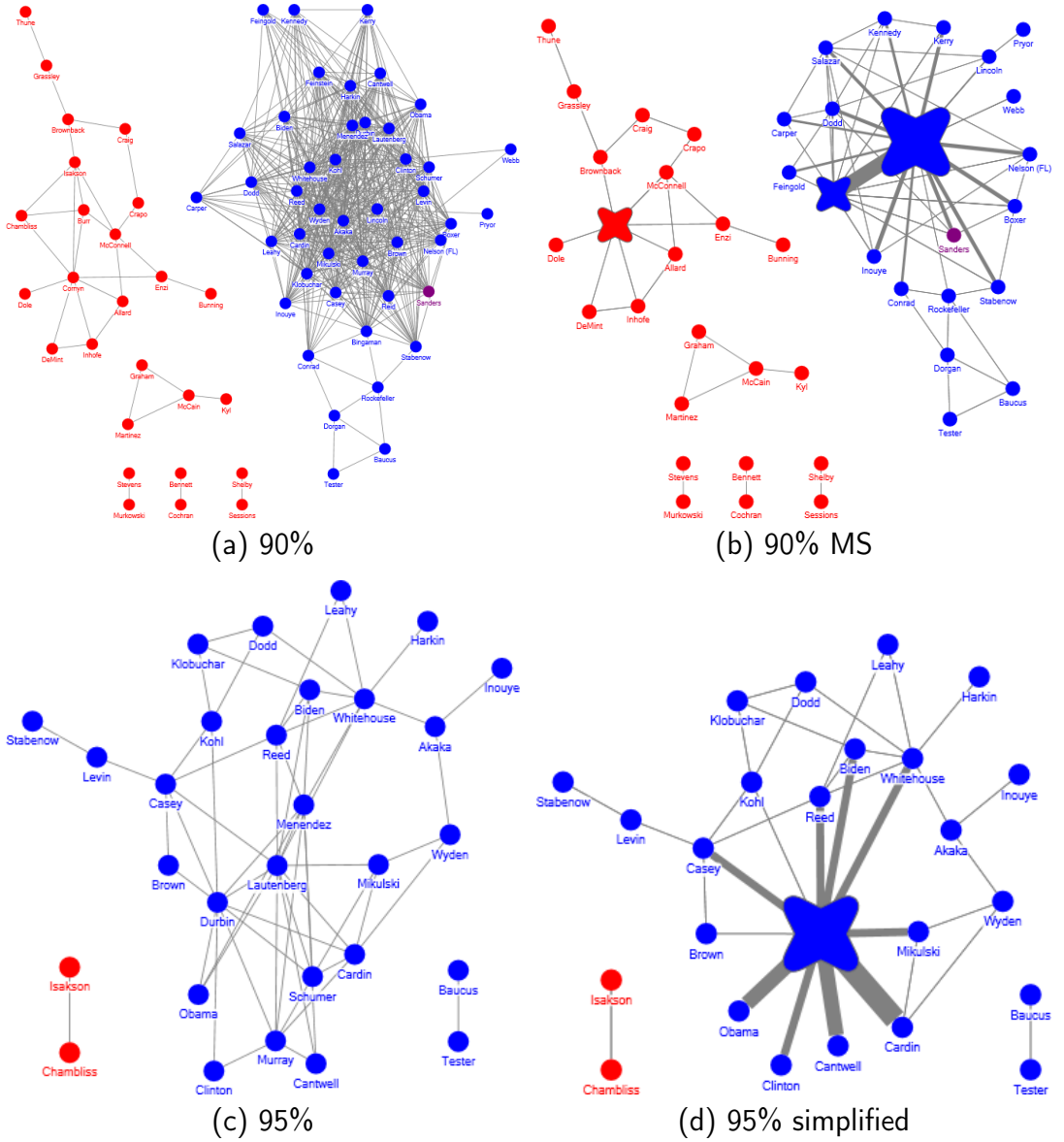


Figure 4.11: U.S. Senate 2007 co-voting network at 90% and 95% agreement cut-offs, simplified using clique motif glyphs. We see the Republican party fragment, with only the two senators from Georgia remaining at 95% agreement.

After simplifying cliques, several additional features are visible (Fig. 4.9b). There are three completely connected groups: one with 48 Democrats, the two independents, and a Republican (Snowe); another with 42 Republicans; and a 4-clique of Collins, Smith, McCain, and Specter. I worked with a political scientist studying at the University of Wyoming to see if these cliques highlighted known behavior, and, in fact, they did. The 4-clique represents moderate Republican bridge builders that were often decisive votes, though they have stronger ties to the Republican clique. The only Senator not in a clique is Coburn, a staunch Republican on contentious issues but who often votes his heart.

I increased the cutoff to 0.70 and ran the layout again (Fig. 4.9c). However, the simplified version (Fig. 4.9d) has become quite intriguing. While the Democrats and Independents still form a 50-clique, a few members trickled out of Republican cliques. Snowe returns to the middle with high connectivity with her former Democrat clique. Collins and Specter also move to the center, replaced in the McCain clique by Coleman and Lugar – more moderates. The corner outliers are known wildcards that do not follow the party.

Extending this process to higher cutoffs, we begin to see party fragmentation, led by the Republicans (Fig. 4.10). At 0.80 the network bisects (Fig. 4.10a), and the Democrats split into three cliques and a solitary Nelson, a Blue Dog moderate (Fig. 4.10b). The top right 4-clique is the east-coast liberals, while the left 4-

clique are moderates. The Republicans splinter further, and by 0.95 only the two Senators from Georgia remain (Fig. 4.11d).

All told, the political scientist was impressed that motif simplification could highlight many of the features he was already aware of. That the simplified network highlights these known features helps validate the design of the clique motif glyphs, as well as the greedy heuristic for choosing which non-overlapping set of cliques to simplify. Moreover, several new insights came from analyzing these visualizations and then checking other sources like Wikipedia and Politico to provide additional evidence for the pattern.

4.3.2 Lostpedia Wiki Edits

An example of overlapping motif simplification is shown in Fig. 4.12, which represent the bipartite network for the Lostpedia wiki community collected by Beth Foss. Boxes with labels show wiki pages, linked to the colored discs representing their associated editors. The editors are colored and sized according to two measures of their activity in the wiki. Fig. 4.12 shows the initial network, while the Fig. 4.13 shows a simplified version. By combining fan and connector glyphs, I only have 13 nodes to lay out and compare instead of the original 513, only 23 edges instead of 586, and use a fraction of the screen space. While these simplifications are not entirely necessary to understand such a small and well-arranged diagram,

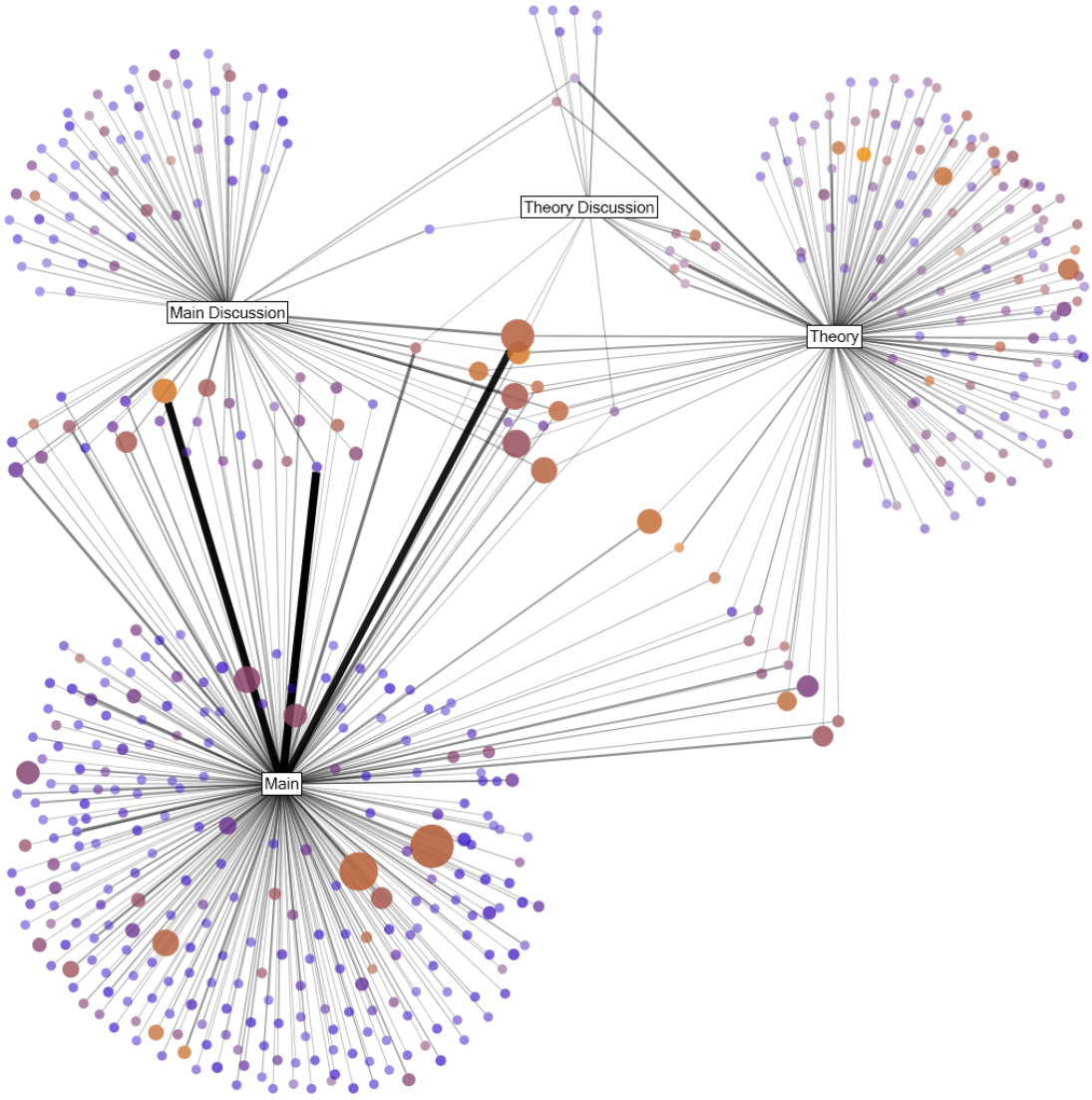


Figure 4.12: A bipartite network of Lostpedia wiki edits showing wiki pages as boxes and their associated editors as discs.

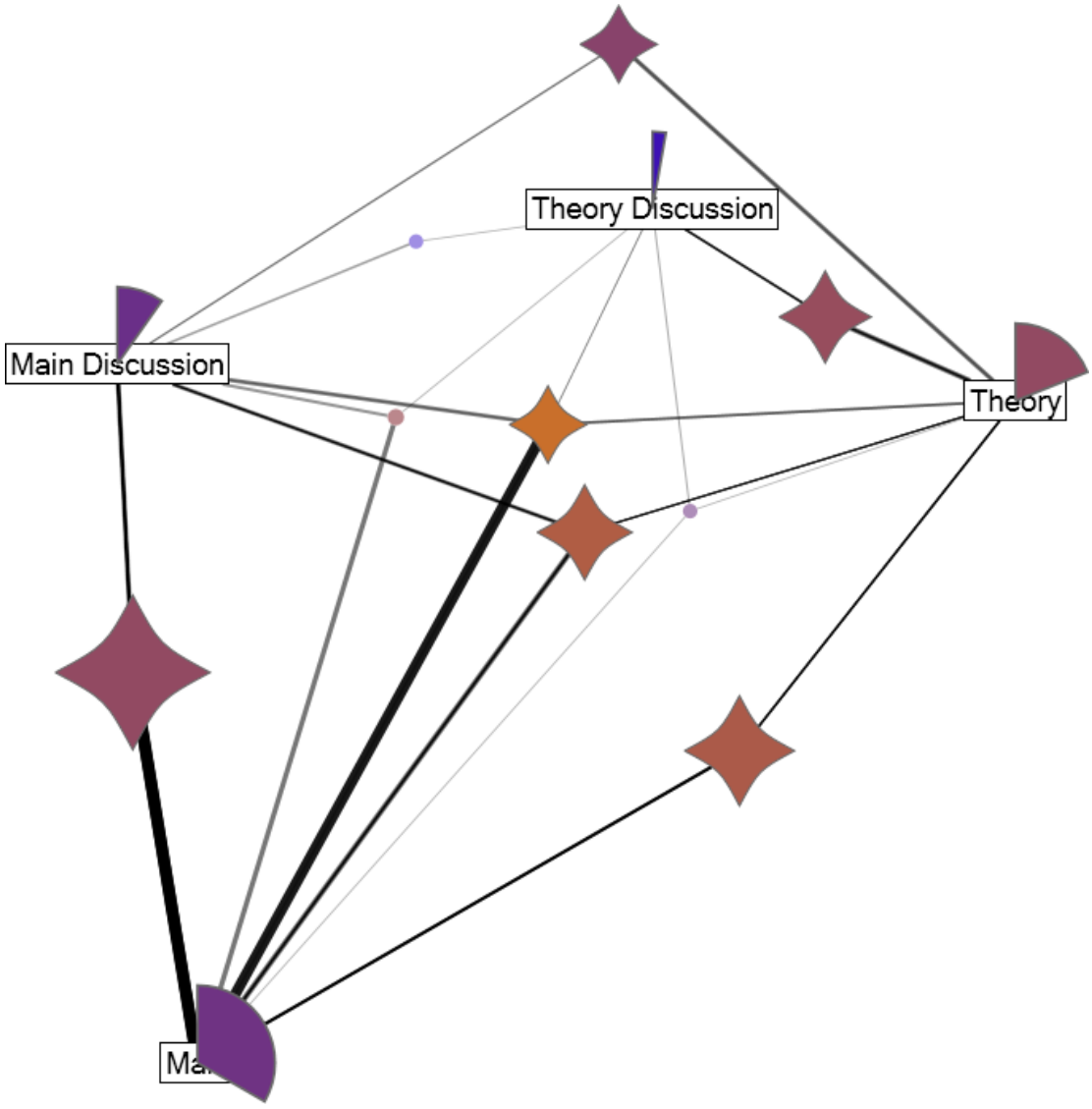


Figure 4.13: The Lostpedia wiki edits after being simplified using fan and connector motif glyphs.

they are effective at showing aggregate relationships like the large number of highly active main page editors.

4.3.3 Ravelry Forums

Another straightforward example I investigated is shown in Fig. 4.14, which I adapted from Fig. 9.10 of the NodeXL book [HSS11, p. 139]. Fig. 4.14a represents the bipartite network for the Ravelry communities collected by a student in Derek Hansen's Communities of Practice class. Three forum nodes shown as small blue discs are connected by the contributers posting in them, with some contributers posting in only one forum and others posting in two. After simplifying the fan and connector motifs present in the network, I created the representation displayed in Fig. 4.14b. Note that the connector glyph used here is the older diamond shape. While these simplifications are not necessary to understand such a small and well-arranged drawing, they are easy to understand.

4.3.4 VOSON Web Crawl

A larger dataset I encountered is shown in Fig. 4.15, which I modified from the NodeXL book, Fig. 12.9 [HSS11, p. 192]. This network of 3958 web pages and 4380 hyperlinks was collected by crawling sites connected to voson.anu.edu.au. It is immediately evident that large fans of nodes dominate the periphery, in in part because the NodeXL [Smi+10] implementation of the Fruchterman-Reingold

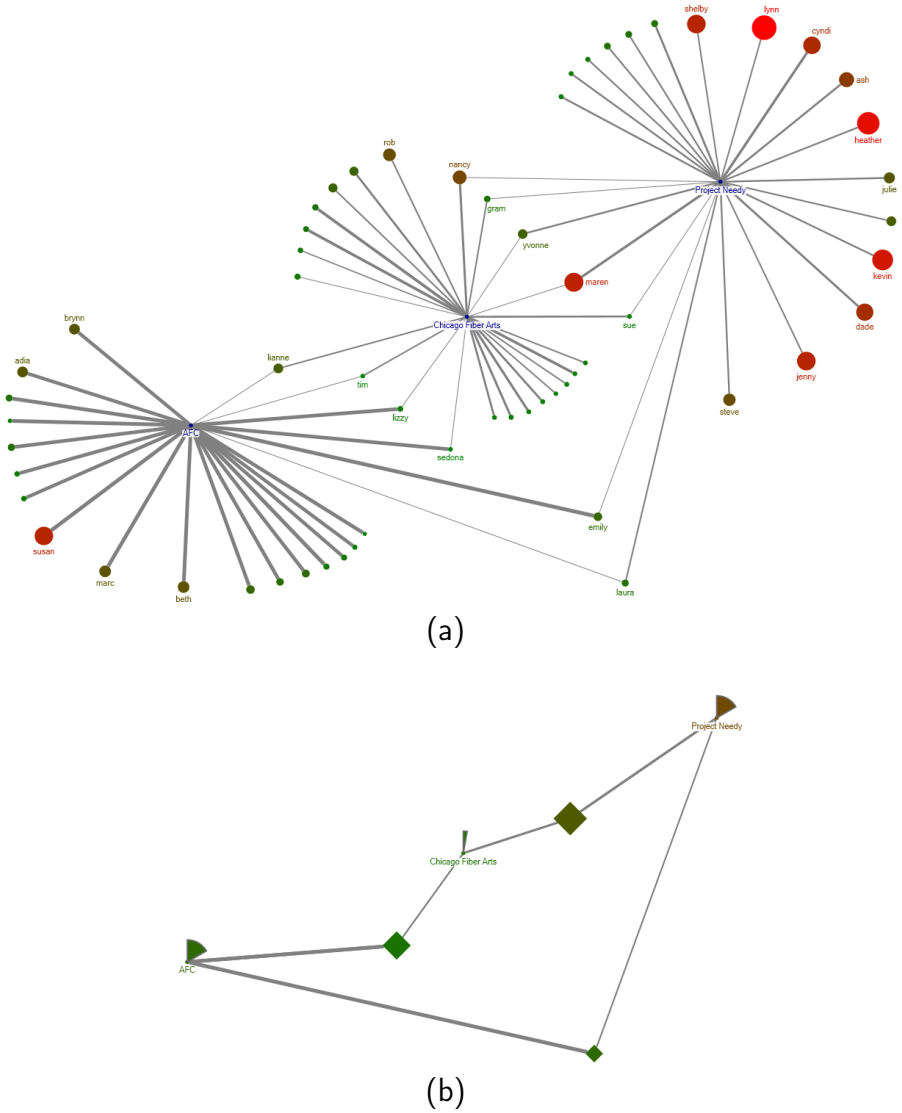


Figure 4.14: This network of relationships between Ravelry forums and their users was created by a student in Derek Hansen's Communities of Practice class. In (a), three forums represented in blue are connected to contributers, and the contributers are sized and colored by the number of completed projects. Edge width is based on the number of posts by each user. This version was adapted from Fig. 9.10 of the NodeXL book [HSS11, p. 139]. (b) shows a simplified version of this network, where the fan and connector motifs have been replaced by representative glyphs. The glyphs are sized by the number of nodes they replace and colored according to the average node attribute value. Likewise, aggregate edges between glyphs are sized and colored by the average of the edge weights of the edges they replace.

layout [FR91] tends to draw elliptical layouts within a rectangular space. However, the fans tend to dominate the visualization regardless of the layout. For example, Fig. 4.16 shows the same graph using the Harel-Koren FMS layout [HK02a].

My manual calculations using Gimp showed that 21% of the screen space in Fig. 4.15 is wasted as blank space in the corners, with 33% showing the core network with its connector motifs, and the remaining 46% used to show the fan motifs. Calculating only for the elliptical visualization region, approximately 58% of the space available is used to show the fan motifs. This is a substantial amount of area dedicated to showing a very common structure in network datasets obtained by crawling web sites or using surveys. Moreover, these fans do not show any information besides the rough number of nodes they contain. The fans in Fig. 4.15 vary from 17 to 852 nodes, but due to overlap this can be hard to see.

Some of the overlap between motifs and and with other nodes is not visible in the original image, but there is substantial overlap in the bottom-right and many of the smaller fans are spread in several directions or hidden in the interior. Some of this is visible in Fig. 4.17, where I have colored and shaped each of the network motifs distinctly. You can see in the bottom-right that the large light green and dark green fans overlap substantially, while many of the smaller fans are spread in several directions or hidden. Moreover, many of the fans overlap and obscure other more important nodes that are not participating in any fan, such

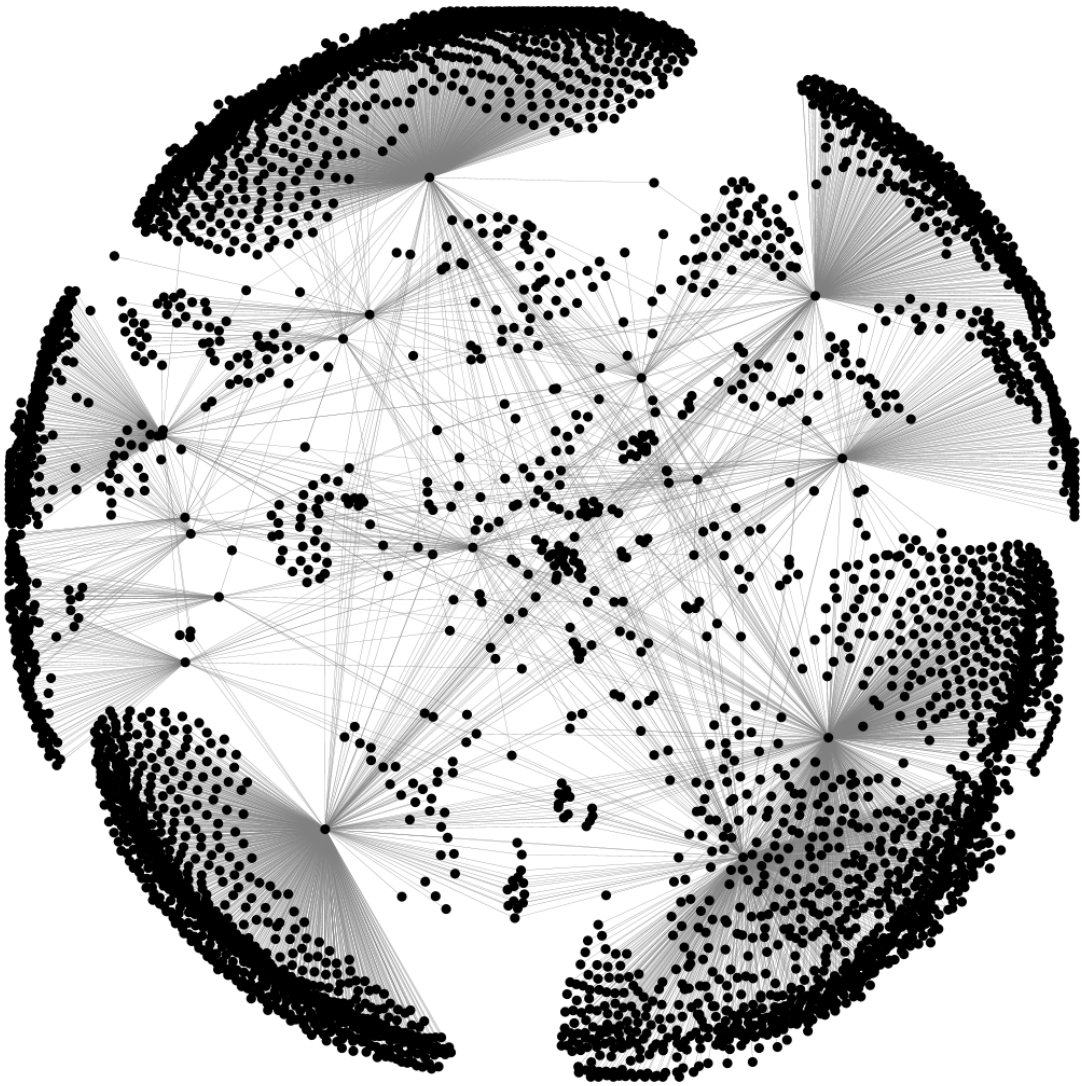


Figure 4.15: This drawing represents the network of web pages connected to vo-son.anu.edu.au obtained by a web crawl. I modified it from Fig. 12.9 of the NodeXL book [HSS11, p. 192]. A similar graph for wiki structure is shown on p. 259. The layout is done using Fruchterman-Reingold [FR91] in NodeXL, and head nodes for the fans of singly-connected nodes are shown in blue.

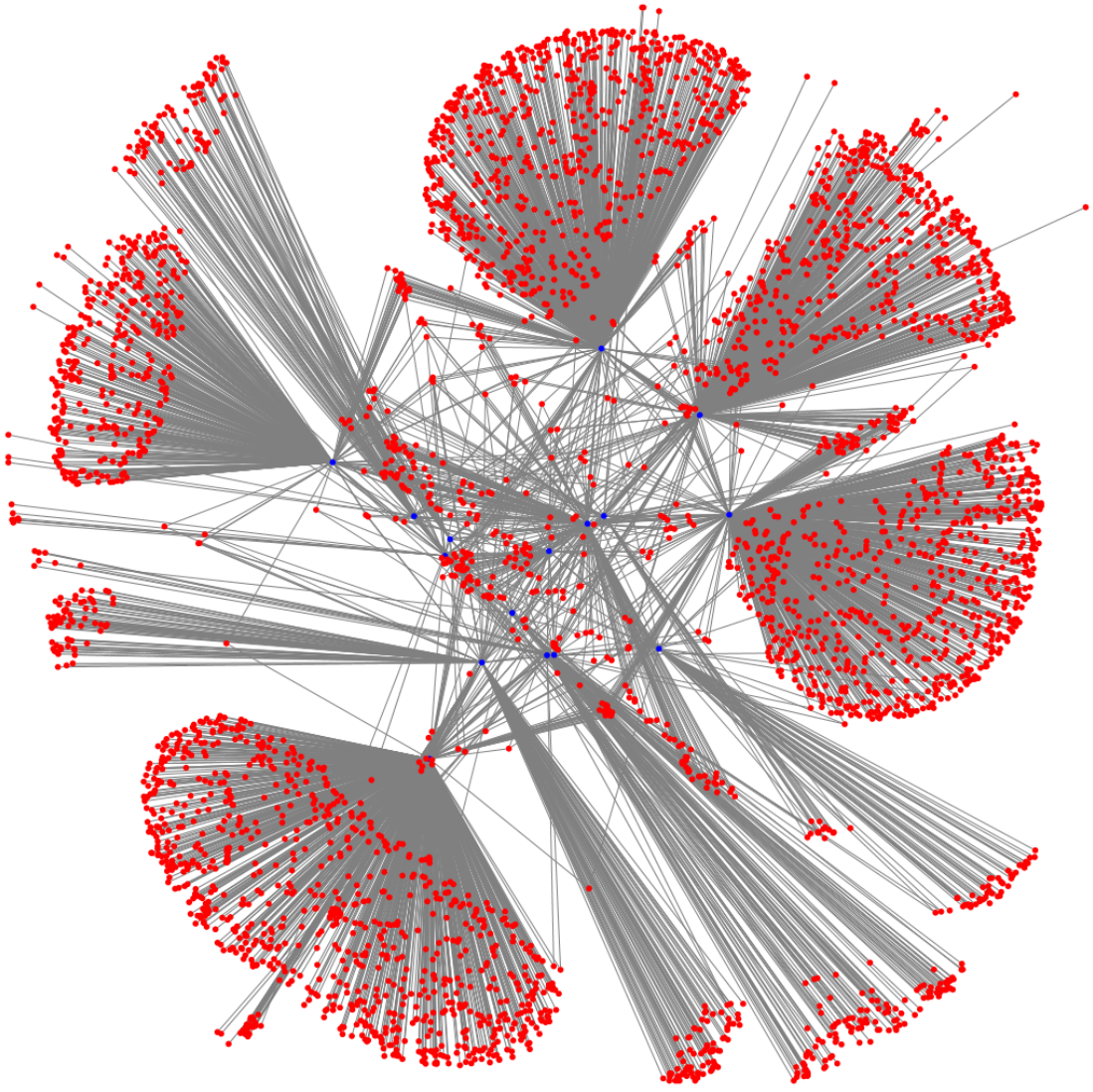


Figure 4.16: A web crawl starting at voson.anu.edu.au, modified from Fig. 12.9 of the NodeXL book [HSS11, p. 192], and laid out using the Harel-Koren FMS layout [HK02a].

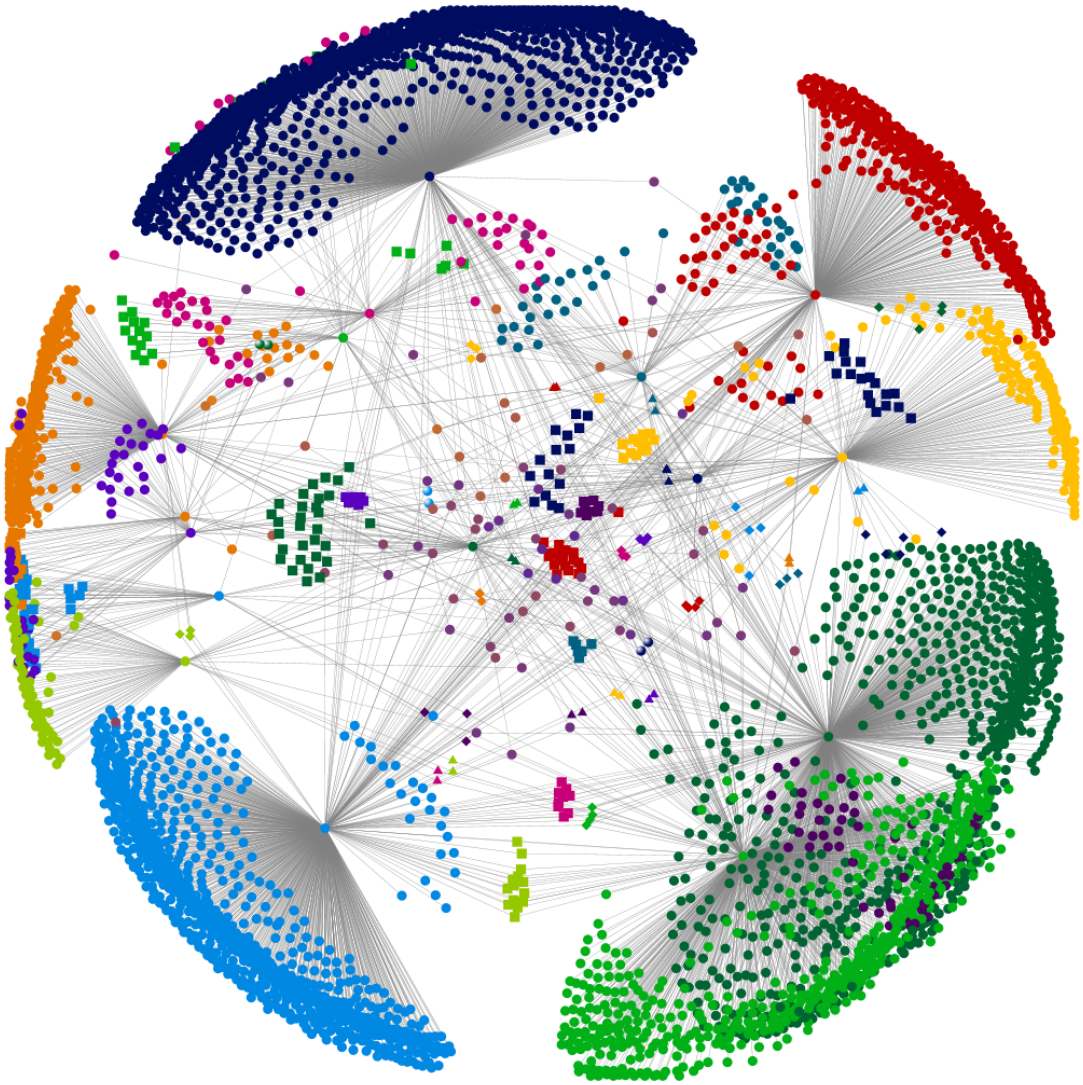


Figure 4.17: Web crawl network with each fan and connector motif shown in a distinct color and shape.

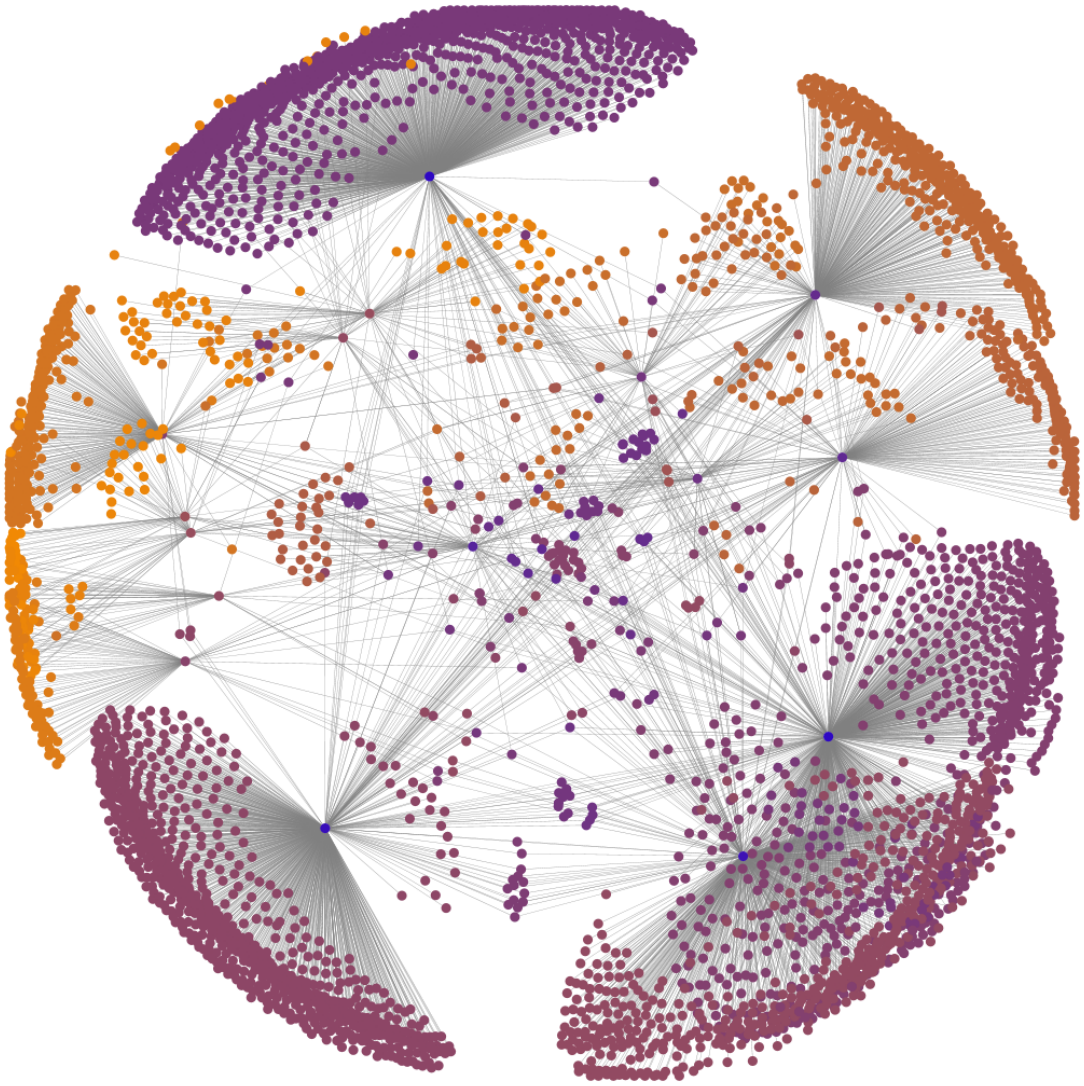


Figure 4.18: Web crawl network with nodes colored by their eigenvector centrality.

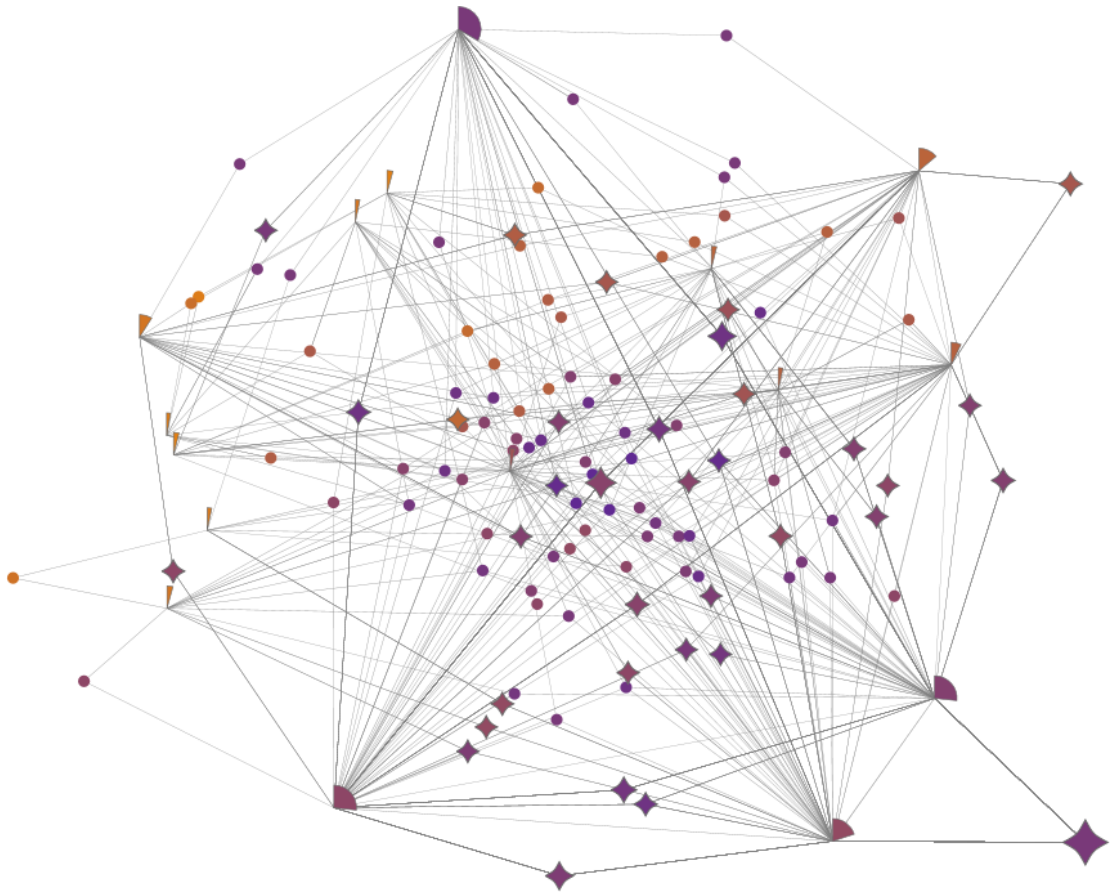


Figure 4.19: Web crawl network with fan and connector motifs simplified and colored by underlying eigenvector centrality.

as a huge 2-connector motif with 50 purple span nodes in the bottom-right. This 2-connector motif, as well as the several others connecting parts of the web page network together, are quite hard to detect among the clutter.

I then simplified these fan and connector motifs, going from 3958 nodes to 559 and 4380 edges to 765, creating a much less cluttered visualization (Fig. 4.19). After simplification, it became evident that the large connector motif is the linked the web sites for the Summer Doctoral Programme at the Oxford Internet Institute

and the National Center for eSocial Science. Applying a layout algorithm to the simplified network would result in a new layout that makes more effective use of the newfound space. This visualization is much clearer at presenting (1) the size and membership of the various fans motifs and (2) the large connector motifs connecting pairs of fan heads. Moreover, it appears to have minimal loss of information and visual clutter compared to the original.

4.3.5 Patient Discharge Summaries

Another complex network to which I have applied motif simplification maps the connections between medical patients and concepts related to their care. These concepts have been extracted from the patient discharge summaries, and include any associated symptoms, diseases, drugs, and procedures. They were provided by Todd Johnson, director of Biomedical Informatics at the University of Kentucky. The goal in analyzing this dataset was to see if motif simplification would help medical researchers understand overall patient trends, such as comparing the efficacy of competing treatments for the same condition.

Dr. Johnson suggested that I investigate two medication concepts in the anonymized network, "hops5325" and "orch7323", where "hops" stands for Hazardous or Poisonous Substance and "orch" indicates Organic Chemical. I extracted from the overall network only those patients connected to "hops5325" and/or

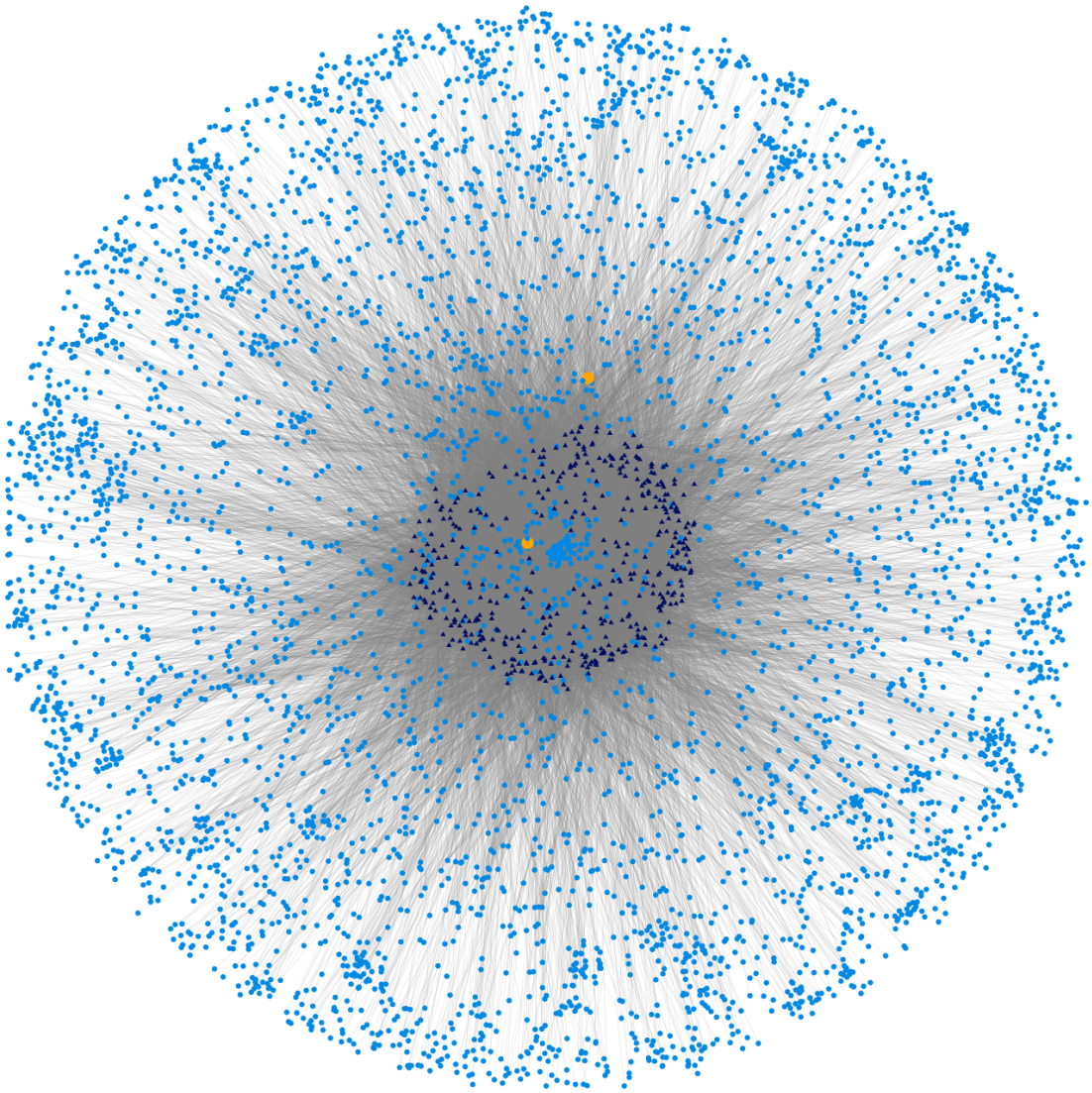


Figure 4.20: Patients related to concepts from their medical discharge reports. This subnetwork focuses on the concepts "hops5325" and "orch7323" (orange discs) and their associated patients (purple triangles) and concepts (blue discs). The network is laid out using the Harel-Koren FMS layout algorithm [HK02a].

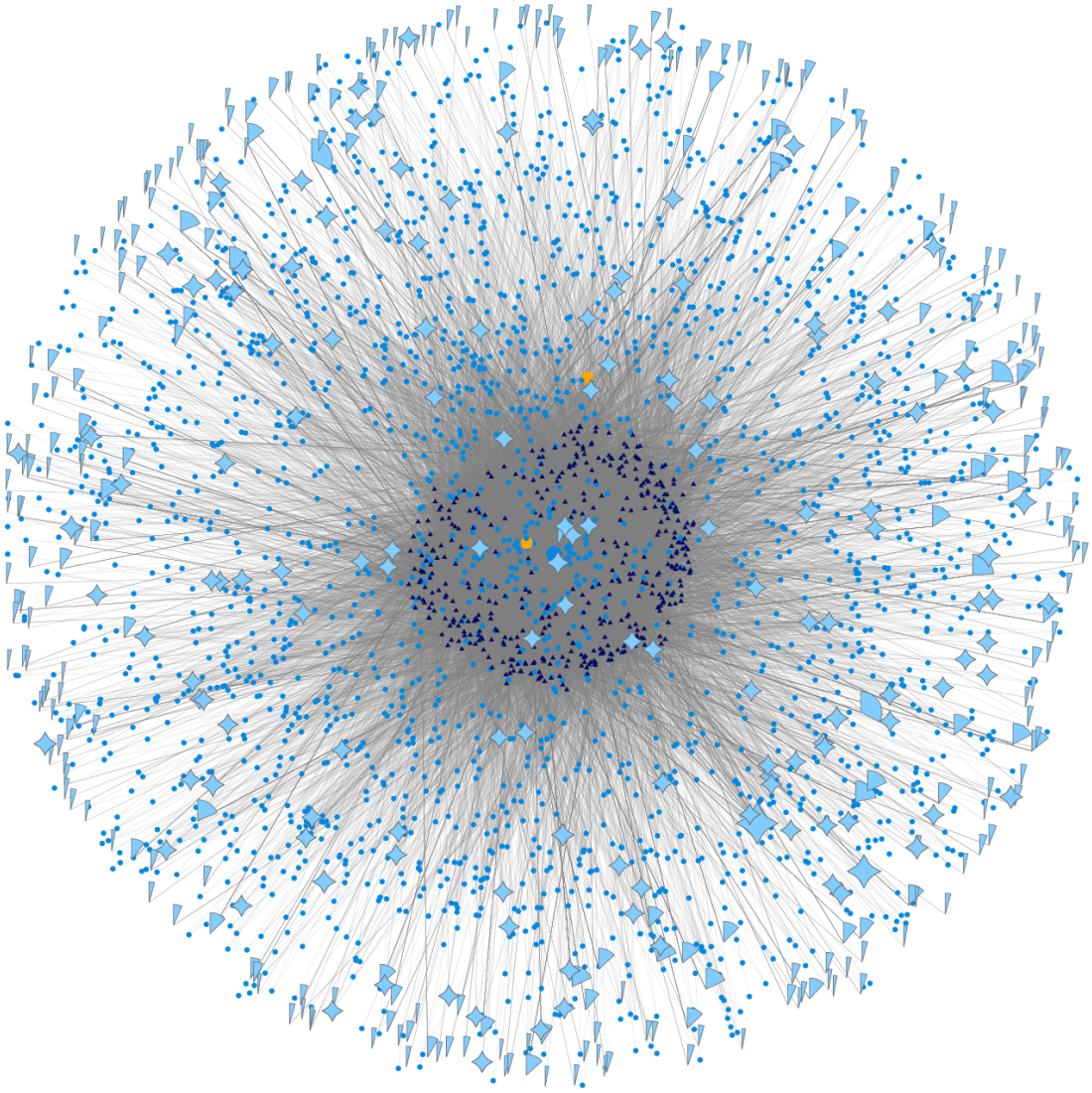


Figure 4.21: Patients and concepts from Fig. 4.20 after applying fan and connector motif simplification.

"orch7323", as well as any additional concepts associated with those patients (a 2-degree subnetwork). This resulted in 433 patients connected to 4701 concepts, including "hops5325" and "orch7323". Fig. 4.20 shows a node-link visualization of this subnetwork using the Harel-Koren FMS layout [HK02a]. The two ego concepts "hops5325" and "orch7323" are shown large and in orange, other concepts are blue, and the patients are purple triangles. This initial view does not show much structure, aside from "orch7323" being more central to the network and connected to more of the patients. Applying motif simplification, specifically the fan and connector motifs, reduces the complexity somewhat but not spectacularly (Fig. 4.21). The exact reduction is from 5134 nodes to 2695 nodes and 439 motif glyphs, and from 31,518 edges to 28,375 edges and meta-edges.

Now that we have the motifs, I can use them to highlight or drill down into interesting patterns. Fig. 4.22 shows the largest fan motifs highlighted in red, where each fan has at least 20 concepts and up to 42 for the largest. These concepts are unique to a single patient, and the patients and their connections to the fans are highlighted in red as well. A medical researcher may be interested in exploring these singleton concept groups and drilling down to them or, alternatively, filtering them out to see the more common patterns. In this case I drill down to show only those patients and their connected concepts, displayed in Fig. 4.23 without simplification. "hops5325" is peripheral to this network, only connected to two patients on the

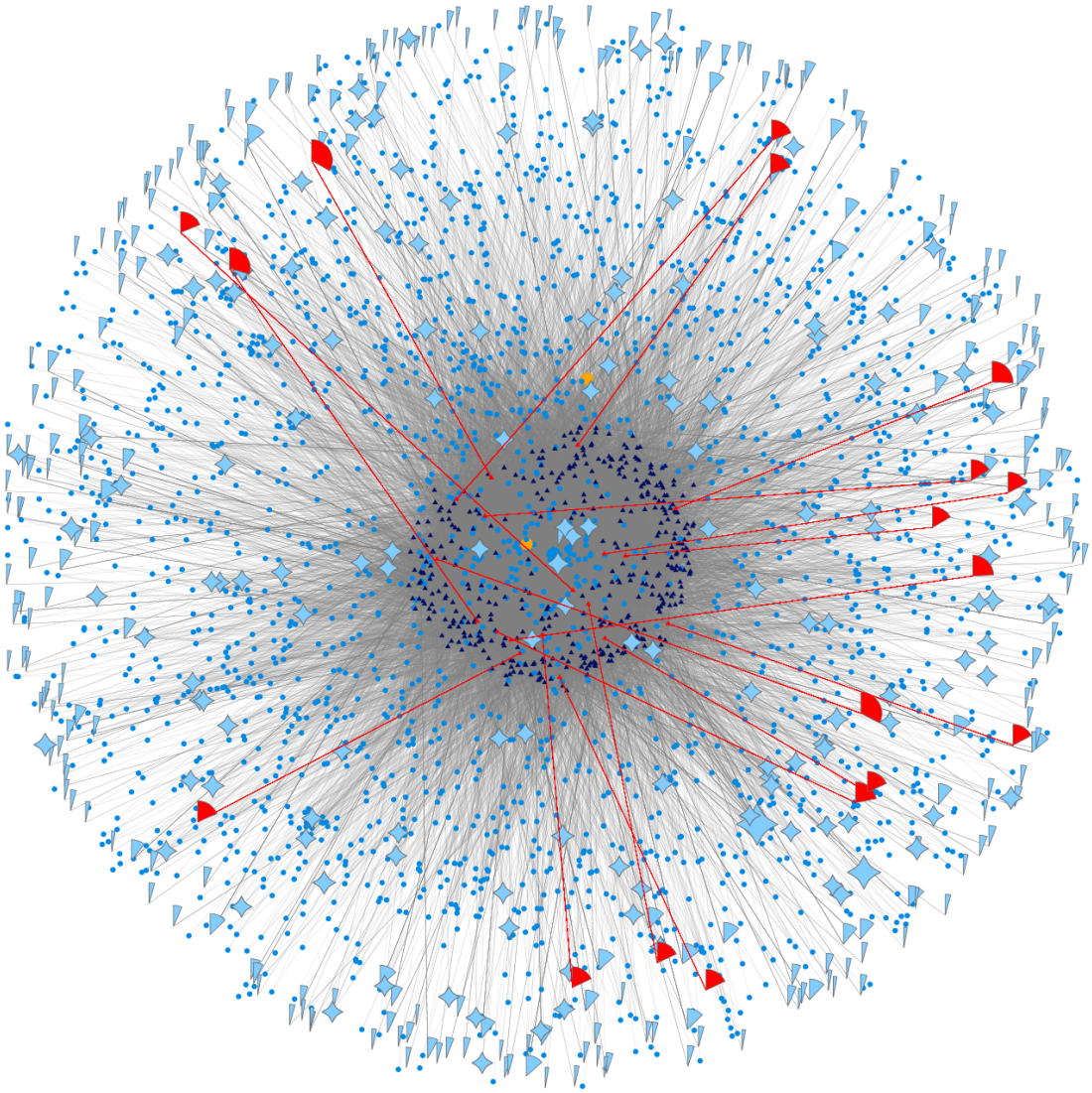


Figure 4.22: Simplified patient and concept network from Fig. 4.21 with fans of 20 or more concepts highlighted. This shows groups of concepts that are uniquely associated with a single patient. Edges from these fans to their associated patient, as well as the patient themselves, are highlighted too.

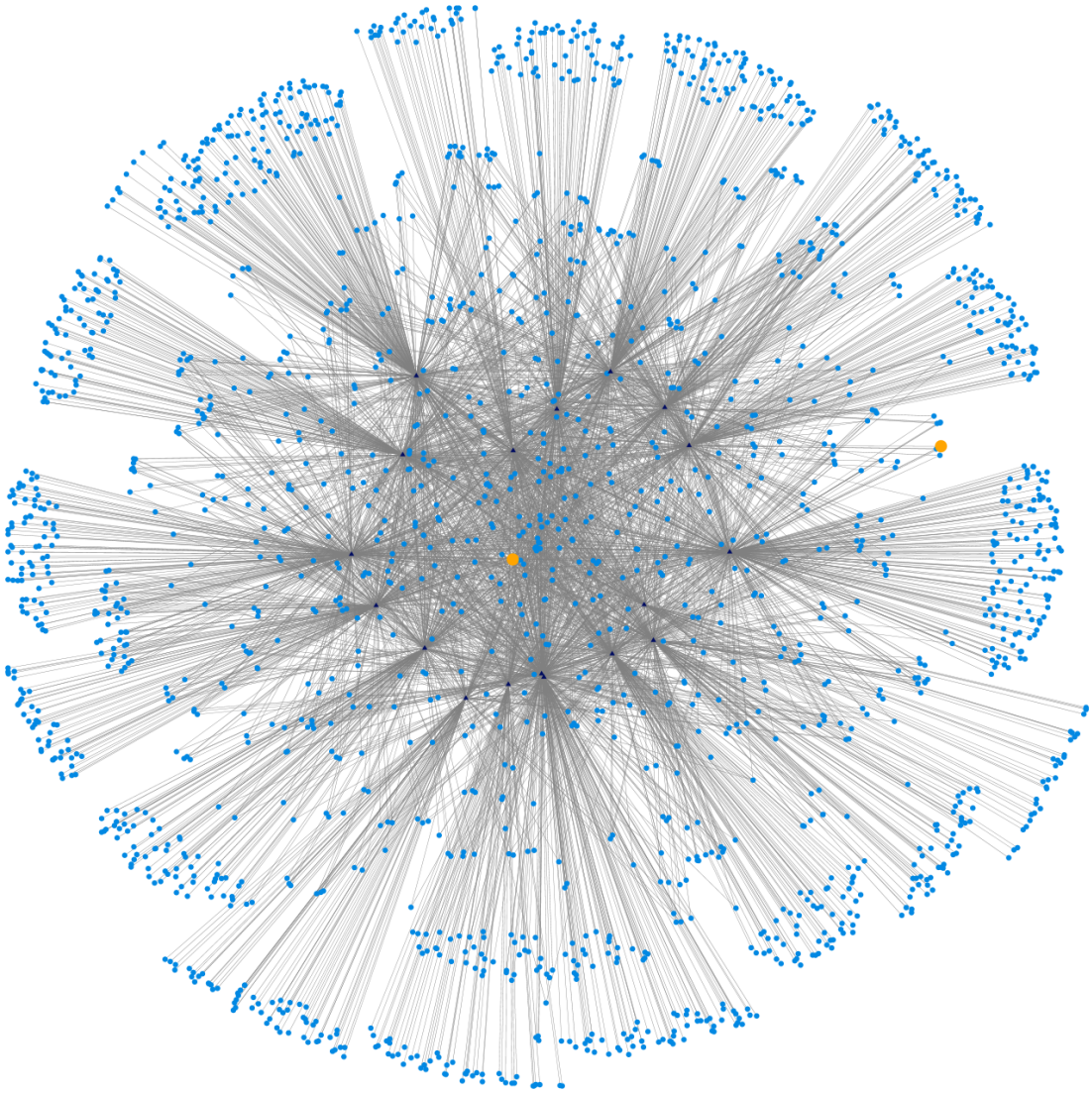


Figure 4.23: Patient and concept network of only the patients connected to the large highlighted fans from Fig. 4.22, as well as any associated concepts. The initial "hops5325" concept is on the far right, connected to only two patients.

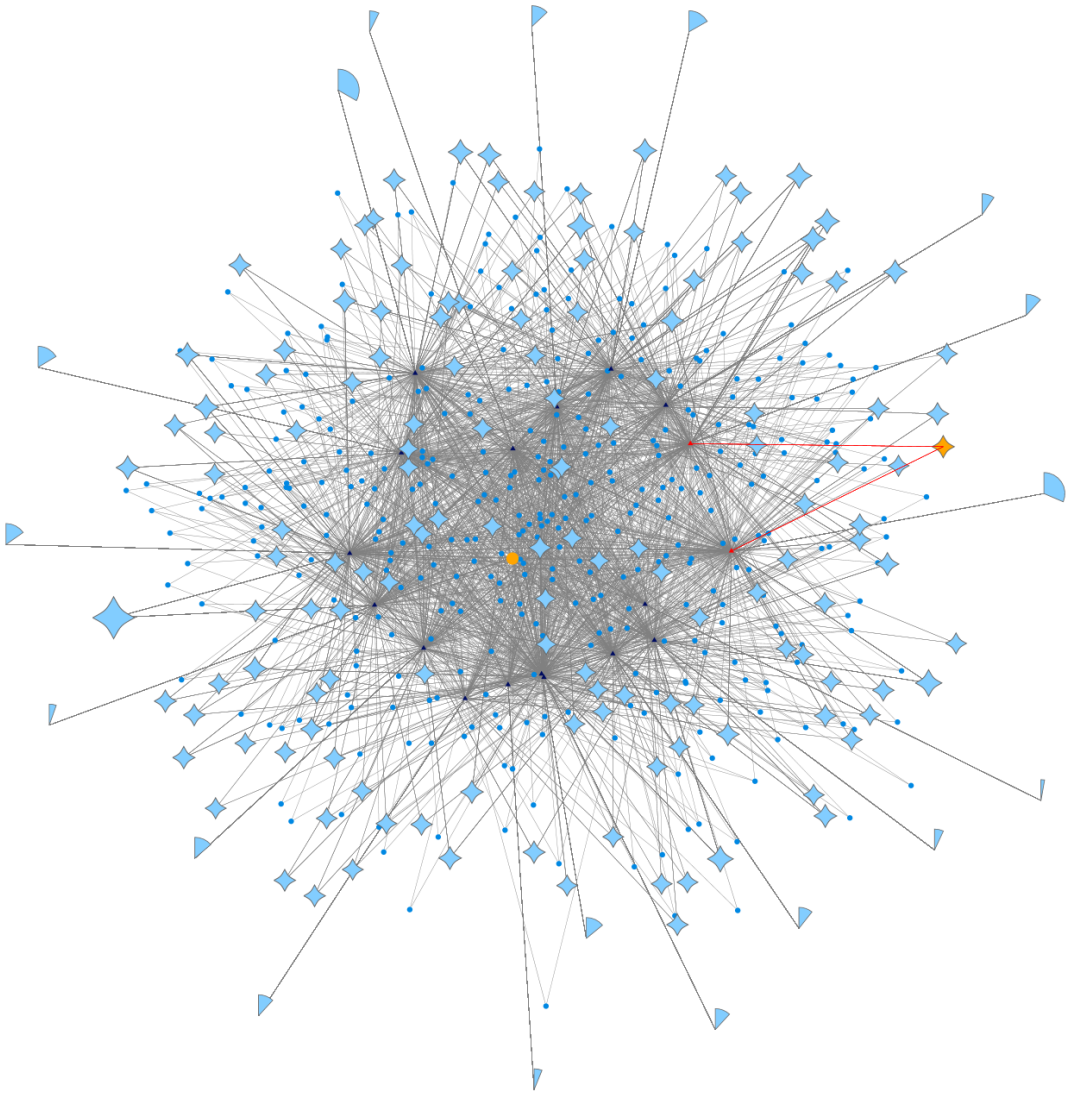


Figure 4.24: Patient and concept network from Fig. 4.23 after applying motif simplification. The connector motif which contains the initial "hops5325" concept and three other concepts is highlighted in orange. These four concepts are only connected to two patients.

right. In the simplified view (Fig. 4.24), "hops5325" is in a connector motif with three other concepts that are only connected to those two patients: "orch7268", "hlca5025", and "hlca5238". Interestingly, only one of these patients is connected to "orch7323". Another pattern of note is the large connector motif on the left, which consists of 36 concepts associated with two other patients who are connected to "orch7323". These concepts are "aapp155"; "dsyn 2382, 2732, 2842, 3006, 3092, 3171, 3464, 3576, 3577, 3817, 3837, 3927, 4009, 4261, 4528, and 4827"; "lbpr 5981, 5990, and 6419"; "mobd 6668, 6673, 6688, 6690, and 6715"; "orch 7921, 8368, and 8369"; "patf 8787, 8818, and 8983"; "phsu9097"; and "topp 10357, 10429, and 10856".

An alternate kind of exploration is visible in Fig. 4.25, where I have highlighted connectors of concepts connected to at least 20 patients. These small connectors consist of two or more concepts that occur with many patients in the exact same way, but the connectors each have different sets of the 433 patients as anchors.

The true power of motif simplification becomes evident when I drill down to only show the patients connected to four specific concepts. I chose our original "hops3525" and "orch7323", as well as two other Hazardous or Poisonous Substances: "hops5323" and "hops5324". The node-link visualization of these relationships is partially understandable (Fig. 4.26), but after applying motif simplification the aggregate patient relationships between the concepts are much more clear (Fig. 4.26). Note that here the motifs consist of patients, not concepts.

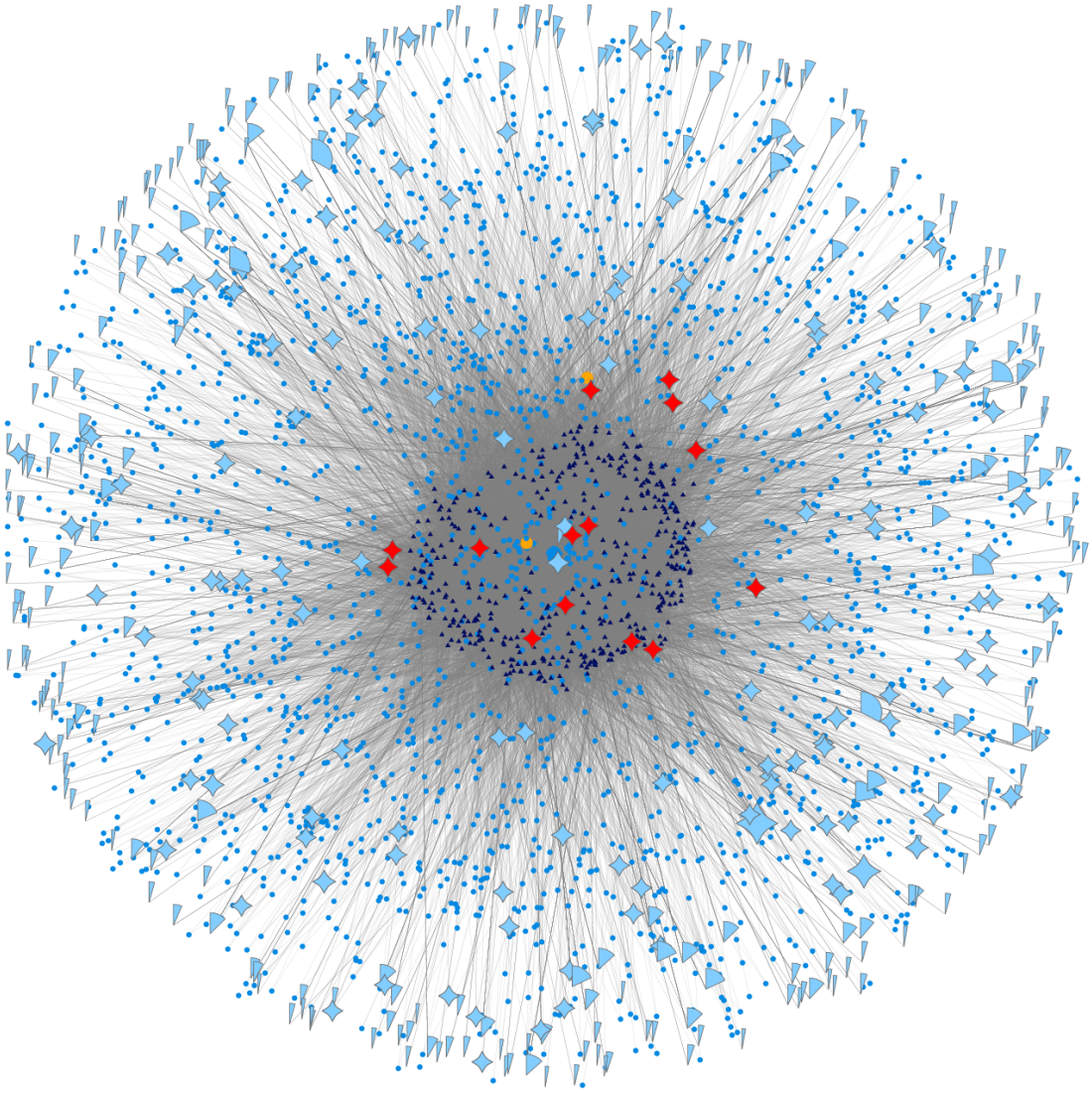


Figure 4.25: Patients and concepts from the original simplified view in Fig. 4.21. Connector motifs of concepts connected to at least 20 patients are highlighted.

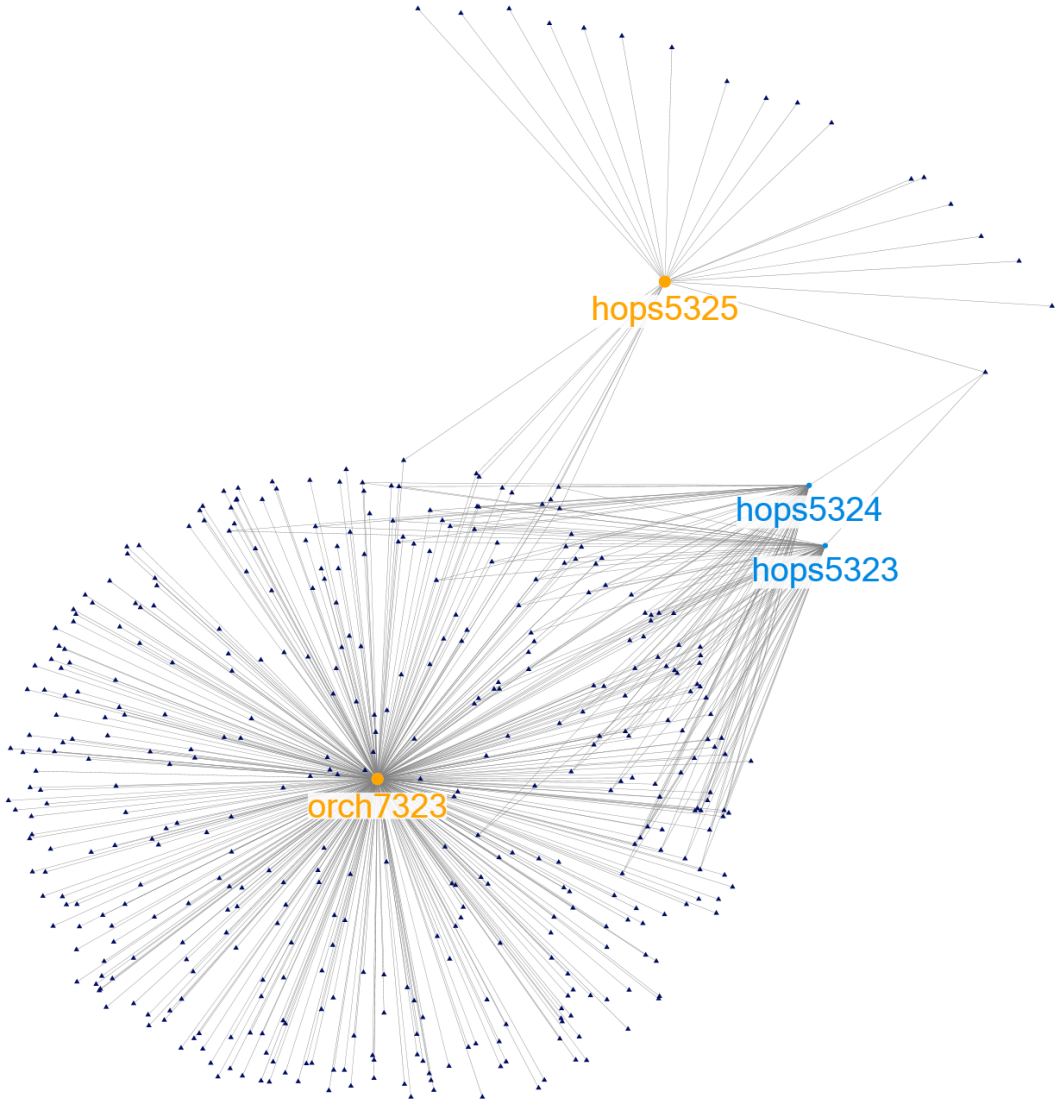


Figure 4.26: Patients and concepts from Fig. 4.20, after drilling down to only those patients connected to our original "hops3525" and "orch7323", as well as two other Hazardous or Poisonous Substances: "hops5323" and "hops5324".

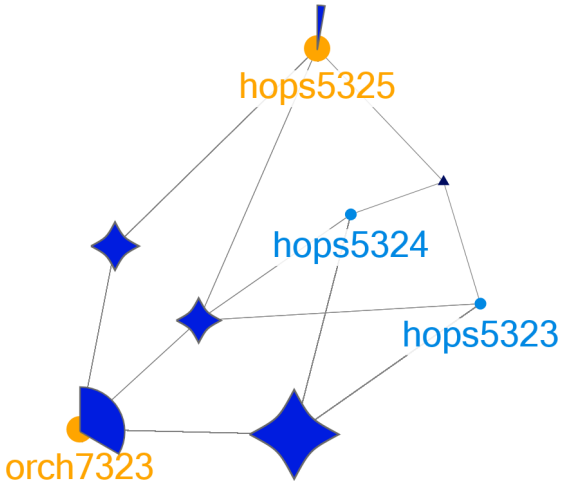


Figure 4.27: A simplified view of the patients and concepts in Fig. 4.26, which highlights the aggregate patient relationships between the concepts.

It is immediately visible that two patients are connected to all four concepts and one patient is shared between only the "hops" concepts. Another 7 patients connect "orch7323" and "hops5325" while 67 connect "hops5323", "hops5324", and "orch7323". Of course, 339 patients only have "orch7323" as a concept while only 17 are only connected to "hops5325".

Overall, I believe that motif simplification can definitely help medical researchers understand the relationships between patients and a small number of concepts, as in Fig. 4.27. For larger datasets with thousands of concepts, the motifs seem to highlight particularly unusual connections like large groups of concepts associated with one patient or a few patients. To understand these relationships in detail, the motifs can be used to drill down to the relevant parts of the network. For additional analyses of this network using Group-in-a-Box layouts, see Section 5.5.3.

4.3.6 Larger Networks

I analyzed several other large networks not pictured here. One was a network of innovation and funding ties with 7124 nodes and 16,109 edges. Another showed acquisitions of JP Morgan Chase, with 5766 nodes and 6752 edges. Both were visualized interactively with no performance issues, and had drastic reductions in complexity with motif simplification.

4.4 Initial Usability Study

I invited four individuals from our lab to use the motif simplification techniques inside NodeXL in order to understand any usability issues and general ease of use. I asked them to analyze three networks: Lostpedia wiki edits (Section 4.3.2), the VOSON web crawl (Section 4.3.4), and a network of innovation in Pennsylvania used as a Group-in-a-Box layout case study (Section 5.5.2). These participants had varying backgrounds, including Computer Science, Information Studies, and Economics. They also had varying education, including a recent undergraduate student, two graduate students, and a professor. All had little or no experience with NodeXL and none with motif simplification.

After an initial hands-on training session I invited participants to explore the networks and recorded anything they had difficulty with or mentioned. Their explorations ranged from 45–60 minutes. Overall they were excited by the motif

simplifications, and were especially eager to change to the simplified version in the VOSON example. One of them stated about the original VOSON view, "I'm overwhelmed, ... this is like one of those vision tests at the eye doctor", but when asked to switch to the simplified view emphatically stated, "Yes please!". Asked afterward about her overall impression of motif simplification, one participant said, "I like it because it makes more sense. For specific nodes it is easier to look at the spreadsheet side". No participant detected the bottom-right connector motif hidden in the VOSON fan motifs, but did immediately in the simplified view.

There were several issues the participants encountered. First, they wanted to simplify all repeating patterns they saw, not just my defined motifs. One even did the simplification manually using standard meta-nodes. Next, they were unsure about the design of the crescent connector motif used at the time. They did not understand why edges connected to the arch in several places instead of only the corners, and had difficulty comparing connector glyph size exactly. A few even confused the connector glyphs with overlapping or odd fan glyphs. I revised my glyph design based on this feedback to more effectively allow these analyses, as discussed in Section 4.2.1.1.

In spite of these challenges, participants strongly appreciated the benefits of simplifying complex networks and expressed enthusiasm for integration of the glyphs in node-link visualizations. By replacing the common repeating motifs

with representative glyphs, many nuances of the network are revealed. When one participant was looking for relationships, she stated, "I could only look at two at a time". This seems to indicate that the simplified view will help users understand larger relationships in the network, as glyphs allow comparisons of larger subsets of the network and reduce the number of analyses.

4.5 Controlled Experiment

The usability testing guided any necessary interface revisions. Then, I ran a controlled experiment to determine the effect motif simplification has on user performance across several common network visualization tasks.

4.5.1 Tasks

I chose a varied set of tasks relating to topology, attributes, and overviews from a taxonomy [Lee+06], which demonstrates how all complex tasks can be seen as a series of low-level tasks. These tasks are also used in many recent papers evaluating network visualizations [HF07; SA06; GFC04]. I asked:

1. About how many nodes are in the network?
2. Which individual node would we remove to disconnect the most nodes from the main network?

3. Which is the largest (fan | connector | clique) motif and how many nodes does it contain?
4. Which node has the label "XXX"? (where XXX was a name or number)
5. What is the length of the shortest path between the two highlighted nodes?
6. Which of the two highlighted nodes has more neighbors?
7. How many common neighbors are shared by the two highlighted nodes?
8. Which of two pairs of nodes has more common neighbors?

4.5.2 Data

Current random network generators do not produce realistic data [HF07], which I confirmed trying to generate several networks with similar characteristics. Thus I chose to use three interesting networks produced by actual users solving their own problems. Lostpedia wiki edits (Section 4.3.2), U.S Senate voting patterns (Section 4.3.1), and the VOSON web crawl (Section 4.3.4).

4.5.3 Participants

I began with a pilot study with two participants from my lab, in which the tutorial and format of the questions were refined. I then recruited 36 students from my university (19 males, 17 females) using mailing lists and in-class announcements. The participants were mostly graduate students, half from Computer Science and

the balance from eight other departments. 9 had used network visualization tools and an overlapping 9 had seen motif simplification, though none had used it. As I could not generate sufficiently varied datasets with similar properties, I used a between subjects design. I randomly divided participants into two groups which had similar distributions of gender, department, grade level, and experience.

4.5.4 Procedure

Each 45-minute session began with 5-10 minutes of training on the tool and for the specific tasks, followed by about 35 minutes for answering a total of 31 questions across the three networks and eight tasks. Each participant received the same order of questions and visualizations. The control group was provided with an interactive node-link visualization in which they could select nodes along with their incident edges, as well as move the nodes. The treatment group received a simplified version of each new visualization, with additional interactive tooltips and the ability to expand and collapse the motifs. Each visualization is presented consistently, originally computed using the Harel-Koren FMS layout [HK02a].

As in [GFC04; HF07], users were given one minute to answer each question, told to answer as quickly and accurately as possible, and that they could skip if they could not answer a question. The evaluator spoke each question, gave the participant time to ask for clarification, then revealed the next visualization in turn

and began the timer. Participants were told how well they performed at the end of the study. Users were given $10 plus a $15 bonus for the fastest, most accurate participant in each group.

4.5.5 Analysis

The recorded data was analyzed in several ways. As is common with response time data, the response times were not normally distributed so were normalized using a log transformation. The two groups were then compared using a t-test. Answers to questions consisted of a categorical answer (a specific item), which was recorded correct or not, and/or an integer answer. For questions with categorical answers, the groups were compared with Fisher's exact test instead of the chi-square test as none of the statistically significant group-by-correct matrices had expected values of five or higher in all four cells. For numeric answers I computed $error = (answer - truth)/truth$, skipping any questions that had an incorrect categorical answer the integer answer depended on, and compared error across groups using a t-test.

4.5.6 Results

Here I report only the statistically significant findings, though all the analyses are shown in Figs. 4.28 to 4.35. I expect overview tasks like identifying the maximal motif of a type would be easier with the less visual complexity of a simplified

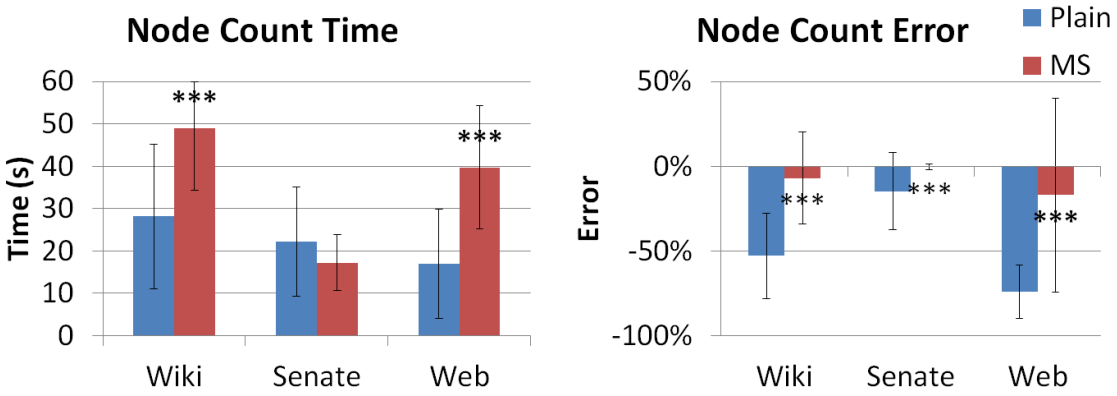


Figure 4.28: Bar charts showing performance for Task 1: "About how many nodes are in the network?" The left chart shows the time spent answering the question while the right chart shows the error in the node count estimate. In this chart, and in the following ones, error bars indicate one standard deviation and asterisks show the level of significance of the statistical test ('*', '**', and '**' denote $p<0.10$, 0.05, and 0.01 respectively). Negative numbers, if present, show the number of users that skipped the question or ran out of time.

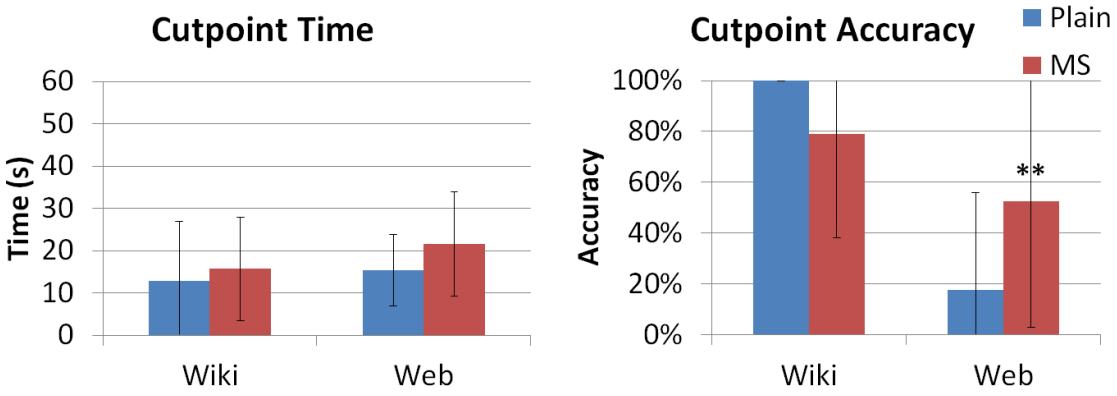


Figure 4.29: Bar charts showing performance for Task 2: "Which individual node would we remove to disconnect the most nodes from the main network?" The left chart shows the time spent while the right chart shows the accuracy at selecting the correct node.

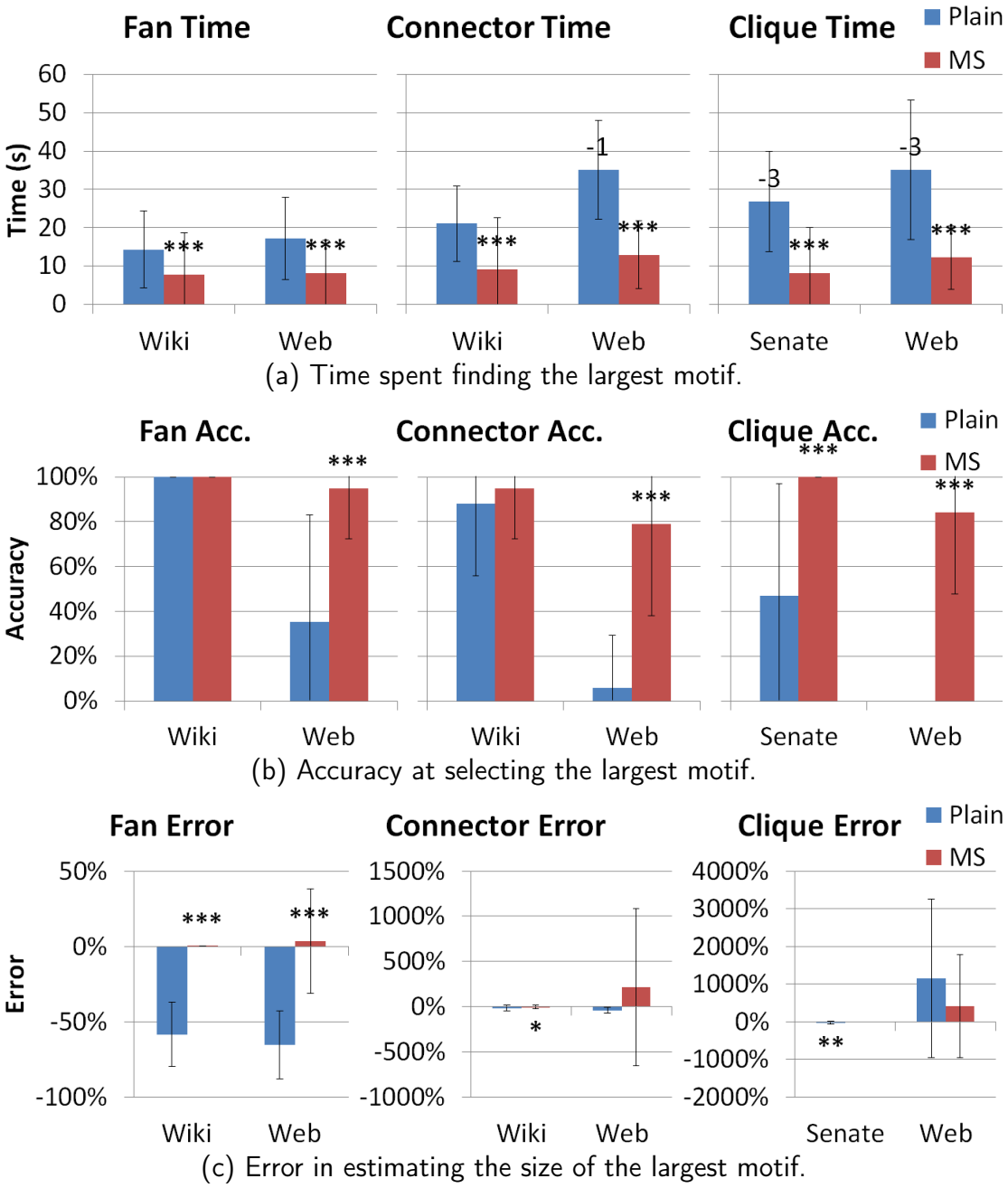


Figure 4.30: Bar charts showing performance for Task 3: "Which is the largest (fan | connector | clique) motif and how many nodes does it contain?" The left charts show the results for fans, the middle for connectors, and the right for cliques.

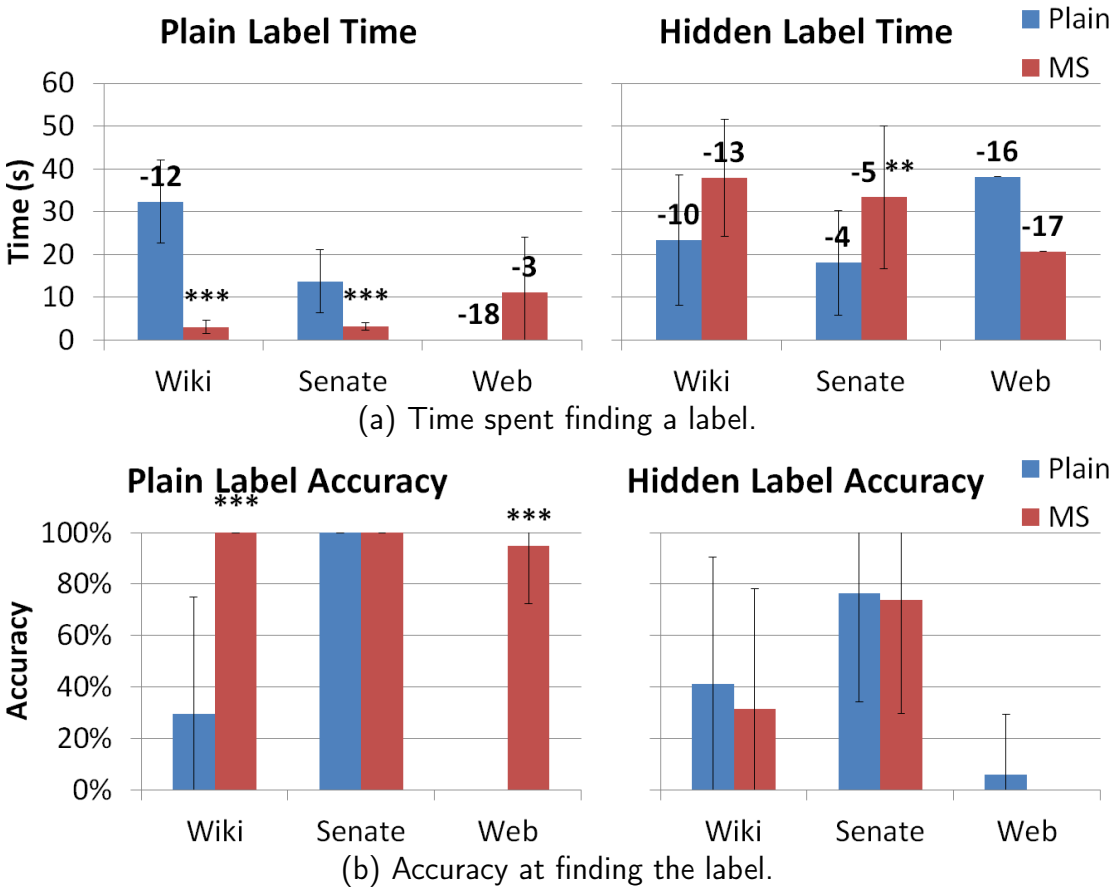


Figure 4.31: Bar charts showing performance for Task 4: "Which node has the label "XXX"? (where XXX was a name or number)" The left charts are for plainly visible nodes, while the right show labels hidden inside a simplified glyph.

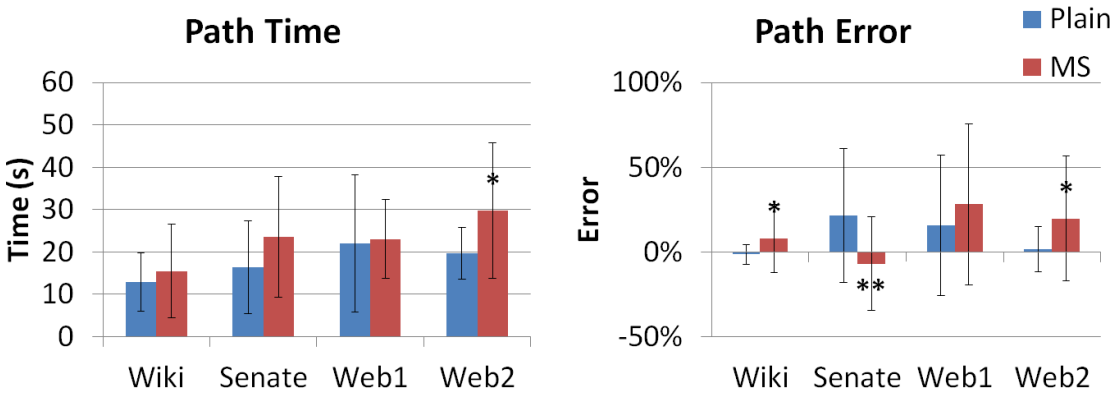


Figure 4.32: Bar charts showing performance for Task 5: "What is the length of the shortest path between the two highlighted nodes?" The left chart shows the time spent while the right chart shows the error at estimating path length.

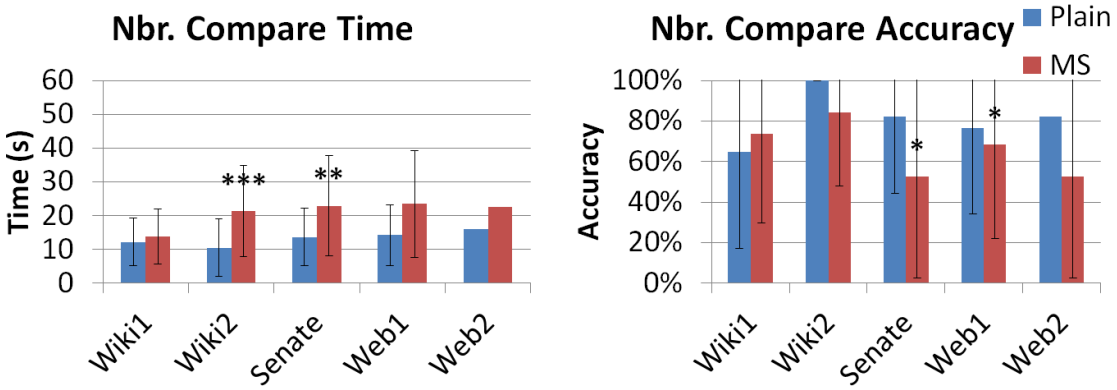


Figure 4.33: Bar charts showing performance for Task 6: "Which of the two highlighted nodes has more neighbors?" The left chart shows the time spent while the right chart shows the accuracy at selecting the correct node.

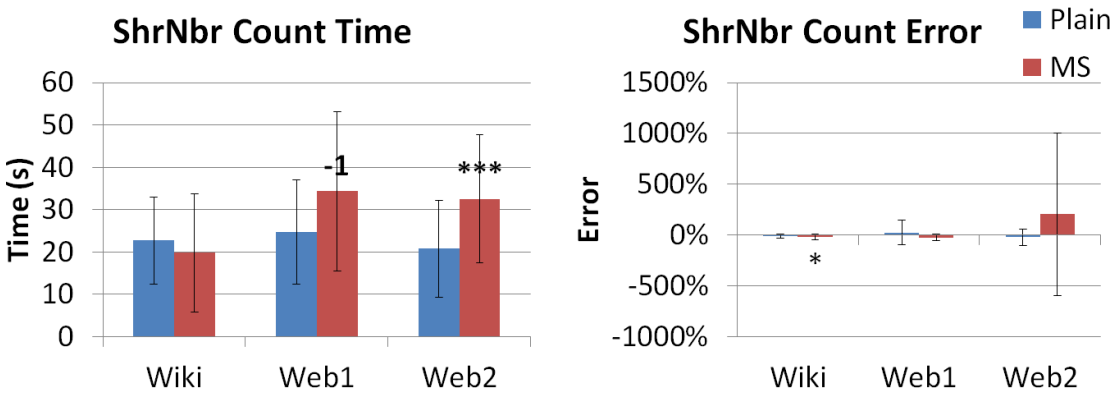


Figure 4.34: Bar charts showing performance for Task 7: " How many common neighbors are shared by the two highlighted nodes?" The left chart shows the time spent while the right chart shows the error in the shared neighbor count estimate.

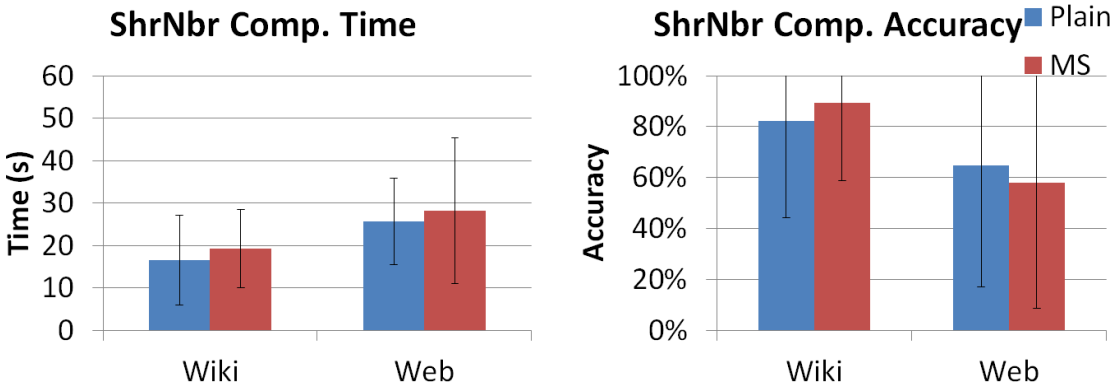


Figure 4.35: Bar charts showing performance for Task 8: "Which of two pairs of nodes has more common neighbors?" The left chart shows the time spent while the right chart shows the accuracy at selecting the correct pair of nodes.

network. This was true for all three motifs across all three networks (Fig. 4.30). Cliques, the epitomical clusters, were found in the two networks they occurred in faster ($p<0.01$, -20.82s), more accurately ($p<0.01$, 92% vs. 23.5%), and with fewer people giving up (3 vs. 0). Moreover, in the Senate network there was higher accuracy in size estimates ($p<0.05$, 0% vs. -28% error), which could be true for the web network but I could not measure it as not one control participant detected the maximal 5-clique. Fans were found in both the networks they occurred in faster ($p<0.01$, mean -7.77s) and their size was approximated more closely ($p<0.01$, 2% vs. -62% error). In the large web network the maximal fan was also found more frequently ($p<0.01$, 95% vs. 35%). Connectors were detected in both their networks faster as well ($p<0.01$, mean -17.13s). In the web network the largest connector was found more frequently ($p<0.01$, 79% vs. 6%), and in the wiki network its size was estimated more precisely ($p<0.1$, -5% vs. -17% error).

These results show that using glyphs for motifs makes the motifs easier to detect and measure, but how does simplifying motifs affect the rest of the network? I hypothesized that estimating the number of nodes would be easier in the simplified, interactive view. As Fig. 4.28 shows, my participants could indeed gauge the size of all three networks with significantly more accuracy ($p<0.01$, -8% vs. -47% error), but for the wiki and web networks users took longer to do so ($p<0.01$, 21.82s). How about finding a specific node by its label? Logically reducing the number of

visual items makes finding a label easier. My results in Fig. 4.31 show that finding labels that are not in motifs is significantly faster ($p<0.01$, -19.93s), they are found more frequently except in the Senate case ($p<0.01$, 97.5% vs. 14.5%), and fewer users give up or run out of time (12 did on the plain wiki and web networks). I only saw worse search time for labels in motifs for the Senate clique case ($p<0.05$, 15.29s), with no significant differences in accuracy.

What about topology-based tasks? It seems that with fewer items on the screen tracing edges would be easier. For some questions it did turn out better, like finding the node to cut (Fig. 4.29) in the web network correctly ($p<0.05$, 53% vs. 18%) and the accuracy of the shortest path length (Fig. 4.32) between two clique members in the Senate network ($p<0.05$, -7% vs. 22% error). For others topology questions, the results were mixed to poor. Shortest path length time and accuracy (Fig. 4.32) worsened in the web network ($p<0.1$, 10.06s & 20% vs. 1% error). Comparing the number of neighbors (Fig. 4.33) was slower on the wiki ($p<0.01$, 10.89s) and senate ($p<0.05$, 9.26s) networks, and the choice accuracy dropped for the senate ($p<0.1$, 53% vs. 82%) and web ($p<0.1$, 68% vs 76%). Lastly, the shared neighbor count tasks (Fig. 4.34) were slower in the web network ($p<0.01$, 11.73s), and reduced accuracy in the wiki network ($p<0.1$, -21% vs. -10%). There were no significant differences in the task to find which of two pairs of nodes has more common neighbors (Fig. 4.35).

4.5.7 Discussion

Overall it appears that motif simplification is beneficial for many analysis tasks. Naturally identifying maximal motifs is faster, more accurate, and I can estimate their sizes more accurately when I have glyphs and interaction. Counting nodes in the network turned out to be slower, but more accurate when using the glyphs. Finding unsimplified labels became much quicker, while simplified labels were only slower in one case. Finally, it seems like topology-based tasks are a mixed bag. Finding cut nodes is more accurate, but path-based tasks were better and worse in different circumstances. Comparing the number of neighbors and shared neighbors turned out slower and less accurate in a few cases, while counting them was more error-prone.

I have already implemented additional features to increase user performance on topologic tasks. When I ran the study I did not yet use the sized meta-edges that are shown in Figs. 4.9 to 4.11. With this simple modification, I believe we can show much of the aggregate connectivity. However, user education is likely the most promising way to improve the glyph performance. Many participants had difficulty understanding the topology inside the collapsed glyphs.

It is important to note that the participants generally had little to no experience with network analysis, nor did they necessarily have any interest in or knowledge of the networks they were analyzing. Despite these limitations, I found significantly

better task performance with the simplified view in many cases. With more than the 5-10 minutes of training provided in this study, user performance would likely improve on many of the tasks.

4.6 Summary

Analyzing networks involves understanding the complex relationships between entities, as well as any attributes they may have. The widely used node-link visualizations excel at this task, but many are difficult to extract meaning from because of the inherent complexity of the relationships and limited screen space. To help address this problem I introduce a technique called **motif simplification**, in which common patterns of nodes and links are replaced with compact and meaningful glyphs. Well-designed glyphs have several benefits: they (1) require less screen space and layout effort, (2) are easier to understand in the context of the network, (3) can reveal otherwise hidden relationships, and (4) preserve as much underlying information as possible. I tackle three frequently occurring and high-payoff motifs: **fans** of nodes with a single neighbor, **connectors** that link a set of anchor nodes, and **cliques** of completely connected nodes. I contribute design guidelines for motif glyphs; example glyphs for the fan, connector, and clique motifs; and algorithms for detecting these motifs. I have also developed a free and open source reference implementation, made publicly available as part of NodeXL [Smi+10].

With case studies and a controlled study I demonstrate the effectiveness of motif simplification as well as areas to focus on for improving glyph design. Motif simplification can result in substantial reductions in visual complexity, allowing easier understanding and manipulation of large network visualizations. There are several avenues for exploration opened up by this work, including additional glyphs for other common motif types, algorithms and glyphs for fuzzy motifs, and methods for showing edge directionality within glyphs. Now that motif simplification is available to all users of NodeXL, my hope is that it becomes commonly used as a first step when dealing with large, complex networks. It is particularly suited for simplifying data collected in an egocentric fashion, such as web spiders and crawls of social media websites.

Chapter 5

Meta-Layouts for Subdividing Networks

5.1 Introduction

Visualizing a network's topology in a node-link visualization can be useful for seeing its overall structure and tracking individual relationships or paths. However, with large, dense networks it can be challenging for a user to understand this structure due to the high number of edges and the resulting visual clutter. The large number of edge crossings and tightly packed nodes in visualizations of these networks can be difficult for the human eye to comprehend, though automated techniques can aid understanding. Various automatic techniques can algorithmically group related nodes together based on (1) the topology of the network [CNM04; WT07; GN02], (2) any attributes the nodes have [Llo82], or (3) some combination of both [Nav+09]. Topologic clustering finds groups of nodes such that the connections within groups (referred to as the intra-group edges) are tighter than those between groups (called the inter-group edges). Another popular method is to group the nodes based on some common attribute such as geographical location or interests,

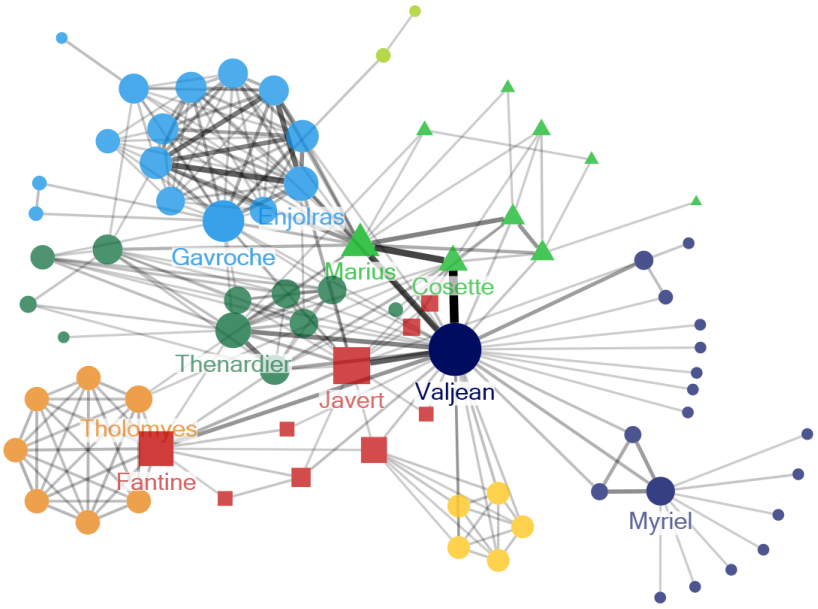


Figure 5.1: Co-appearance network in *Les Misérables*, originally compiled by Knuth [Knu93] and made into an edge list by Newman and Girvan [NG04]. Available in the NodeXL format from nodexl.codeplex.com/wikipage?title=NodeXL%20Teaching%20Resources

or a clustering of several attributes. As the nodes in a community tend to behave similarly or share characteristics, it can be useful to study individual communities.

Regardless of the source of a grouping, a persistent problem is that of displaying the results of a grouping in the network visualization. Displaying the groups using node color or shape alone (like in Fig. 5.1) can be challenging, especially if the groups are intermingled in a complex network visualization (e.g., Fig. 5.32). As the network layout does not take group membership into account when placing nodes, it can cause groups to be occluded within the visualization and loss of information

about the structure of clusters and their relationships [Rod+11]. Meta-nodes can show aggregate relationships, but hide the internal structure of the groups.

One approach showing these groups in the layout is to try to visually separate groups of nodes in the final visualization, such as in the Lin-Log layout [Noa04]. However, it is hard to understand the relationships between groups in these layouts and these visualizations use much more screen space than regular force-directed layouts. Moreover, force-directed layouts in general and these types of group-aware layouts in particular require substantial parameter adjustment to work across a range of datasets [Bar+08]. It can be challenging to balance the various forces acting on nodes, especially as the networks increase in size. Furthermore, as noted by Barsky et al. [Bar+08] when working with immunologists, domain experts using network analysis tools can be completely unwilling to tweak layout parameters in order to obtain the best visualization.

I present several new approaches for showing node groupings using **meta-layouts**, which take take an underlying grouping into account when placing nodes in the node-link visualization. The first, the **Midichlorian-Directed Layout**, is a modified force-directed layout that varies attractive forces between nodes based on group membership. Next, rather than using node-link visualizations and force-directed layouts of network topology alone, I describe several **Group-in-a-Box meta-layouts** that augment topology visualizations with the group memberships

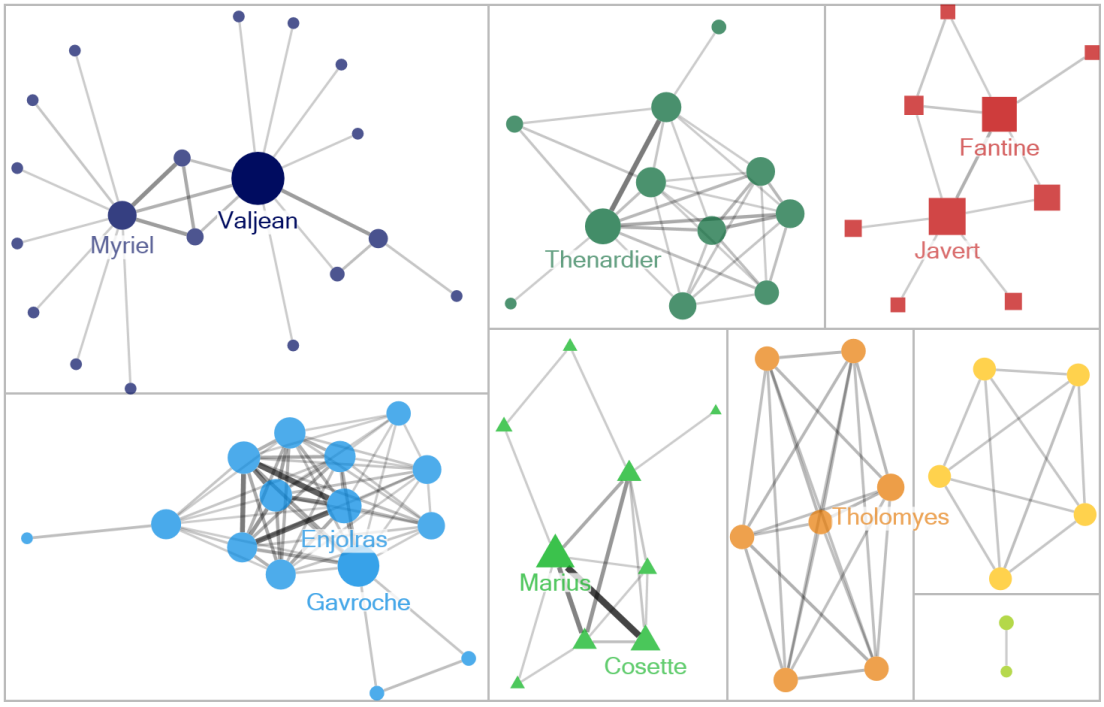


Figure 5.2: Co-appearance network in *Les Misérables* from Fig. 5.1, after using the squarified Treemap Group-in-a-Box layout. Each box shows a cluster found using the Wakita-Tsurumi algorithm [WT07]. Inter-group edges are hidden to better show internal cluster topology. This visualization highlights the structure of each group, such as the Javert & Fantine cluster and the Thenardier cluster.

of the underlying nodes. These Group-in-a-Box layouts draw a separate box for each group, sized according to the number of nodes in the group. The subnetwork the group represents is then laid out within the box, independent of the rest of the network. An example Group-in-a-Box layout for the *Les Misérables* co-appearance network from Fig. 5.1 is shown in Fig. 5.2.

I detail three Group-in-a-Box (GIB) layouts, each with a unique way of laying out the group boxes. First, I describe the **squarified Treemap GIB layout**, created by my colleagues on the NodeXL team [Rod+11]. Next, I move to the two

Croissant-Donut GIB layouts: the **Donut**, which places the most connected group in the center of the visualization and wraps the other group boxes around it in a space-filling manner, and the **Croissant**, which places the most connected group in the top of the visualization and similarly wraps the other group boxes around it. The Croissant-Donut layouts were created in conjunction with three graduate students I mentored for a course project [Cha+13]. Finally, I discuss a **Force-Directed GIB layout** I created which arranges the group boxes based on the aggregate connections between groups. I algorithmically choose which Group-in-a-Box layout to use depending on the disconnected components present in the visualization, number of groups, and distribution of group sizes. I evaluate these Group-in-a-Box layouts through several case studies and an empirical study of 309 of Twitter scrapes, which demonstrates the effectiveness and trade-offs of the various layouts. These Group-in-a-Box layouts have several benefits: (1) they optimize the layout of relationships within groups, (2) they highlight aggregate relationships between groups, and (3) it is easier to see group membership and size. These layouts are publicly available as part of NodeXL [Smi+10].

5.1.1 Chapter Overview

Specifically, the contributions of this chapter are:

- A meta-layout called the Midichlorian-Directed Layout which spreads groups

apart in a standard node-link visualization;

- A Croissant-Donut Group-in-a-Box layout that places subnetworks in boxes arranged using a Donut or Croissant pattern, and balances space-filling properties with showing group relationships;
- A Force-Directed Group-in-a-Box layout that places subnetworks in boxes arranged by their connectivity, and shows group relationships well at the expense of additional screen space;
- A set of automatic choices that are made for the user to better show disconnected components, few groups, or different distributions of group sizes and connectedness;
- Supporting case studies and an experiment on Twitter networks; and
- A free and open source implementation as part of NodeXL.

Parts of this chapter have been published in an overview paper on novel network analysis techniques in NodeXL [SD12] or are under submission [Cha+13]. I first discuss various automatic techniques for grouping the nodes in the network that I will be able to leverage in my meta-layouts (Section 5.2). Next, I cover my preliminary work on Midichlorian-Directed Layouts in Section 5.3, then move on to the three Group-in-a-Box layouts in Section 5.4. I then describe evaluations of the Group-in-a-Box approach using case studies (Section 5.5) and an experimental study on 309 Twitter scrapes (Section 5.6). I end by summarizing in Section 5.7.

5.2 Grouping Techniques

Before my meta-layouts can be applied, we first have to create meaningful groupings of the nodes in the network. Various automatic techniques can algorithmically group related nodes together based on (1) the topology of the network, (2) any attributes the nodes have, or (3) some combination of both. The choice of which technique to use for grouping the nodes depends on the target analysis task.

5.2.1 Clustering to Identify Structural Components

Understanding the complexity of human anatomy is often facilitated by decomposing into subsystems such as circulatory, muscular, skeleton, neural, digestive, etc. These decompositions favor functional structures over physical adjacency. Since networks represent complex phenomena, clustering by connectivity into functional subsystems often proves to be beneficial. An example of this **topologic clustering** is shown in Fig. 5.1, which displays the network of characters in *Les Misérables*. This co-appearance network shows the relatedness among characters. Edge thickness shows the number of scenes in which pairs of characters appear, while node size shows the number of scenes for each character. Nodes are colored based on their automatically detected topologic clusters. Clustering is often used as an exploratory data analysis method to discover unexpected inclusions within a known cluster, unexpected separation into other clusters, or surprising clusters.

There are many topology-based clustering techniques, usually directed at finding groups of nodes that are more tightly connected with each other than with nodes outside the group. NodeXL implements the Clauset-Newman-Moore [CNM04], Wakita-Tsurumi [WT07], and Girvan-Newman [GN02] clustering algorithms, which all result in mutually exclusive cluster membership. The NodeXL implementations currently work only on undirected graphs, but additions to support directed and weighted graphs are planned. The effectiveness of such clusterings can be determined using metrics such as **modularity** [NG04], which is roughly the number of edges within groups minus the expected number in an equivalent random network. However, verifying the quality of a clustering outcome is often hampered by the lack of a ground truth.

5.2.2 Grouping to Find Attribute Relationships

Instead of highlighting individual structural features like topologic clustering, attribute aggregation can display overall topology and attribute patterns. Nodes may represent people, places, documents, or roles, which are readily understandable in small networks. However, with thousands or millions of nodes, analysts may gain insights by replacing nodes of a common type with a single group node, e.g. author nodes in a scientific citation network might be grouped by their current institution into a single node for each institution. This node could be sized by

the number of authors, thereby showing the productive institutions and revealing the degree of collaboration across institutions. Simplifying a million-node author network into a 3000 node institution network removes information, but reveals important patterns.

Attribute-based node aggregation has been leveraged by several tools to understand overall relationships at the expense of showing the underlying topology explicitly. PivotGraph [Wat06] groups nodes based on the intersection of a pair of attributes, and arranges the meta-node for each group on a grid with each attribute as an axis. Aggregate links between groups are shown with arcs. Similarly, my GraphTrail (Section 3.3.2) groups nodes by attribute into standard charts, where the groups can be further filtered, merged, or used to pivot to connected groups of other node types. One advantage of this aggregation is a dramatic reduction in screen space required, a fact leveraged by GraphTrail to show the exploration history directly integrated into the network analysis canvas. Identical value grouping can be used to show the relationships between semantic groups as well as the relationships within them, for example with semantic substrates [SA06].

NodeXL allows grouping nodes into **meta-nodes** by their attributes. As an example, Fig. 5.3 shows U.S. Senate co-voting patterns. Nodes are colored by the party affiliation attribute: red for Republicans, blue for Democrats, and orange for independents. Fig. 5.4 shows the same network with senators grouped by their

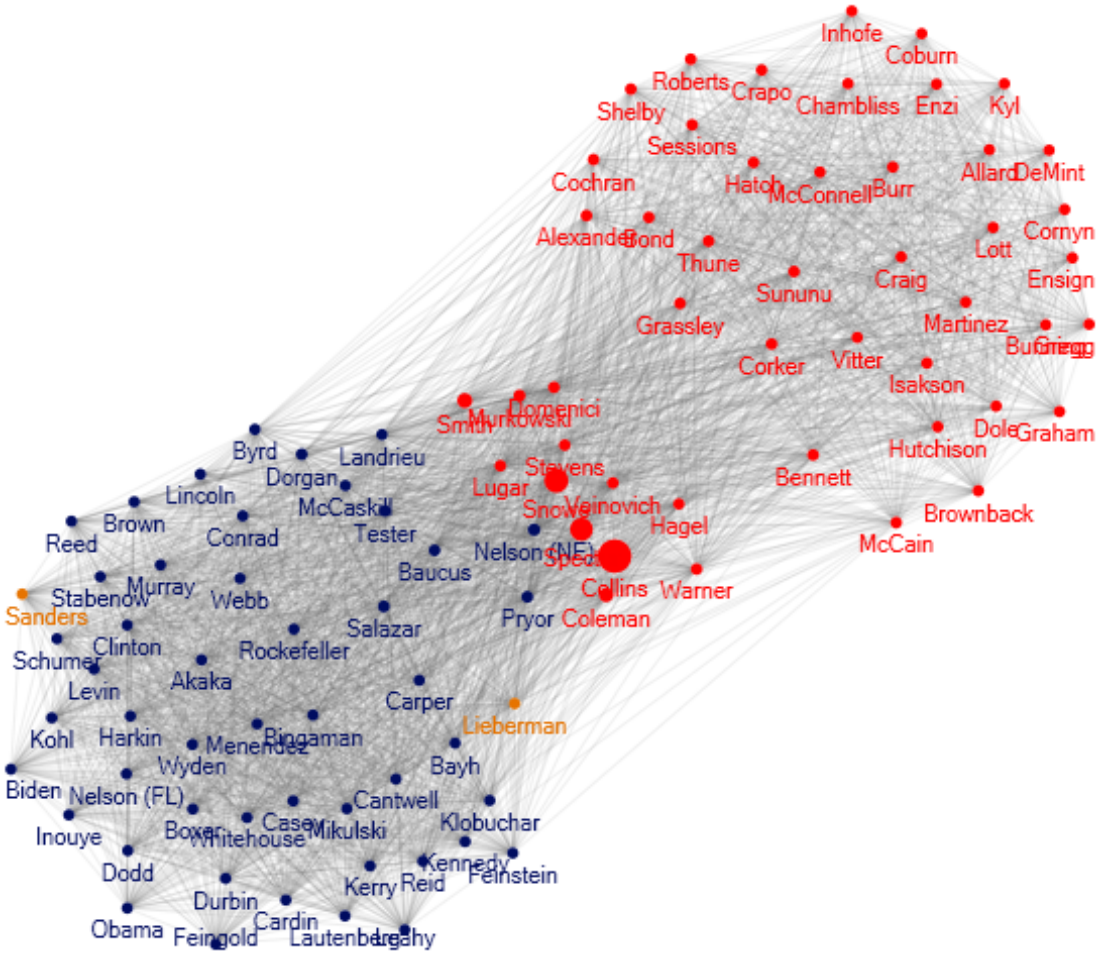


Figure 5.3: The U.S. Senate co-voting network for 2007 in shown here, with nodes for individual senators colored by their parties (blue Democrats, red Republicans, orange Independents), sized by betweenness centrality, and laid out using Furuchterman-Reingold [FR91]. Edges tie senators together and are weighted by their percent of voting agreement. Only those edges with at least 50% agreement are shown.

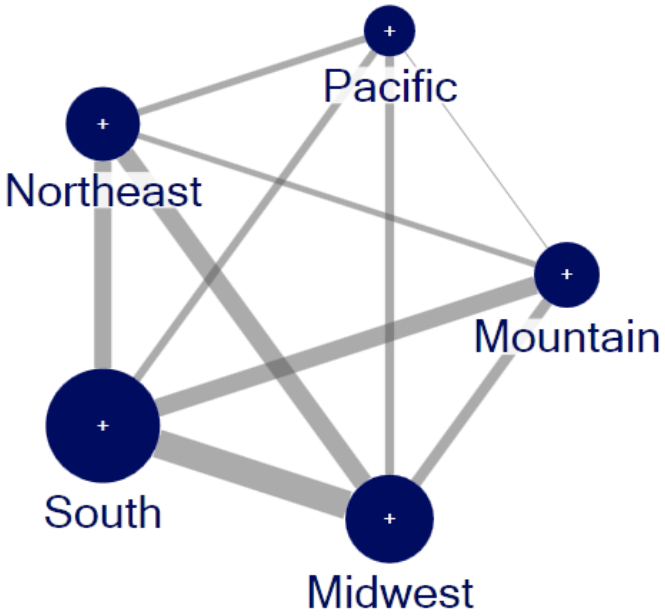


Figure 5.4: 2007 U.S. Senators grouped by their regional affiliation into meta-nodes. Aggregate meta-edges show the number of senators between the two groups that vote the same way on bills at least 50% of the time. Collapsed from the network in Fig. 5.3.

regional affiliations into meta-nodes. Grouping multiple nodes into a single meta-node can produce measurable improvements in readability.

5.2.3 Advanced and Combined Approaches

Additional ways to group nodes by their attributes include the ubiquitous k-means clustering algorithm [Llo82], which can be used to cluster nodes by similar attribute values or sets of attribute values. This provides ways to create "fuzzy" groups with related, but not identical, node attributes. Another approach called VI-Cut [Nav+09] combines hierarchical clustering with topologic clustering. Navlakha et

al. use node attributes to create attribute-driven cuts of a hierarchical topology clustering, specifically focusing on biologic networks and predicting operational taxonomic units based on hierarchy of sequences and annotations. NodeXL does not currently support non-exact attribute clustering, but the results of these algorithms can be easily copied into the groups worksheets.

5.3 Midichlorian-Directed Layout

The first meta-layout I developed is a modified force-directed layout that takes group membership into account when computing forces between nodes, reducing the spring forces between nodes in separate groups. This approach is called the **Midichlorian-Directed Layout (MDL)**, in reference to how individuals in the fictional Star Wars universe have varying levels of Force sensitivity depending on their midichlorian count. To paraphrase Darth Vader, "The Force is strong with this [cluster]." This approach was developed in conjunction with Darya Filippova.[1]

Our motivation for creating such a layout is that our current techniques for showing group membership, like displaying convex hulls on a node-link visualization, can be challenging to interpret. Figs. 5.5 and 5.6 show how challenging this can be with even simplified biologic networks. The network in these images is the the human protein interaction network obtained from the HPRD database, simpli-

[1] http://www.cs.cmu.edu/~dfilippo/

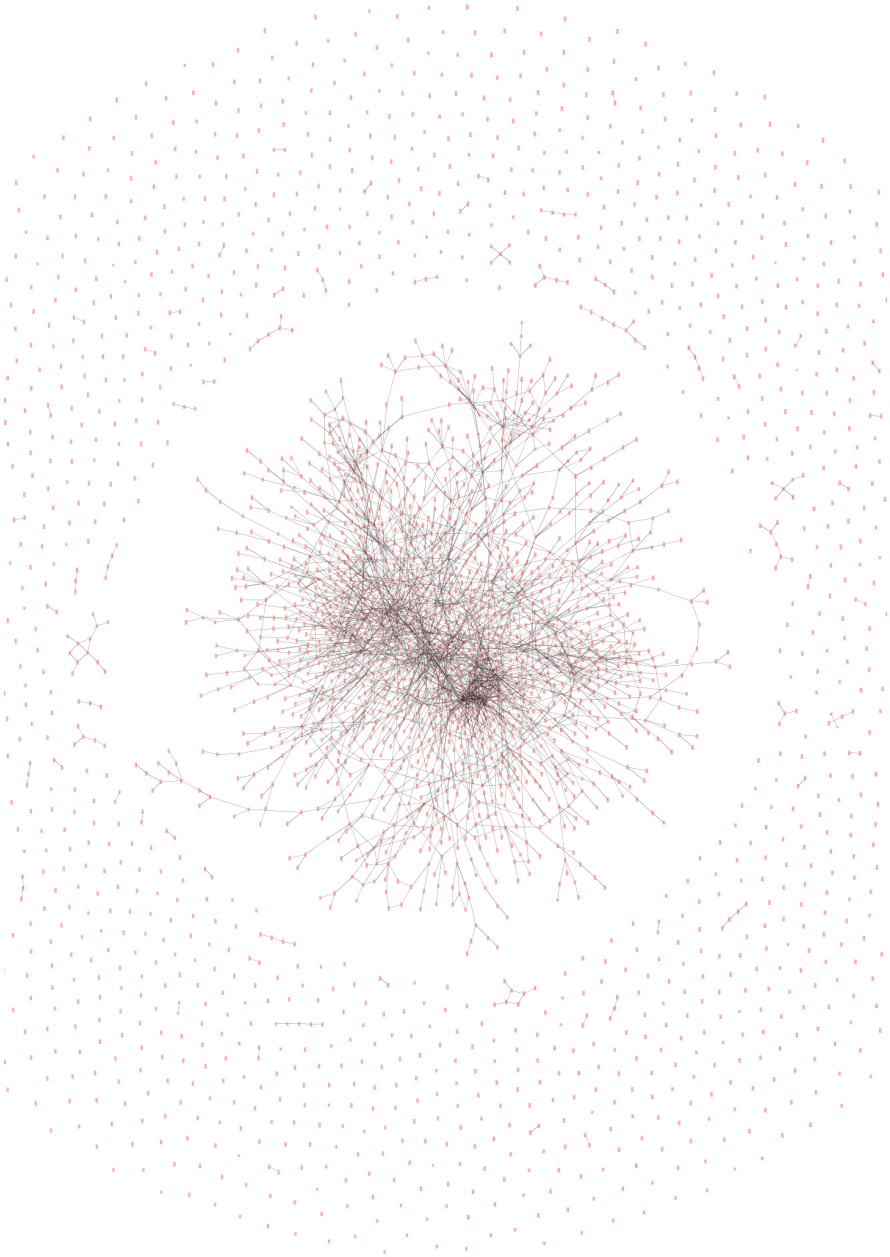


Figure 5.5: Graph summarization of the human protein interaction network from the HPRD database drawn with the Prefuse Force-Directed Layout with a global anti-gravity coefficient of 9×10^{-6}.

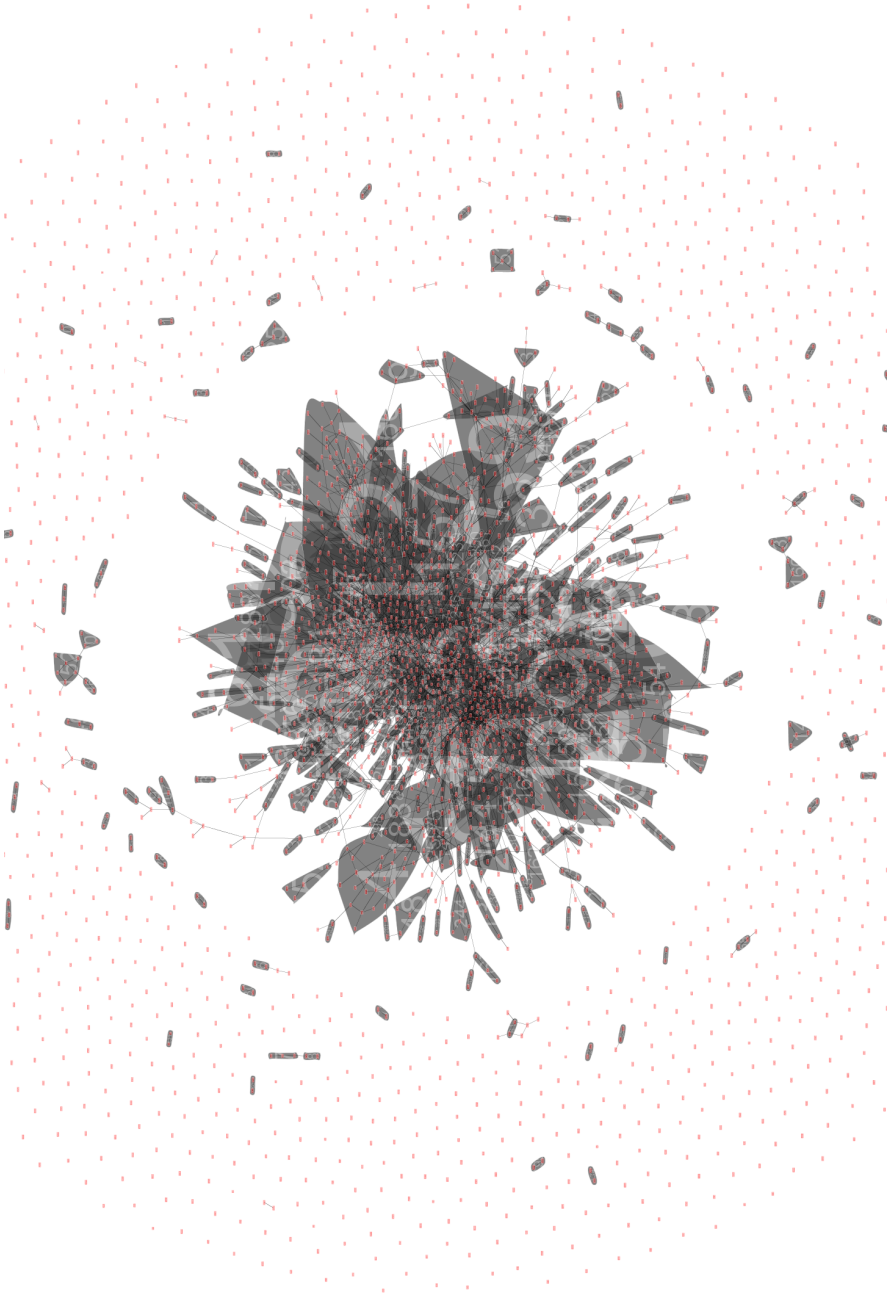


Figure 5.6: Same summarized human protein interaction network as Fig. 5.5, but clustered using Newman's heuristic with convex hulls surrounding each cluster.

fied using graph summarization [NRS08] down to 3312 nodes and 4746 edges. The groups shown with convex hulls in Fig. 5.6 are computed using Newman's fast community finding heuristic [New04]. Notice how the substantial occlusion among the clusters prevents getting an accurate cluster count and limits the viewer's ability to see relationships between them.

We based our approach for showing group membership more clearly on the interactive Force-Directed Layout provided by Prefuse [HCL05], which is a physics simulation with three main forces:

1. Nodes exert anti-gravity on each other to enforce spacing following an inverse square law. This is computed using the efficient $O(n \log(n))$ time approximation of the gravitational n-body problem of [BH86] which uses a quad tree to find accurate local interactions while aggregating body masses.

2. Edges are modeled by springs that pull connected nodes together with globally constant spring coefficients and length.

3. Drag forces for nodes similar to air resistance are used to prevent oscillations.

At each timestep the forces are updated and the new node position and velocity is calculated by integrating over the timestep with the 4th-order Runge-Kutta method or, optionally, the faster but less accurate Euler Forward Method. Both of these integration techniques are described in [Pre+93].

Algorithm 5 Force-directed layout algorithm

```
addSpringForces()
addRepulsiveForces()
for every node u do
    for every node v do
        calculateForce()
        integrateForce() //Runge-Kutta integration
        assignPosition(v)
```

Algorithm 6 addSpringForces() function used in MDL

```
for every node u do
    for every node v do
        if u,v are connected then
            if u,v in the same community then
                a = sharedNeighbors(u,v)
                k = a * 10; //tighten the spring
            else
                k = a / 10; //relax the spring
```

Our modified algorithm was inspired by the Vizster layout algorithm [HB05]. In Vizster, the edge spring coefficient between the adjacent nodes varied based on the minimum degree of the two nodes incident on that edge (see Algorithm 5). This way nodes with few neighbors are drawn closely together while nodes with many neighbors are spaced farther apart to improve readability. We preserved this behavior, but with smaller weight on the node degree since our primary goal was to highlight group membership.

The Vizster algorithm [HB05] did not explicitly use the community structure which resulted in overlapping communities, as seen in Fig. 5.7. Instead of letting the minimum node degree derive the community structure in the network,

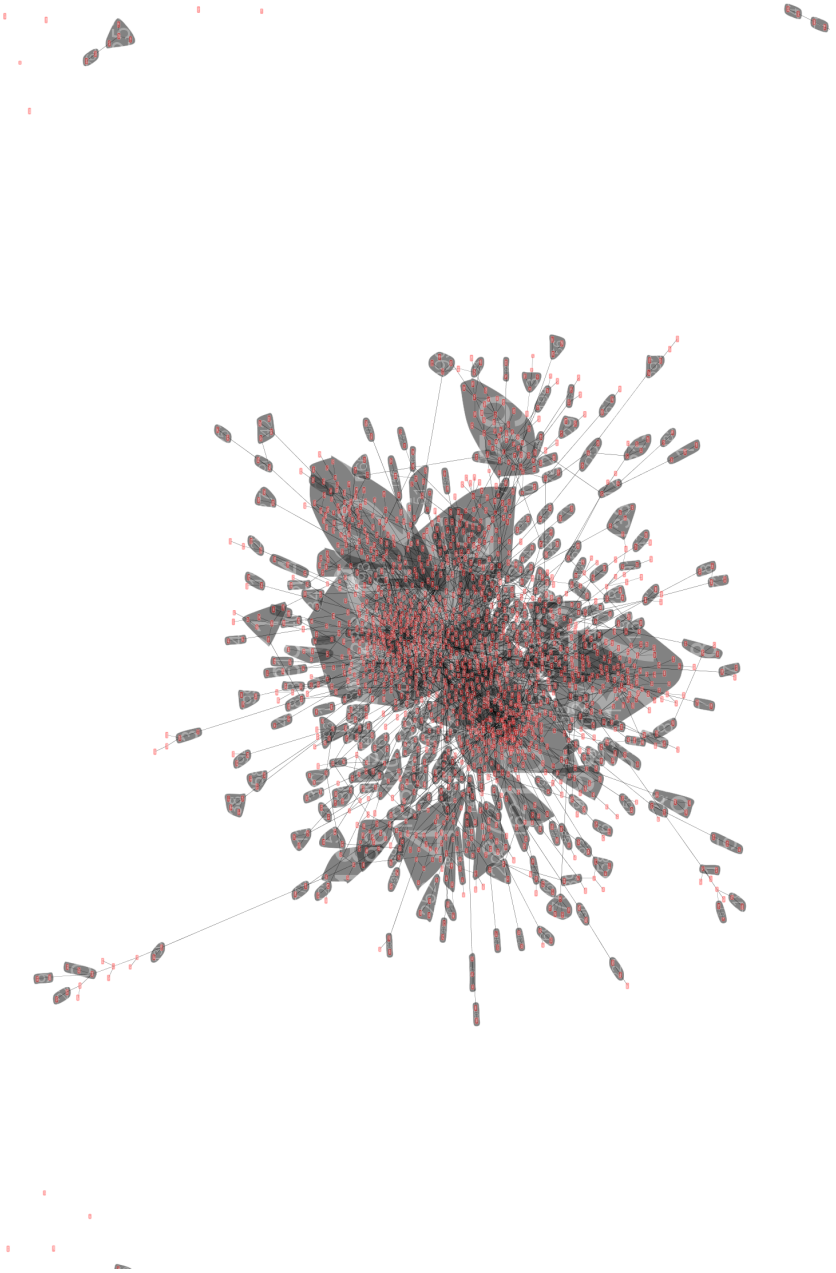


Figure 5.7: Same summarized, clustered human protein interaction network as Fig. 5.6, but using a global anti-gravity coefficient of 9×10^{-5} and zoomed in on the main connected component. Clusters are separated somewhat using the Vizster meta-layout modification to the Prefuse force-directed layout, resulting in less cluster overlap.

we decided to exploit the community information as produced by the Newman's heuristic [HB05]. For each pair of nodes that shared a cluster, we increased the spring coefficient to bring the nodes together. Likewise, for each pair of nodes in different clusters we decreased the spring coefficient and let the nodes drift apart. We wanted to control how close the nodes within the cluster got so we made the edge spring coefficient proportional to the number of neighbors shared by the u,v. This addition decreases convergence time and brings the densely interconnected nodes together. Our algorithm is shown in Algorithm 6.

NodeXL was still in early development when this work was conducted. Instead, we implemented the Midichlorian-Directed Layout in SocialAction [PS06; PS08a; PS08b]. We chose SocialAction because of it's ability to handle online, interactive, and animated layouts through its use of the Prefuse toolkit [HCL05]. In order to easily compare the effects of various force-directed layouts, we wanted to be able to dynamically change layouts and layout parameters while preserving the **mental map** [Mis+95] users had of the network. Preserving this mental map is important so users can understand changes to the network [MB04; PHG07]. We implemented a GUI to enable these interactive layout algorithm switches and animating between them.

Fig. 5.7 shows the result of applying the Vizster SocNet layout [HB05], which provides some spacing between clusters but not enough for everything to be read-

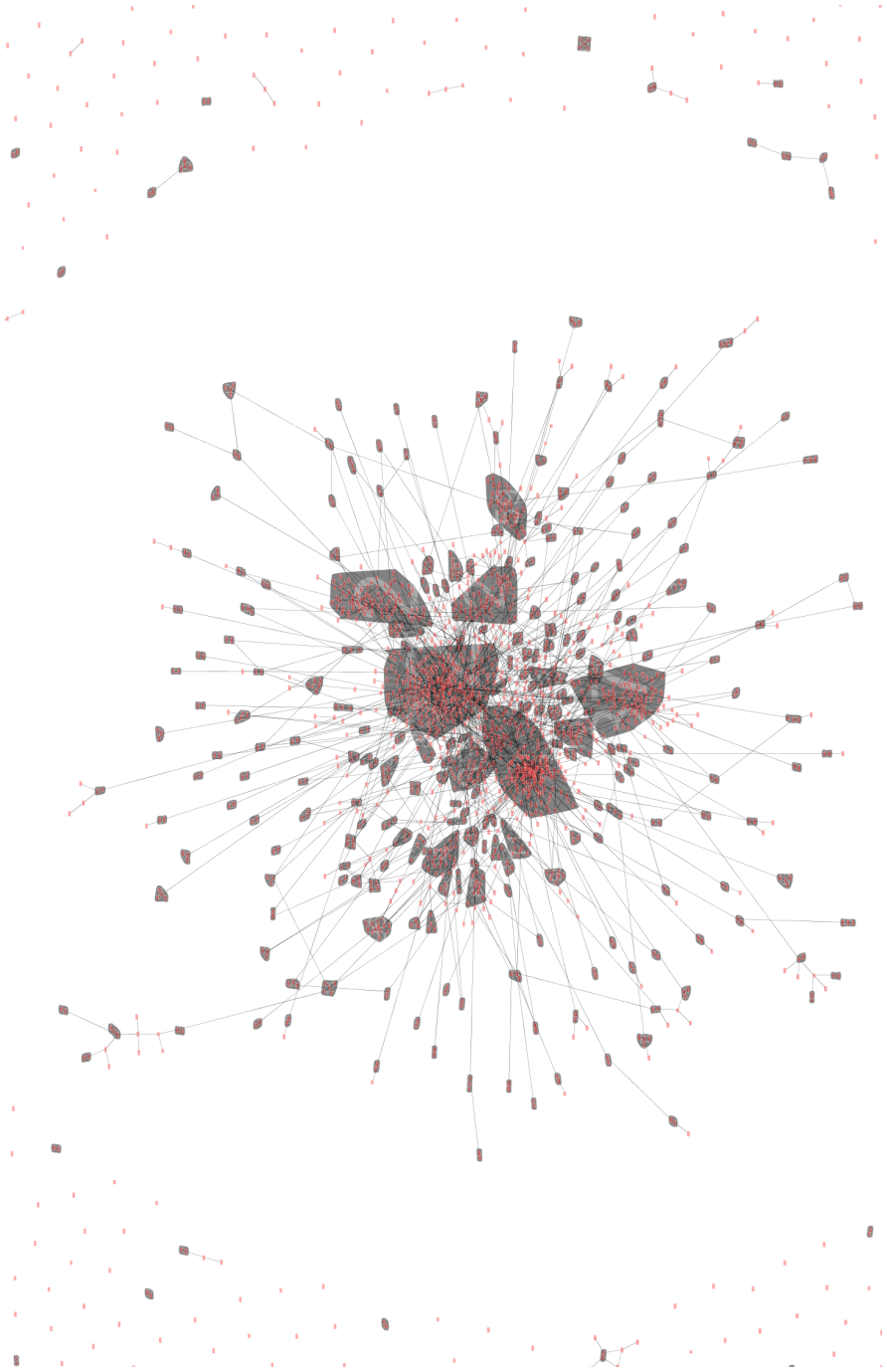


Figure 5.8: Same summarized, clustered human protein interaction network as Fig. 5.7, with clusters separated further using the Midichlorian-Directed Layout. The internal structure of these clusters is more visible, as well as the inter-cluster relationships.

able. Contrast this with MDL in Fig. 5.8, which has almost no cluster occlusion and in which the connections between clusters are clearly visible. Moreover, using the GUI to switch layouts and the animated group separation allows users to see the effect of the grouping immediately, supplementing any convex hulls or color/shape coding.

However, the MDL approach leaves a lot to be desired. It is still challenging to see the aggregate relationships between groups and their relative sizes. Moreover, the large screen space required for laying out the groups separately severely limits how much detail can be seen.

5.4 Group-in-a-Box Meta-Layouts

Modified layout algorithms are not enough to help analysts see groups or clusters in the network clearly. Instead, we on the NodeXL team have chosen to show each group individually in its own region of the screen, bounded by a box sized according to the number of nodes it contains, and laid out on its own. These Group-in-a-Box layouts reveal internal group relationships, make clear which nodes are part of which groups and how many nodes a group contains, and with effective positioning of the boxes can show aggregate relationships between groups.

I discuss three Group-in-a-Box (GIB) approaches that have different trade-offs in how space-filling they are and how well they show relationships between

groups. The first is the **Treemap GIB Layout**, which completely fills the screen space with roughly square boxes. Next, I detail the **Croissant-Donut GIB layout**, which comes in two variants for networks with varying group characteristics: the **Donut** and the **Croissant**. The Croissant-Donut layouts use most of the screen space, while showing group relationships more clearly than the Treemap layout. Finally, present a **Force-Directed GIB layout** that arranges boxes by the group relationships, highlighting the group connectivity at the expense of additional screen space. These layouts are implemented in NodeXL [Smi+10] and can be selected by the user depending on the network they are trying to analyze and their visualization goals. Moreover, I automatically select the most effective layout for the user based on a set of criteria about the network and group structure.

5.4.1 Treemap Layout

The Treemap Group-in-a-Box layout [Rod+11] subdivides the available screen space using a treemap [Shn92; JS91]. Shneiderman [Shn92] employed a **slice and dice** method for representing hierarchical information in a space-filling manner, which could result in boxes with high aspect ratios. Instead, the NodeXL team uses the **squarified treemap** approach of [BHJVW00] which maintains a low aspect ratio for the boxes. The boxes created by the Treemap layout have area proportional to the number of nodes they contain. For our purposes, it is impor-

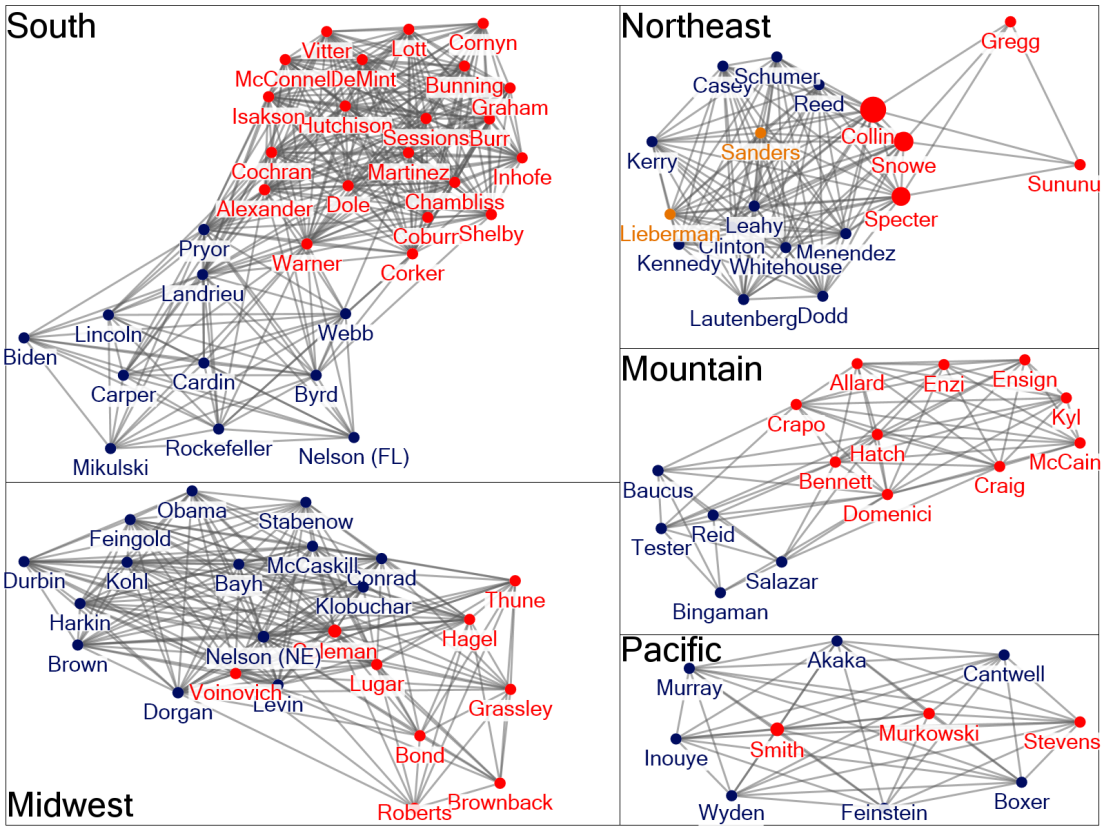


Figure 5.9: 2007 U.S. Senators grouped by their regional affiliation. From [Rod+11]. See Section 4.3.1 for more on this dataset.

tant to keep a low aspect ratio because narrow boxes would not be as effective at displaying the structure of the group laid out inside it, in addition to being difficult to see their area and thus understand the number of nodes they contain. While NodeXL does not currently support hierarchical grouping, the Treemap GIB layout could be easily extended to visualize nested clusters.

Fig. 5.9 provides an example of this approach, where the 2007 U.S. Senate co-voting network is segmented into five geographic regions the senators represent (an attribute-based grouping). See Section 4.3.1 for more on this dataset. The

cross-region edges are hidden in this example. We can see that the South region has the most Senators, while the Pacific is the fewest. Moreover, we can clearly see the internal structure of the groups, such as the general division of each region into the Democrats on one side and the Republicans on the other. In each region, we can also see any moderates that bridge the parties, such as Collins, Snowe, and Specter in the Northeast region.

However, the use of a Treemap layout for the boxes can end up placing highly connected clusters in different regions of the screen, with any connecting edges being drawn across the intermediate clusters (see Section 5.5 for examples). It can be hard to discern whether these long edges connect to nodes in the intermediate clusters, or are merely drawn overlapping. This ambiguity, in addition to the added clutter of these long ties, makes it difficult to analyze the relationships between clusters and draw meaningful conclusion. Our other layouts take the aggregate group relationships into account when determining where to place the boxes, so as to alleviate this issue.

5.4.2 Croissant-Donut Layout

The Croissant-Donut Group-in-a-Box layout [Cha+13] my students and I developed tries to balance the space-filling attributes of the Treemap GIB layout with showing more of the underlying relationships between groups. The layout takes the

overall group relationships into account when choosing where to place the group boxes. We came up with two complementary approaches, the **Donut** and the **Croissant**, each targeted at representing specific types of networks. We choose between these two approaches automatically depending on network and group properties (see Section 5.4.5 for more details). For these algorithms, we will be making use of a per-group metric we call **connectedness**, defined as the number of other groups a group is connected to.

5.4.2.1 Donut Layout

The Donut layout begins by placing the most connected group in the center of the screen. The other groups can be arranged either by their size to reduce wasted space or by their aggregate connections to other groups, so highly connected groups are adjacent. In this discussion I will use the latter, where groups are wrapped around the periphery in decreasing order of their connectedness.

The area occupied by each group is calculated as:

`area(Group)=alpha*screen_area*|Group.nodes|/|Graph.nodes|`

where `Group.nodes` refers to the nodes in the group, `Graph.nodes` refers to the nodes in the entire network, and `alpha` is the initial space-filling factor, which starts at `alpha` = 1.0. Note in Fig. 5.10 the area of each group, proportional to the number of nodes it contains, does not necessarily decrease as connectedness decreases.

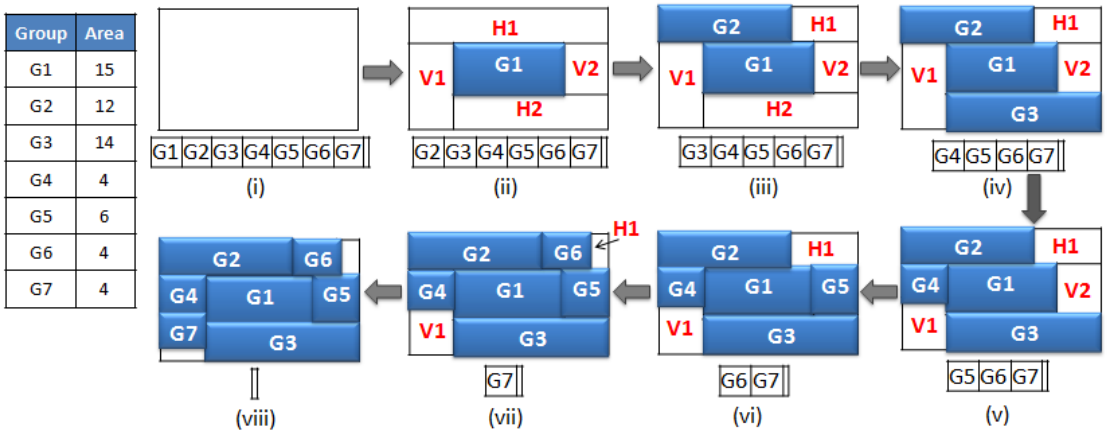


Figure 5.10: The basic principle behind the Donut variant of the Croissant-Donut layout is to place the most connected group in the center of the screen, then placing the other groups around its perimeter based on their connectedness (number of other groups they are connected to).

This process for the Donut GIB layout is illustrated in Fig. 5.10 for a network with 7 groups, listed on the left in decreasing order of connectedness. The steps are marked in sequential order as Steps (i) to (viii). The figure also shows which groups remain to be placed at the bottom of each step. Initially in Step (i), we place G1 the most connected group at the center with an aspect ratio proportional to that of the screen. This is represented as a blue box in Step (ii), also known as the "donut hole". Placing G1 divides the screen into two horizontal (H1 and H2) and two vertical (V1 and V2) empty boxes. The remaining groups will be arranged in these boxes alternating in the sequence H1, H2, V1, V2.

Since we only know the areas and not the dimensions of the groups, we use the orientation of the empty boxes to determine the group's width or height. While placing a group in a horizontal empty box, we set its height to be same as that of

the horizontal empty box. Its width is then determined by dividing its area by the height. Using the dimensions calculated above, the group is finally placed in the horizontal empty space box aligned with the empty box's left side. For example, in Fig. 5.10, Step (ii) has a horizontal empty box labeled H1. The result of placing a group, G2, in H1 is shown in Step (iii) creating a new smaller H1 in (iii). It is easy to see that G2 and H1 of Step (ii) have same heights and they have a common left edge. Similarly, while placing a group in a vertical free box, we set its width to be same as that of the empty box and its height is determined using its width and area. The group and the vertical empty box share the top edges of the empty box. See placement of G4 in V1 in Steps (iv) and (v) of Fig. 5.10 for an example.

After placing G1, we place the next group, G2, in H1; followed by G3 in H2, G4 in V1 and G5 in V2 (Steps (ii) to (vi)). Step (vi) shows that we have G6 and G7 left. So starting again at H1, we place G6 in H1 (Step(vii)). We then try to place G7 in H2, but H2 had no space left. So, we move on to V1 and place G7 there (Step (viii)). No groups remain to be placed at Step (viii).

The method proposed above is not space filling which might result in a situation where the algorithm still has some groups to place on the screen but none of the empty boxes are big enough. In such a situation, we restart the algorithm with `alpha` = 0.9∗`previous_alpha` and repeat this process until all the groups get placed on the screen.

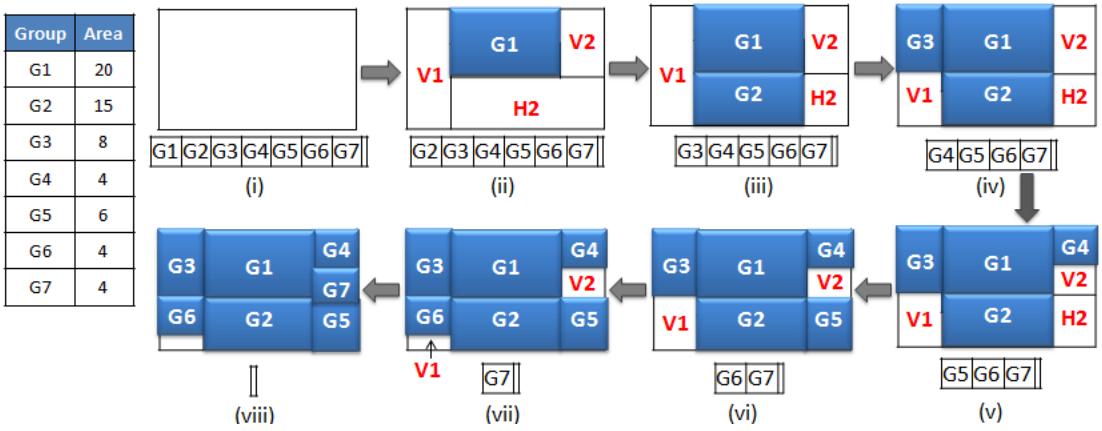


Figure 5.11: The basic principle behind the Croissant variant of the Croissant-Donut layout is to place the most connected group in the top of the screen, then place the other groups around the other three sides based on their connectedness (number of other groups they are connected to).

5.4.2.2 Croissant Layout

As in the Donut layout, the Croissant layout sorts groups in decreasing order of connectedness and computes initial areas that can be decreased iteratively if needed. The box placement is similar, but instead of placing the most connected group at the center of the visualization, it is positioned at the top forming the "croissant hole" shown in Fig. 5.11. The rest of the groups are placed around the remaining three sides, in one horizontal and two vertical empty boxes instead of two each, namely H2, V1 and V2 (Step (ii)). Groups are placed around G1 alternating in the sequence H2, V1 and V2 (Steps (iii) to (viii)).

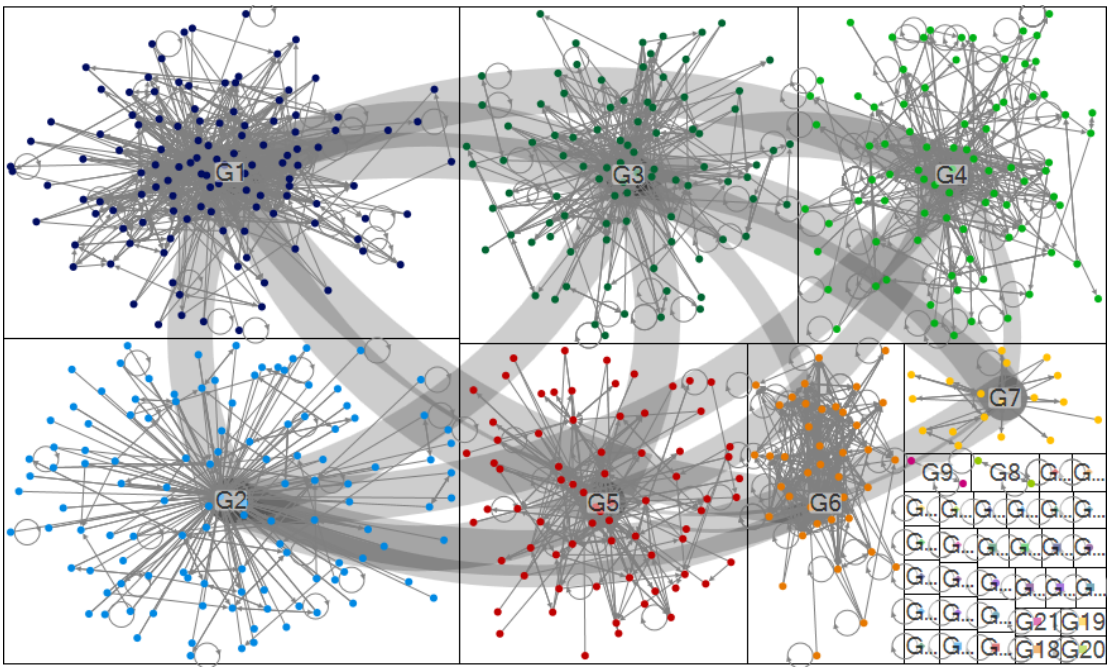


Figure 5.12: A Donut-favoring network & groups, shown in the Treemap layout.

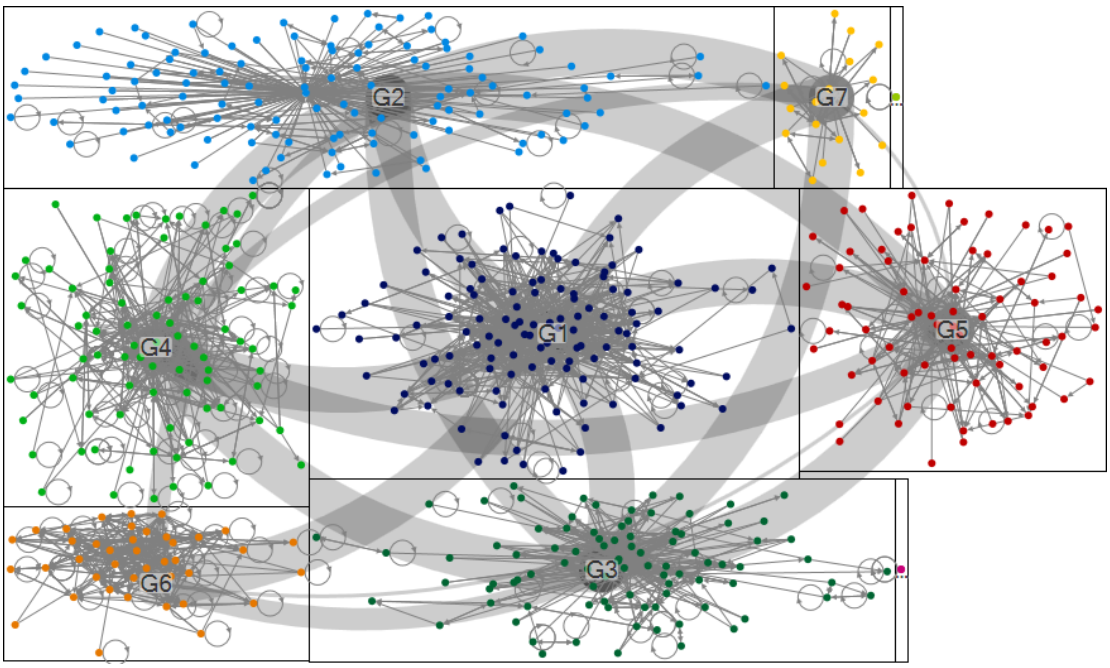


Figure 5.13: A Donut-favoring network & groups, shown in the Donut layout.

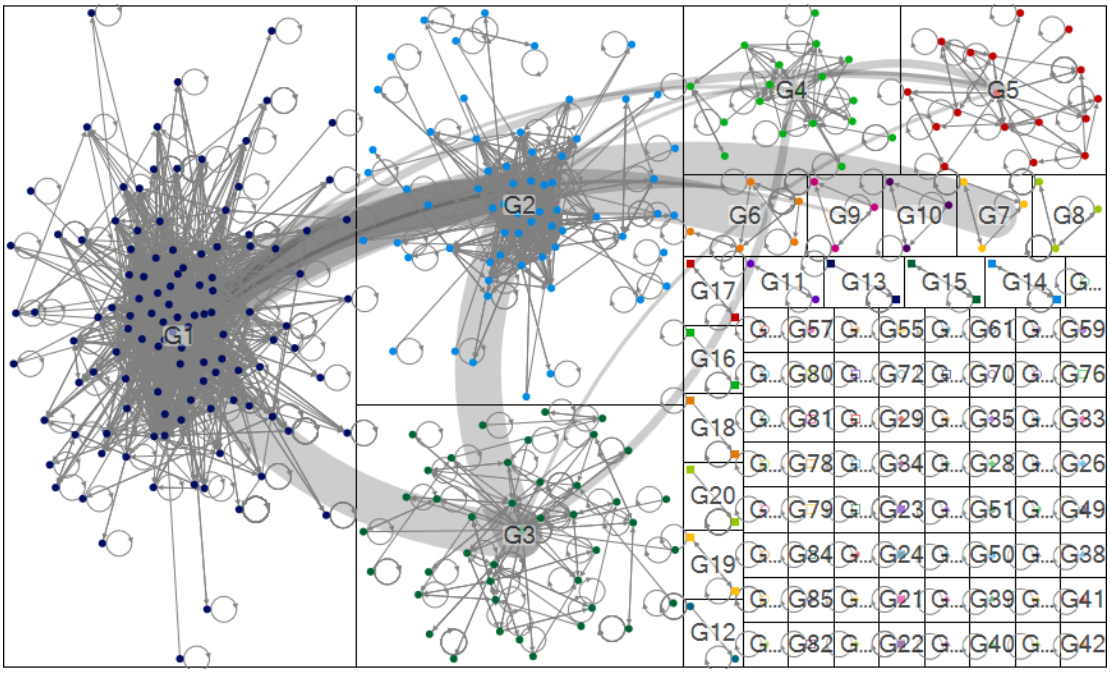


Figure 5.14: A Croissant-favoring network & groups, shown in the Treemap layout.

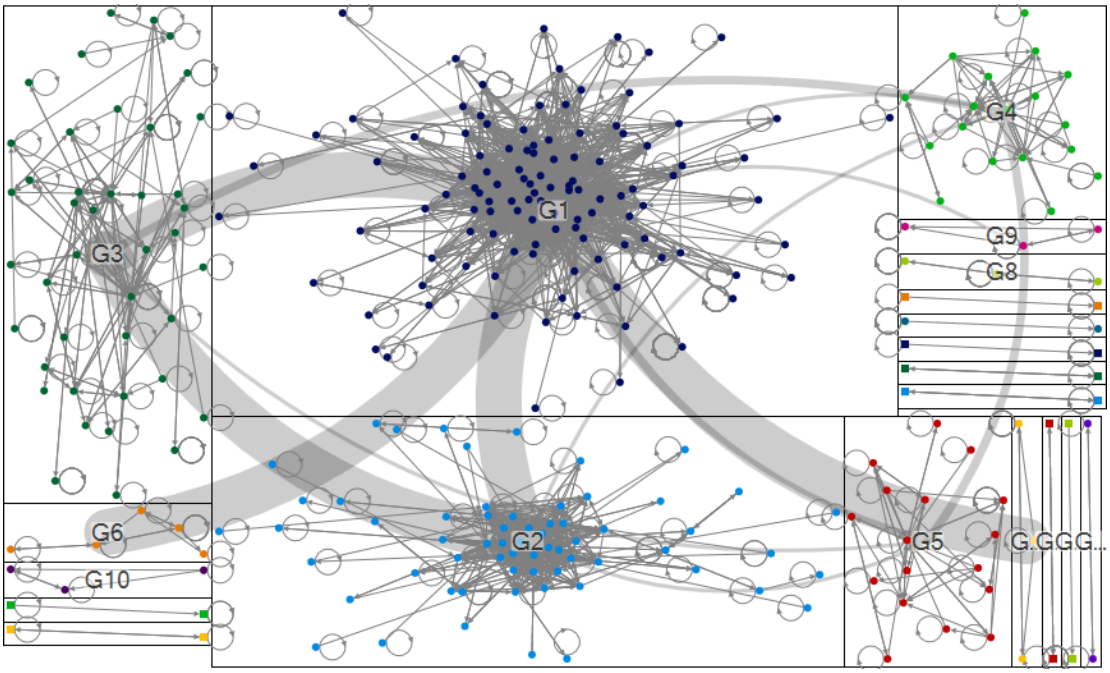


Figure 5.15: A Croissant-favoring network & groups, shown in the Croissant layout.

5.4.2.3 Comparing the Donut and Croissant Variants

In general, the Donut variant of the Croissant-Donut layout is more effective when there are lots of small groups in the network. However, when the network contains one or two big clusters and a few small clusters, it can result in a lot of wasted space. In those cases, the Croissant layout performs better. We choose between these two approaches automatically depending on network and group properties (see Section 5.4.5 for more details).

Figs. 5.12 to 5.15 illustrate these approaches for two separate networks in comparison with the Treemap GIB layout, using combined meta-edges in place of the original inter-group edges. One caveat here is that many of the smallest groups have been filtered out in the Croissant-Donut versions as they do not show large numbers of small groups effectively. The first two figures show a network that is more suitable for the Donut layout, originally in a Treemap (Fig. 5.12) and then the same network in the Donut layout (Fig. 5.13). The meta-edges connecting groups in the Treemap layout suffer from an abundance of overlaps and crossings, especially near highly-connected groups like G3 and G5. However, the Donut layout positions these groups so that there are no crossings near the center of groups, with the remaining crossings around the group edges much easier to follow. The next two figures show a network that is more appropriate for the Croissant layout, first in a Treemap (Fig. 5.14) then in the Croissant layout (Fig. 5.15). Again, we

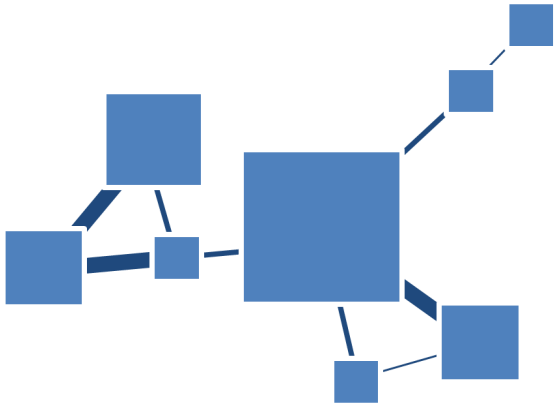


Figure 5.16: The Force-Directed GIB layout explicitly positions groups based on their aggregate connections, showing group relationships clearly at the expense of additional screen space.

see excessive overlap and crossings near the center of G2 that limit readability, while the Croissant layout version of almost completely eliminates the problem. While this approach does well at balancing a space-filling layout with showing the ties between groups, my Force-Directed Layout (described in the next section) chooses to use more white space to show those ties more clearly. These trade-offs are empirically verified in Section 5.6.

5.4.3 Force-Directed Layout

The Force-Directed Group-in-a-Box layout is my approach for explicitly showing the inter-group relationships in the visualization. The boxes are positioned using a standard force-directed layout run on the aggregate network, where the nodes represent entire groups and the edges between them represent the aggregate connections between a pair of groups. The overall concept is illustrated in Fig. 5.16.

I then draw the group boxes on this initial layout centered at the group node's position, followed by a step to remove the overlap created by all these boxes. This layout has the benefit of clearly showing the aggregate topology, but at the cost of more wasted screen space. However, this problem can be reduced by using effective overlap-reduction techniques that minimize the additional screen space required.

5.4.3.1 Initial Configuration

The first step of the force-directed group-in-a-box layout is setting how much of the screen space to use to show groups initially. Groups are represented using squares, sized according to the number of nodes they contain. My experiments with setting the initial space-filling factor ranging from 20% to 100% point to a general trade-off between how space-filling the resulting visualization is and how well the final group positions represent the actual group relationships. For more details on this trade-off, see Section 5.4.3.3.

5.4.3.2 Generate Initial Group Box Positions

The first task is to position the groups according to their connectivity with each other. I create a new network showing the group relationships, with one meta-node for each group and a combined meta-edge joining connected groups. Then I compute a set of initial node positions using NodeXL's implementation of the Harel-Koren fast multi-scale (FMS) layout [HK02a].

I chose to use the Harel-Koren FMS layout [HK02a] because it was implemented in NodeXL already, was sufficiently fast, and produced good results. However, future implementers may wish to use faster or more effective layout algorithms. According to experimental evaluations of several best-of-breed layout algorithms carried out by Hachul and Jünger [HJ06; HJ07], two good choices would be the high-dimensional embedding (HDE) approach of Harel and Koren [HK02c] or the algebraic multigrid method (ACE) of Koren, Carmel, and Harel [KCH03]. Hachul and Jünger report that HDE, followed by ACE, was the fastest algorithm for many test cases. However, if these layouts produce ineffective visualizations Hachul and Jünger suggest using their FM^3 algorithm [HJ05; Hac05] to get comparable or better results while still having a reasonable run time. The FM^3 layout may be complex to implement, and was the focus of Stefan Hachul's dissertation [Hac05].

My current implementation using the Harel-Koren FMS layout [HK02a] does not use meta-edge weight when calculating the layout. While this technique has proven effective in the examples I have explored, I would suggest that future implementers use this meta-edge weight as part of the layout algorithm to pull more strongly connected groups closer together and separate poorly connected groups further. The HDE algorithm [HK02c] can be modified for visualizing weighted networks [Hac05, p.22], and both ACE [KCH03] and FM^3 [HJ05; Hac05] can be applied to weighted networks.

Another issue is that NodeXL's implementation of the Harel-Koren FMS layout [HK02a] does not presently handle multiple disconnected components well, and lays them out individually in the same regions. Thus, care should be taken to ensure to apply the force-directed Group-in-a-Box layout only on networks without disconnected components or isolates. See Section 5.4.5 for a general solution to the disconnected component problem.

5.4.3.3 Remove Group Box Overlap

After the initial group box positions are determined, I have to contend with the fact that the layout algorithm is unaware of the group boxes. If I draw each box centered at its position from the layout algorithm I get a visualization like Fig. 5.17 with substantial amount of box overlap. This initial overlap can be reduced by using smaller boxes, but at a significant cost in wasted space. Instead, I try to eliminate the overlaps while retaining as much of the structural information from the layout as possible and minimizing additional area required. Naturally, worse overlap in the initial visualization leads to a less effective resulting layout.

While the problem of creating a minimum-area layout adjustment is NP-complete, there are several effective node-node overlap removal algorithms that can be applied to these group boxes. I use the PRoxImity Stress Model (PRISM) algorithm of Gansner and Hu [GH09], which iteratively computes box overlap along the edges of a Delaunay triangulation and adjusts those edge lengths accordingly to remove the

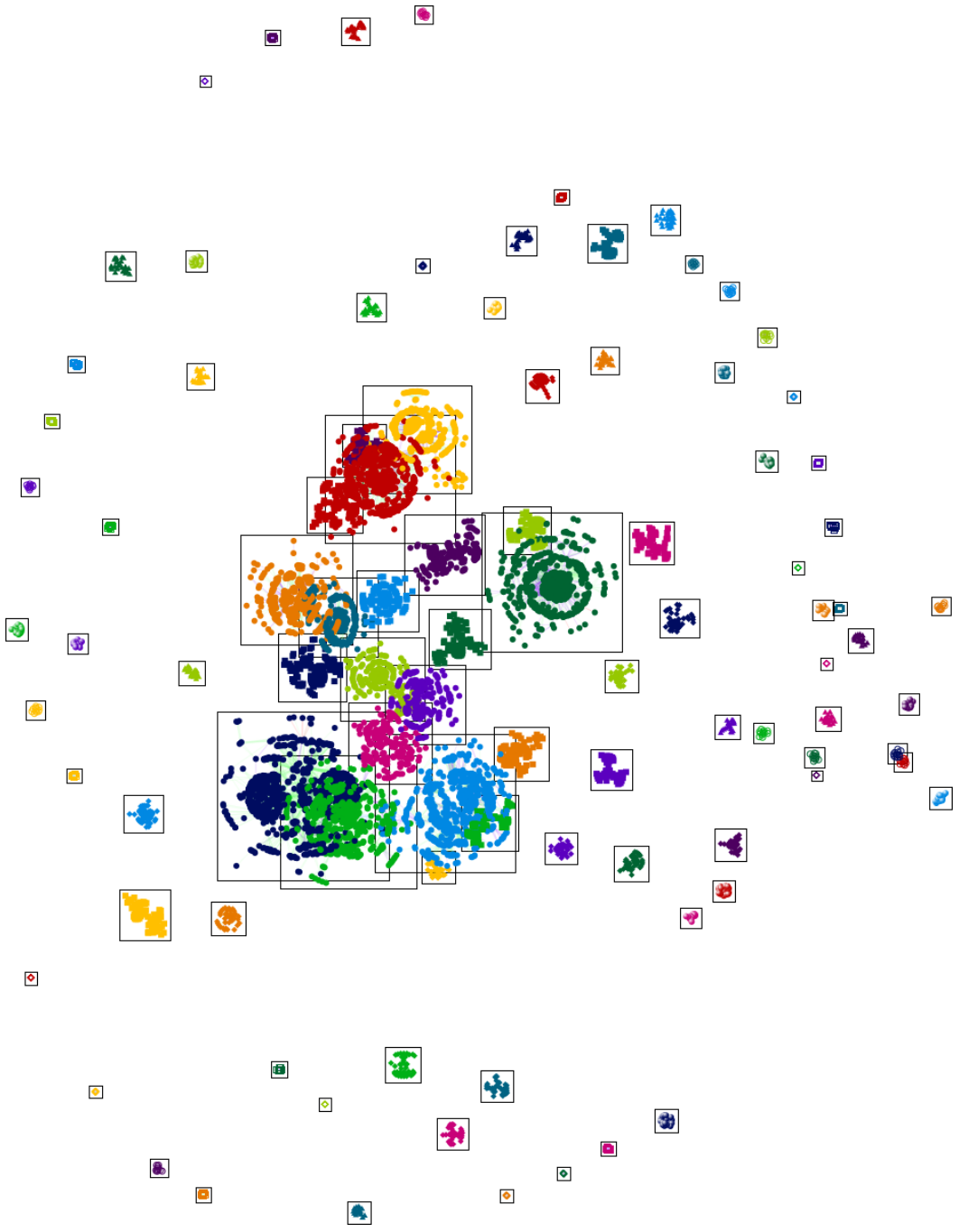


Figure 5.17: Group box positions after running the Harel-Koren FMS layout [HK02a] on the group relationship network of innovations in Pennsylvania (see Section 5.5.2 for dataset details). Edges between groups are hidden.

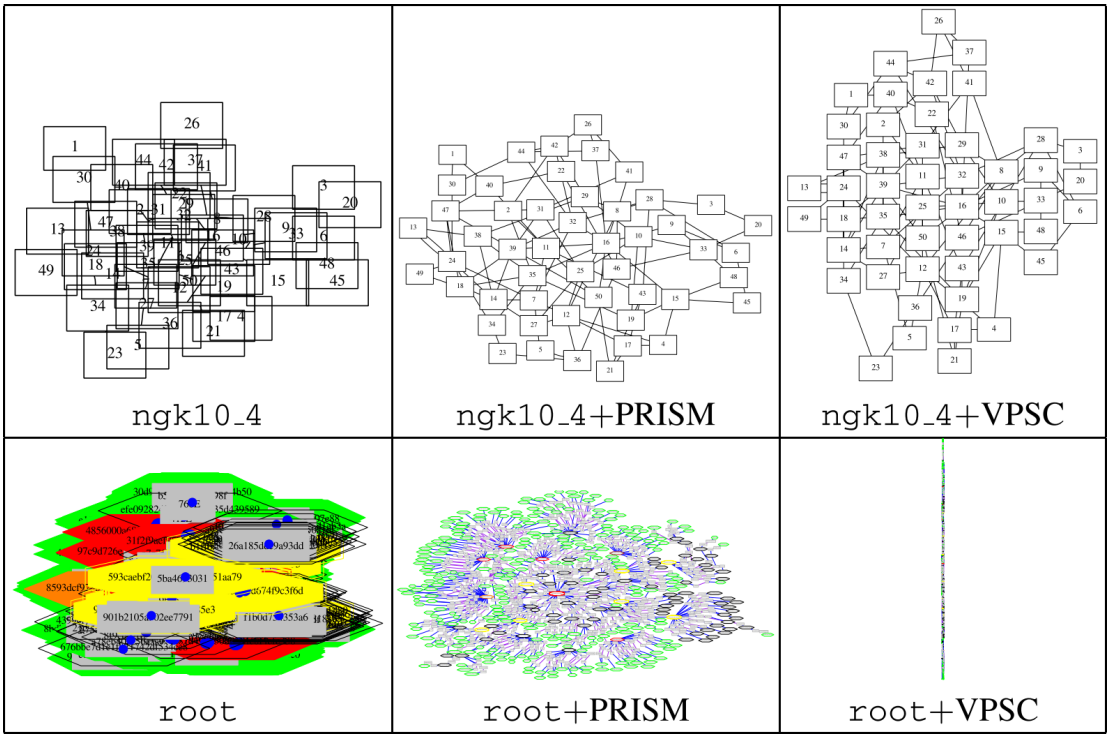


Figure 5.18: An original network visualization (left), the same visualization after removing node-node overlap with the PRISM algorithm [GN98] (center), and after removing node-node overlap with the solve_VPSC algorithm [DMS06; DMS07]. solve_VPSC maintains orthogonal ordering but can result in highly skewed visualizations. From [GH09].

overlap. According to Gansner and Hu's evaluations, the PRISM approach scales up well to large networks while maintaining a good tradeoff between preserving the network shape and limiting the area required by the adjusted visualization.

Several other alternatives exist, though they have various problems with scaling up or preserving the network shape. For example, the scan line approach of Dwyer, Marriott, and Stuckey; Dwyer, Marriott, and Stuckey [DMS06; DMS07] is a quadratic programming algorithm removes overlaps and maintains orthogonal

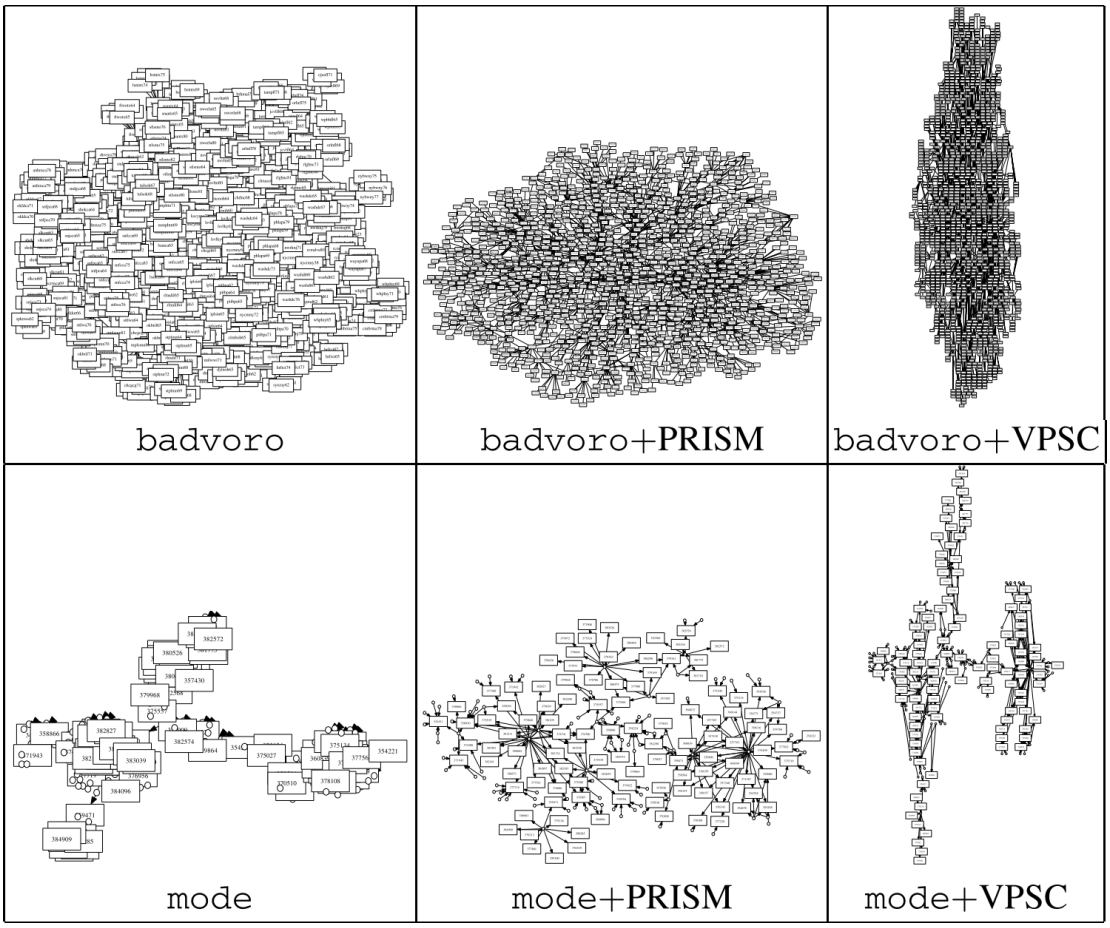


Figure 5.19: An original network visualization (left), the same visualization after removing node-node overlap with the PRISM algorithm [GN98] (center), and after removing node-node overlap with the solve_VPSC algorithm [DMS06; DMS07]. solve_VPSC maintains orthogonal ordering but can result in highly skewed visualizations. From [GH09].

ordering. However, the visualization can become highly skewed as you can see in the right side of Figs. 5.18 and 5.19. For details on many other node-node overlap removal techniques see Section 6.4.1.

The effects of the box removal algorithm are illustrated in for a large network with an initial space-filling factor of 20% (Fig. 5.20) and 50% (Fig. 5.21). In these two figures, the initial positions for each group chosen by the layout algorithm (Section 5.4.3.2) are shown using colored circles, squares, diamonds, and triangles. The boxes, however, are drawn centered around their final non-overlapping positions. For groups with substantial movement, I have drawn a red line connecting the initial position shape to the center of the final box position. As the layout algorithm does not take the box size into account, there can be a substantial amount of adjustment required to reach the final positions. In Fig. 5.20, where the initial space-filling factor was 20% of the screen, there is relatively little movement and the groups retain their relative positions to each other, except for a few crossing movements in the bottom-left. On the other hand, in Fig. 5.21 where the initial space-filling factor is 50%, I see much more group movement.

In general, the the box overlap removal algorithm keeps the relative positions of the groups intact, but there can be large groups that get shoved out to the periphery with a high initial space-filling factor. This layout is more space-filling, but at the expense of obscuring some of the group relationship information. This

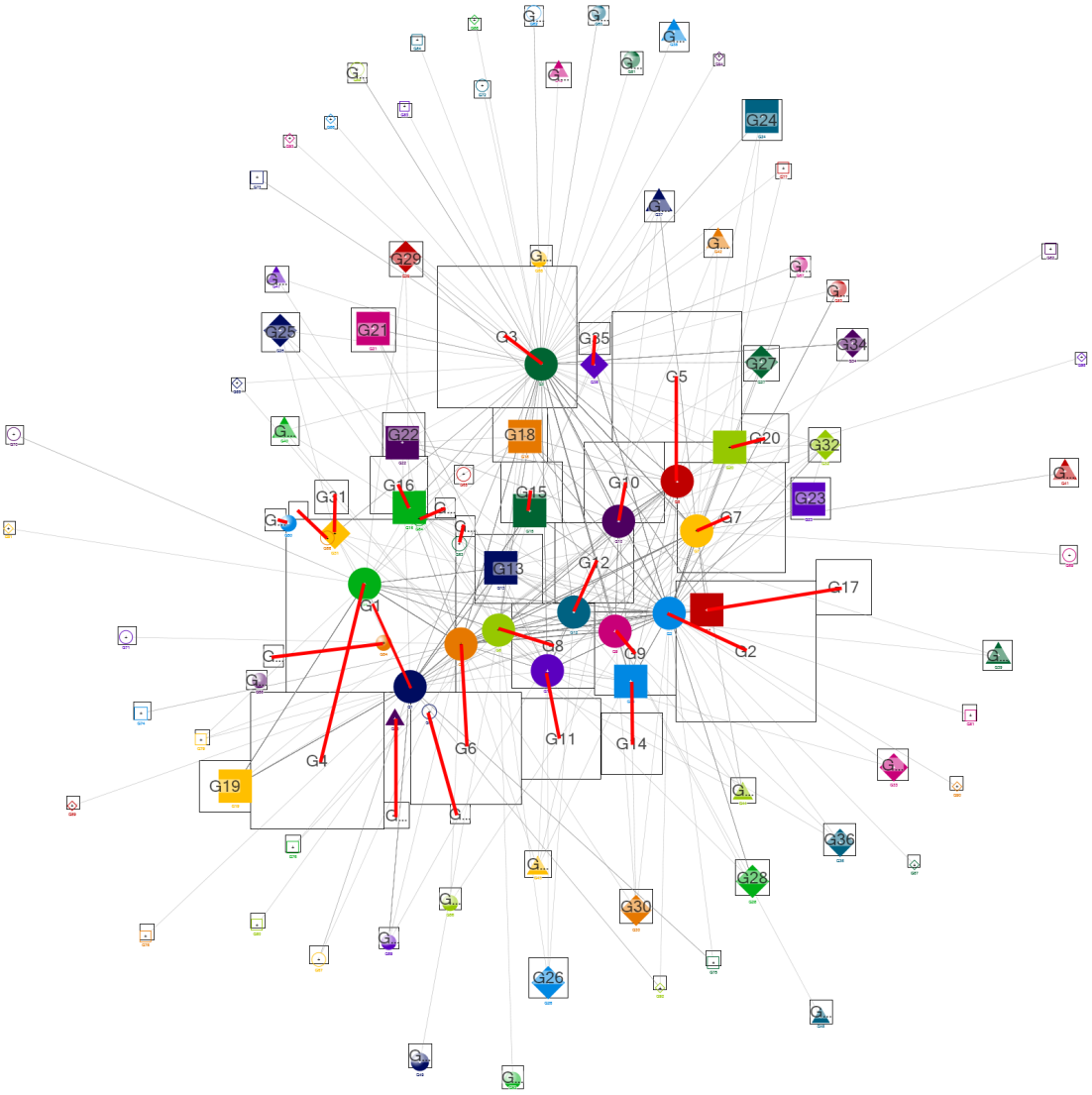


Figure 5.20: Network and groups from Fig. 5.17, using a different initial set of positions from the Harel-Koren FMS layout [HK02a] and after adjusting box positions using the PRISM overlap removal technique [GH09]. In this case I chose an initial space-filling factor of 20%. The red lines map the original group positions, represented by colored shapes, to the final box positions. There is generally little movement.

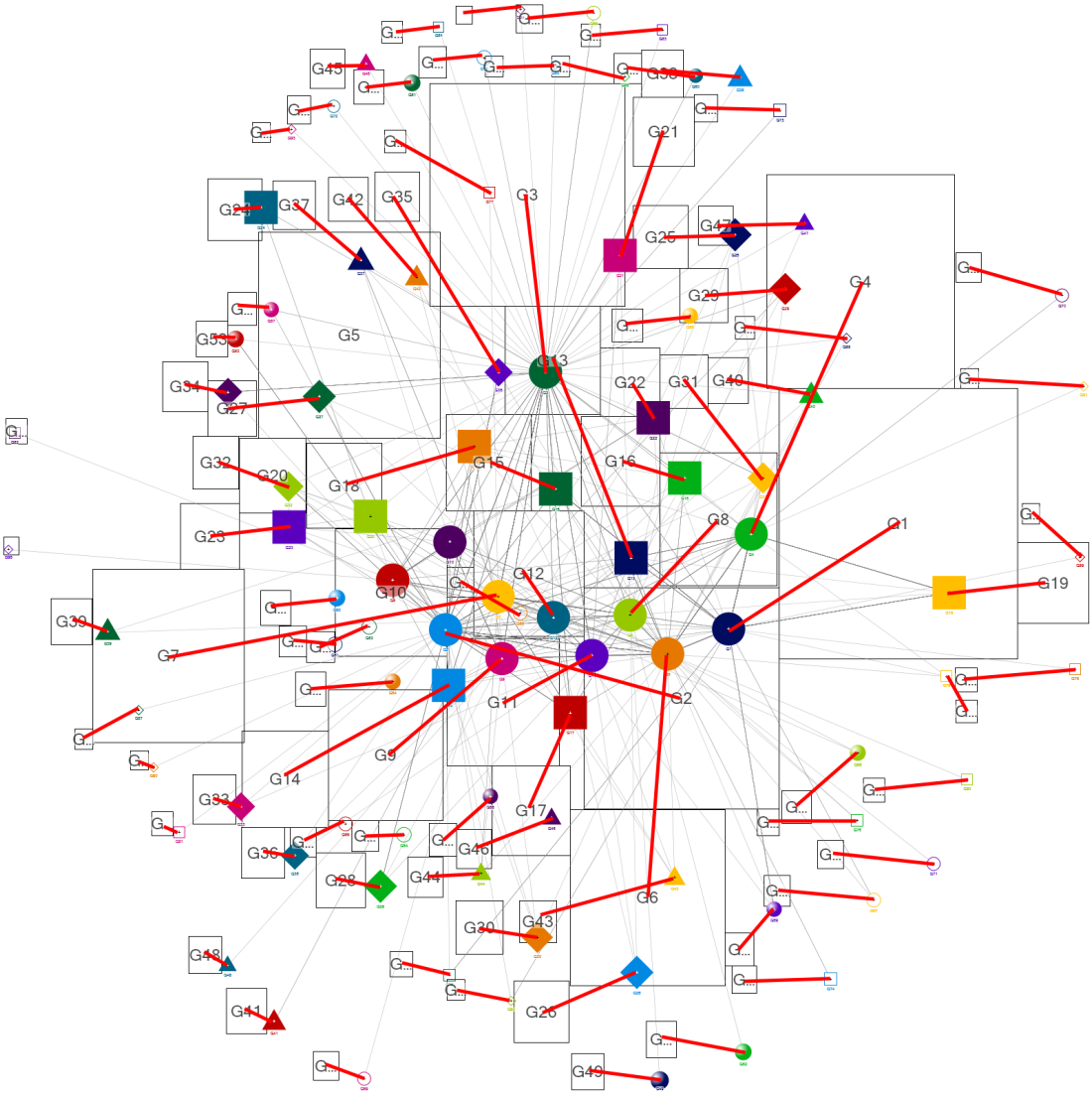


Figure 5.21: Network and groups from Fig. 5.17, using a different initial set of positions from the Harel-Koren FMS layout [HK02a] and after adjusting box positions using the PRISM overlap removal technique [GH09]. In this case I chose an initial space-filling factor of 50%. The red lines map the original group positions, represented by colored shapes, to the final box positions. There is a substantial amount of movement, and while most of it preserves group relationships the largest groups get shoved to the periphery.

problem could potentially be alleviated by reducing the amount of overlap removed in each iteration of the algorithm, allowing the large groups to slowly push small ones out of the way instead of shoving past them to the periphery. This parameter is referred to as s_{max} in [GH09] and must be larger than 1.0. I am currently using their suggested default value $s_{max} = 1.5$, but experimentation could be useful.

5.4.3.4 Finalize the Layout

I do have the initial space-filling factor which sets how much of the screen to use to display boxes, but because the box overlap removal technique usually needs to adjust the boxes outward the final screen space required can increase. If the layout has expanded outside of the available screen space, I then scale the new layout to fit in in the available space. Each box is scaled down using the ratio of the layout space required to the screen space, maintaining its aspect ratio.

As most layout algorithms, including the one I chose, use non-deterministic heuristics to place the nodes, it is possible to get a poor initial layout. For example, all the nodes can be placed on a diagonal line or the like. In those cases, the resulting visualization after overlap removal preserves that diagonal line and requires a significant amount of screen space. As with general force-directed layouts, the user can choose to run the layout again (or several times) to get the most effective force-directed group-in-a-box layout.

5.4.4 Showing Edges Between Groups

After running a Group-in-a-Box layout, we have an option as to how to show inter-group edges. NodeXL [Smi+10] currently supports three techniques: showing the actual underlying edges, hiding them completely, or replacing all the edges between each pair of groups with a meta-edge sized proportional to the number of edges it represents. These options are shown in Fig. 5.22, as well as many other images in this chapter. In addition to straight-line edges, NodeXL can draw curved or bundled edges that may reduce complexity (e.g., Fig. 5.31). However, because the group in each box is laid out independently of the rest of the network, showing the underlying edges explicitly often results in additional edge crossings and other poor layout characteristics. Depending on the user's target analysis tasks, using the combined meta-edges or hiding edges completely can be more effective.

5.4.5 Dividing the Problem

We now have three Group-in-a-Box layouts at our disposal: Treemap, Croissant-Donut, and Force-Directed. Users can choose the variant most suitable to their task using the Layout Options dialog in NodeXL Fig. 5.23, depending on whether they wish to have a highly space-filling layout (Treemap), one that highlights group relationships (Force-Directed), or somewhere in between (Croissant-Donut). However, the onus is not entirely on the user to pick the best layout for their

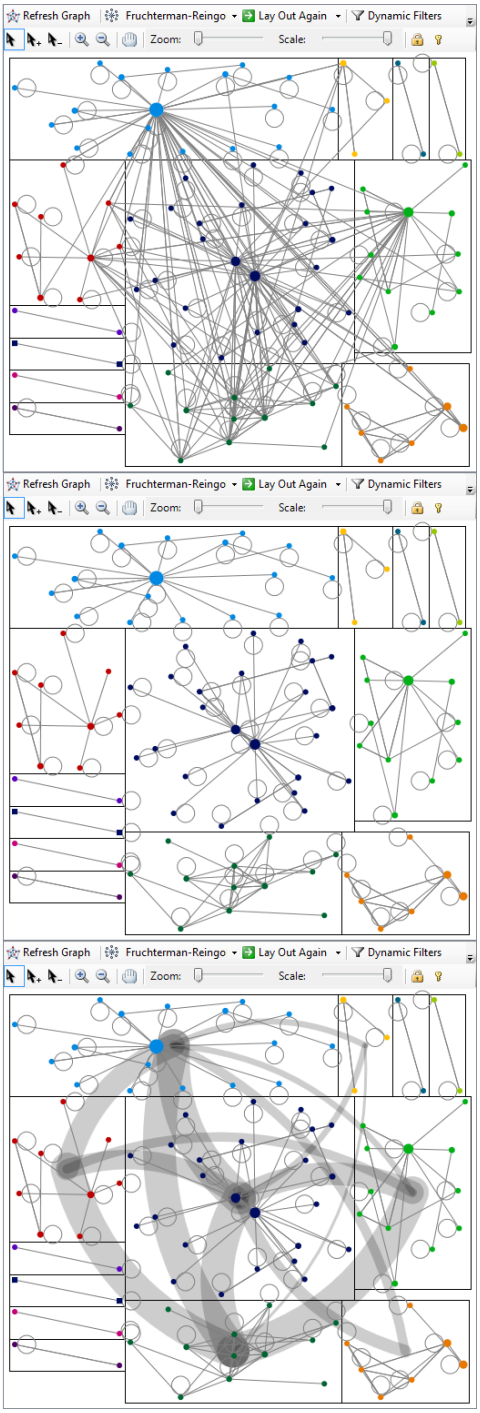


Figure 5.22: Three ways to show edges between groups in a Group-in-a-Box layout. From top to bottom: show all underlying edges, hide all underlying edges, and use aggregate meta-edges.

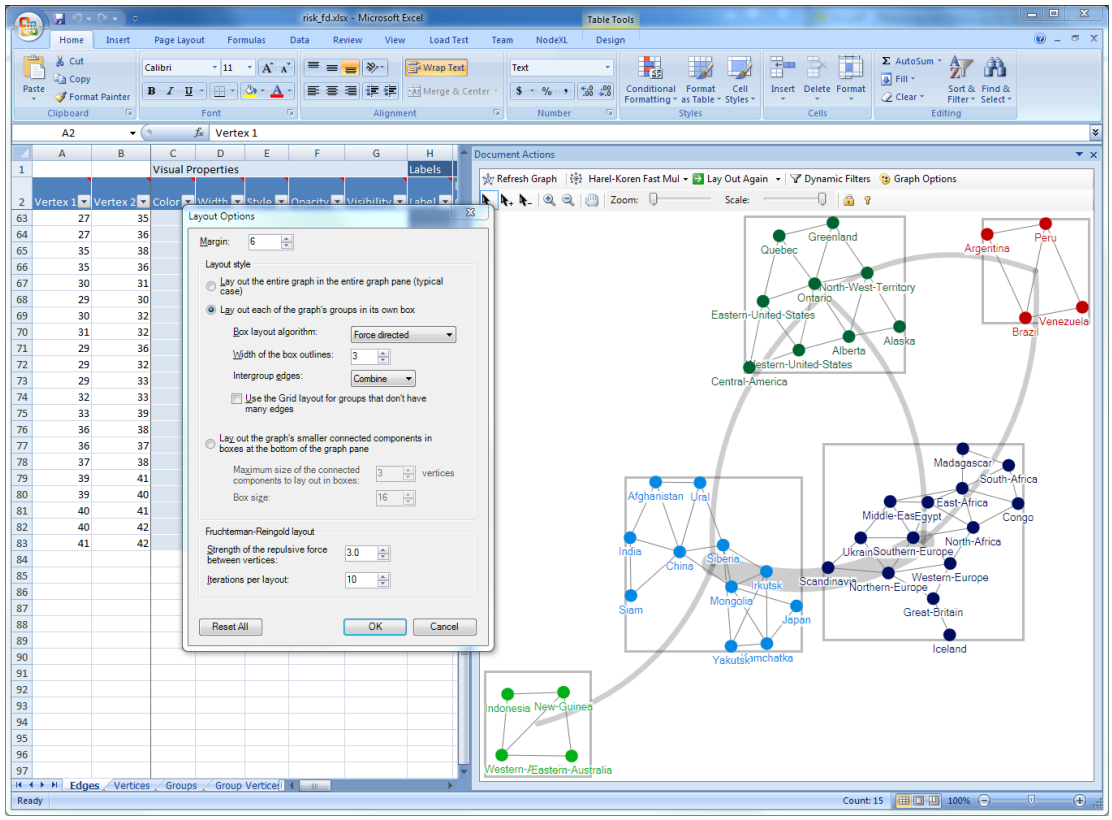


Figure 5.23: The NodeXL Group-in-a-Box user interface. The right graph pane shows a Force-Directed Group-in-a-Box layout of the Risk network, which is described further in Section 5.5.1. The left Edges worksheet shows some of the edges connecting the nodes in the network. The Layout Options dialog in the foreground allows users to select their desired Group-in-a-Box layout, the size of group boxes, how to treat inter-group edges, and whether to use a separate grid layout for groups with few edges instead of the chosen main layout.

network. I make several choices for the user algorithmically, so as to reduce novice user difficulties and speed up the analysis process.

5.4.5.1 Disconnected Components

First, I find any disconnected components in the network and lay each component out individually in a rectangular screen region using the Treemap Group-in-a-Box layout by default. This ensures that disconnected parts of the network are not drawn on top of each other, as most layout algorithms assume a single connected component. Moreover, the Treemap layout more effectively subdivides the screen space for the various group sizes, compared to NodeXL's current option of putting components below a certain size in small same-sized boxes at the bottom of the screen. This also is more space-filling than the approach used by Cytoscape [Sha+03], which lays the components out individually, sorts the components by screen space used, and stripes them in rows where each row is as tall as its tallest component (Fig. 5.24). After the top-level component Treemap layout, if the user is using a regular Group-in-a-Box layout, each component's region will be further subdivided using the chosen algorithm if it contains any groups. This can result in a two-level Treemap, where the top level divides the network by components and the second divides components by any groups present in the groups worksheet. Alternatively, a Force-Directed or Croissant-Donut Group-in-a-Box layout can be used for the second level.

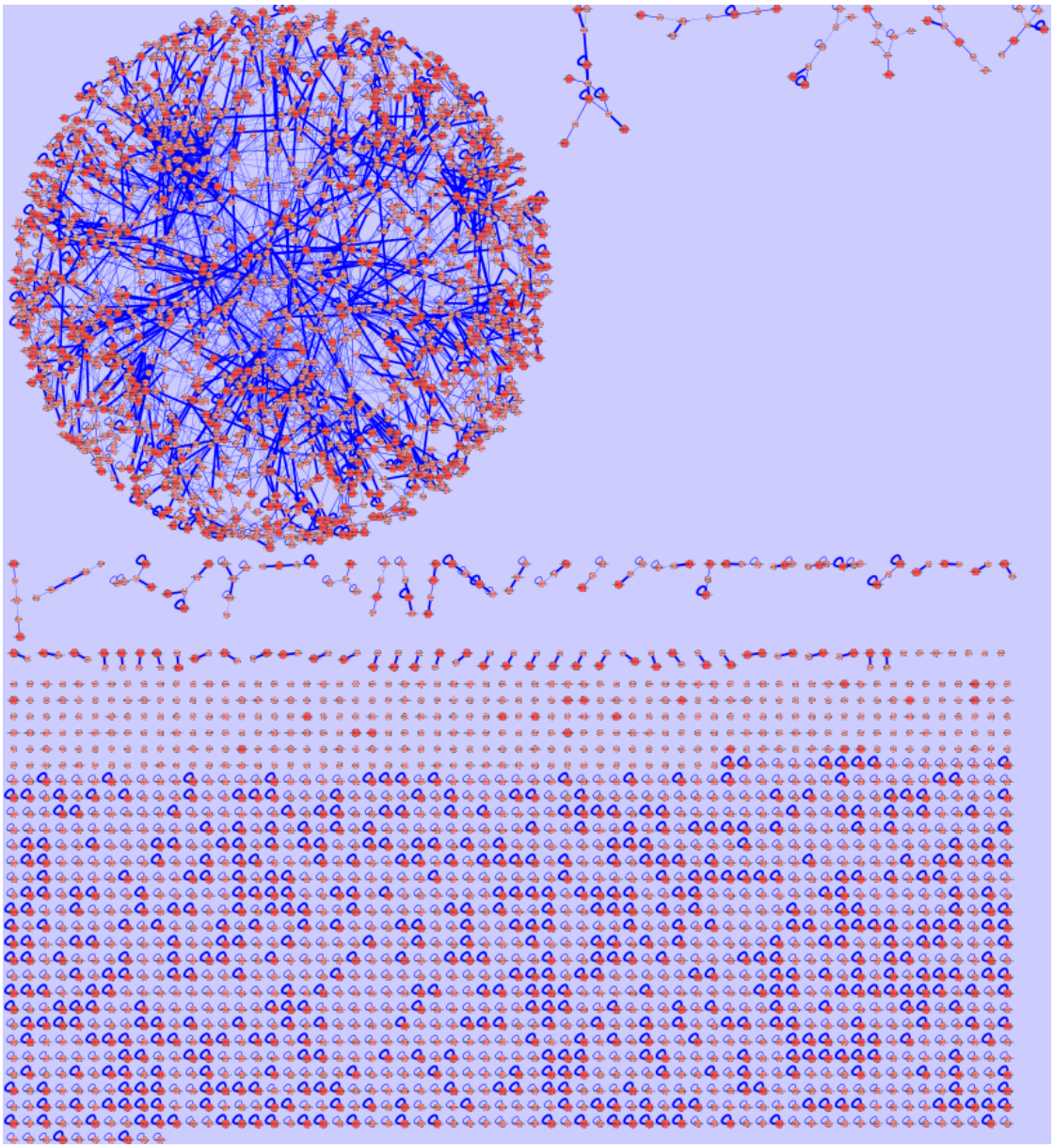


Figure 5.24: The Cytoscape biologic network analysis tool [Sha+03], currently showing the human protein interaction network after applying graph summarization [NRS08]. Disconnected components are laid out individually, sorted by screen space used, and striped into rows with each row height set by the tallest component. This can waste substantial screen space when components have drastically different sizes.

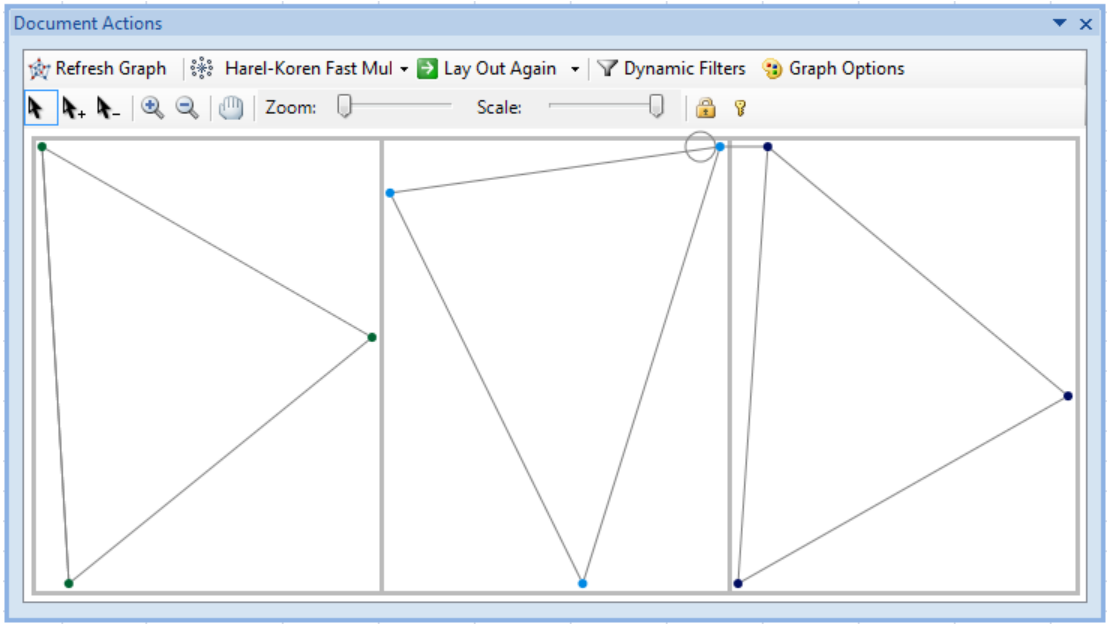


Figure 5.25: Three simple groups in the NodeXL squarified treemap layout demonstrating how window aspect ratio can cause three groups to be laid out in a row.

5.4.5.2 Number of Groups

The second automatic layout choice deals with the number of groups present in a component. If there is only one group (the complete subnetwork), no Group-in-a-Box layout is used. If there are two groups, I choose the Treemap layout to divide the space in half proportionally. For three or more groups, I use the user's selected Group-in-a-Box layout. Ideally, a squarified Treemap layout would allow perfect representation of relationships between three groups without edges unnecessarily crossing group boundaries. However, high aspect ratio layout spaces like those possible when resizing the NodeXL graph pane can result in poor Treemap layouts for showing three-way relationships (Fig. 5.25).

5.4.5.3 Distribution of Group Sizes

Between the two variants of the Croissant-Donut Group-in-a-Box layout, the Donut should be preferred when there are many small groups in the network. Alternatively, if there are one or two big clusters and a few small clusters the Croissant layout will provide a more space-filling layout. We have defined a measure called **G-skewness** to measure this property, defined as the fraction of the network's nodes present in the two groups with the highest **connectedness** (see Section 5.4.2) for a definition of connectedness). We have empirically determined cutoffs of G-skewness that we use to automatically choose the Donut or the Croissant variant depending on this group structure:

- Case1: $|\text{Groups}| \leq 3$ and G-skewness < 0.1 – Use the Treemap layout
- Case2: $|\text{Groups}| > 3$ and $0.1 \leq$G-skewness ≤ 0.45 – Use the Donut layout
- Case3: $|\text{Groups}| > 3$ and G-skewness > 0.45 – Use the Croissant layout

5.5 Case Studies

I explored several real-world networks with the three Group-in-a-Box layouts to determine their effectiveness. Examples of these case studies are detailed below. Two of these studies in particular involved extensive collaboration with domain experts to solve real-world problems: the innovation network in Section 5.5.2 and the medical informatics network in Section 5.5.3.

Figure 5.26: The box and board for the game Risk. The board consists of 42 countries in six continents. From boardgamegeek.com/image/1466865/risk

5.5.1 Continent-Holding Strategies in Risk

One small network that may be meaningful to a broad audience (of geeks at least) is that of the board game Risk. Risk is a turn-based strategy war game from Hasbro, originally released in 1957. The game is played on a political map of Earth with forty-two countries grouped into six continents. On their turn, users collect armies based on the countries and continents they occupy, then attempt to capture countries adjacent to the ones they occupy with combat a matter of attrition

resolved via dice rolling. The game board and pieces are shown in Fig. 5.26. I created a network from the game board, where nodes represent countries and edges between them indicate valid movements across country borders.

This network is shown in Fig. 5.27, where the nodes are laid out using the Harel-Koren FMS layout [HK02a] and clustered/colored using Clauset-Newman-Moore [CNM04]. From this visualization we can see the expected segmentation into continents, which are generally more insular, with specific routes to other continents. For example, we can see the red South America on the left, the light green Australia in the bottom-right, and dark green North America at the top. Holding these three continents can be quite beneficial, as they provide troop bonuses and have limited access. However, we can see in the center that the purple cluster is a combination of Europe and Africa, or EuroAfrica. These two continents are so tightly connected along the Mediterranean that they are considered as one by Clauset-Newman-Moore, indicating correctly that they are harder to hold. Moreover, we see the Middle East is clustered into the bottom-right of EuroAfrica, although it is part of light-blue Asia in the game. As any Risk aficionado or "The Princess Bride" fan can tell you, "never get involved in a land war in Asia" – and this clustering result indicates one of the reasons why!

I also looked at this network using the three Group-in-a-Box layouts, and the results are shown in Figs. 5.28 to 5.30. As expected, the Treemap GIB layout

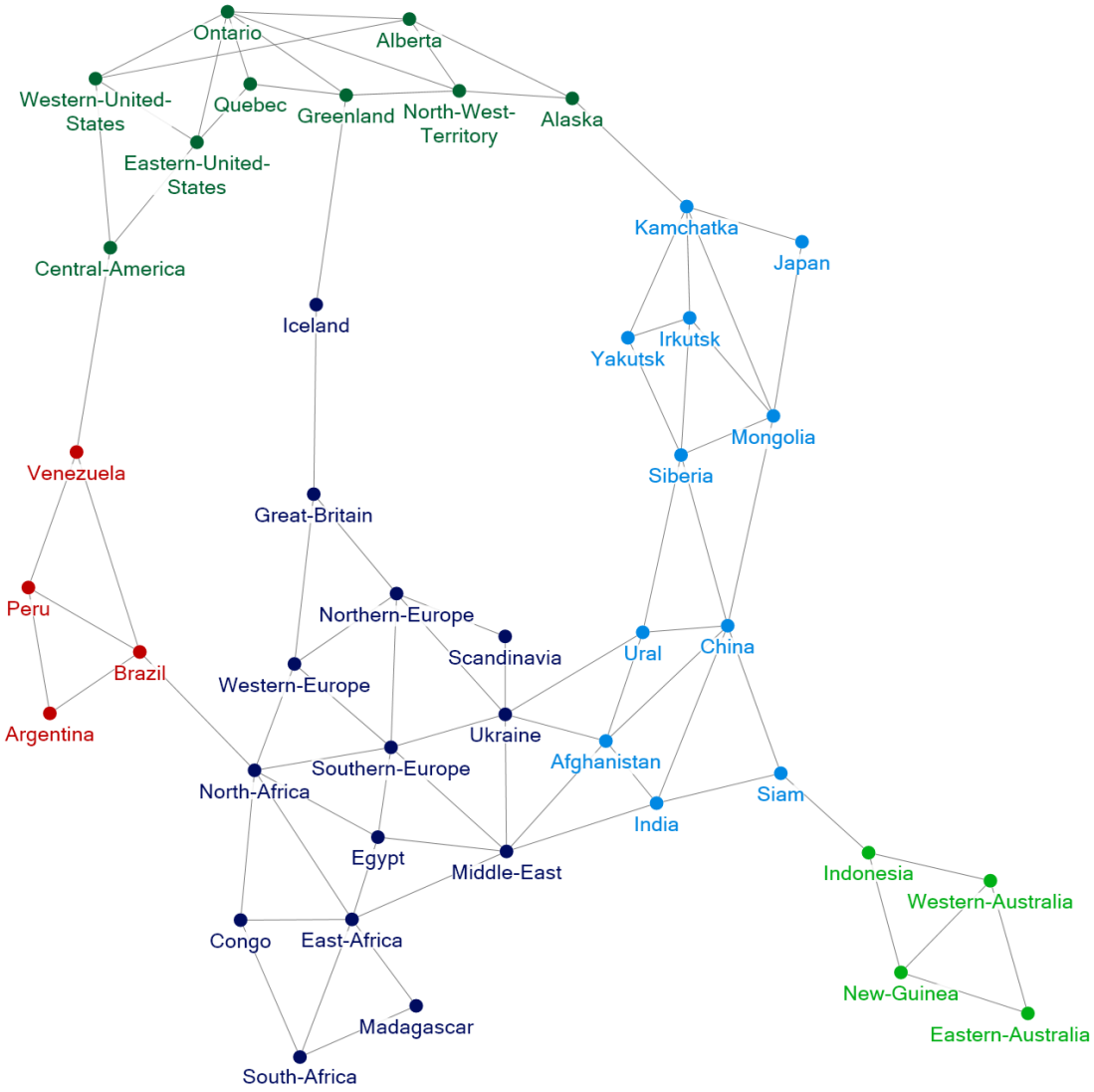


Figure 5.27: The network for the board game Risk, where nodes are countries and edges indicate valid movements. Nodes are laid out using Harel-Koren FMS [HK02a], clustered and colored using Clauset-Newman-Moore [CNM04].

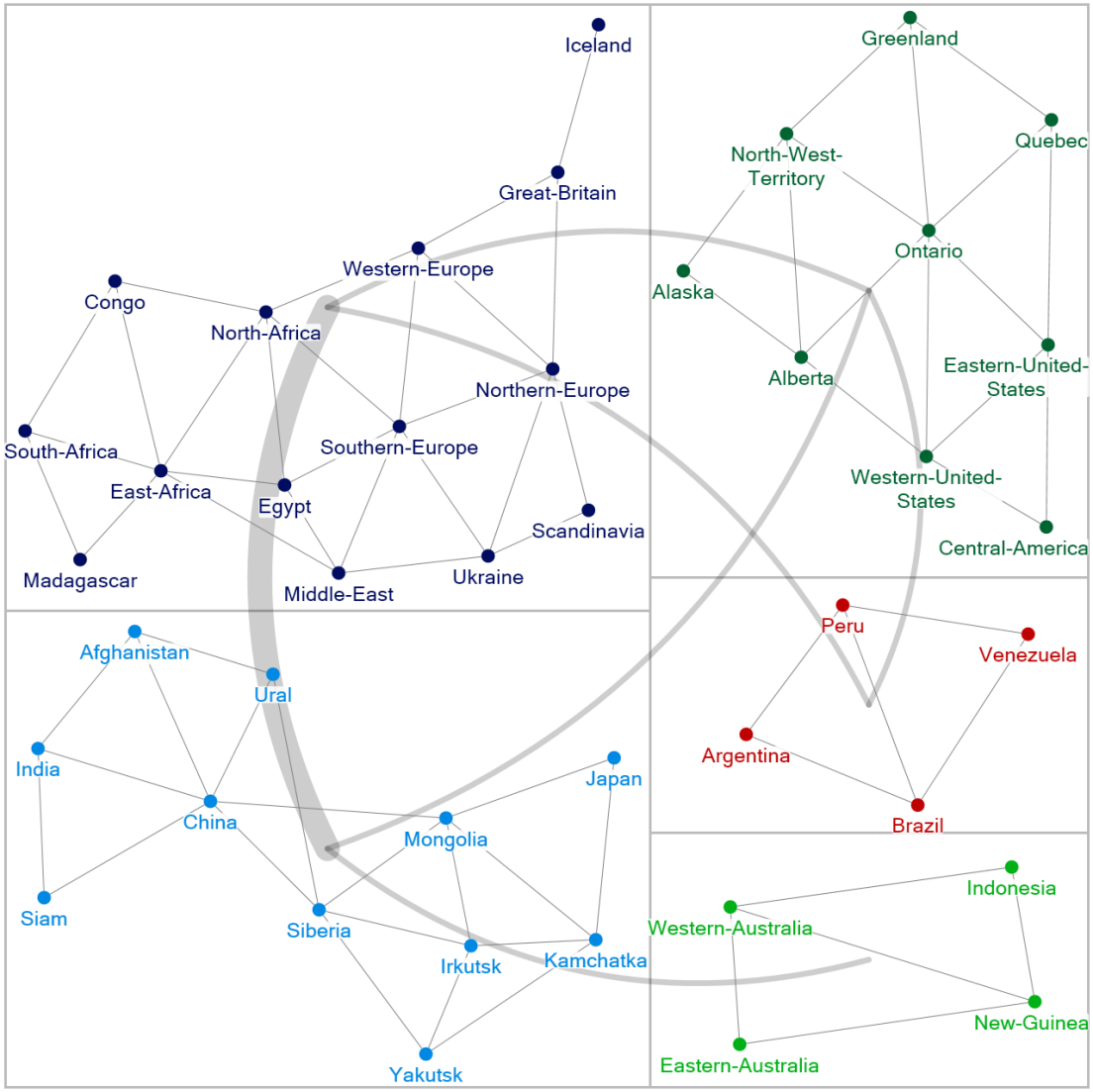


Figure 5.28: Risk network from Fig. 5.27, shown using the Treemap GIB layout with combined inter-group edges.

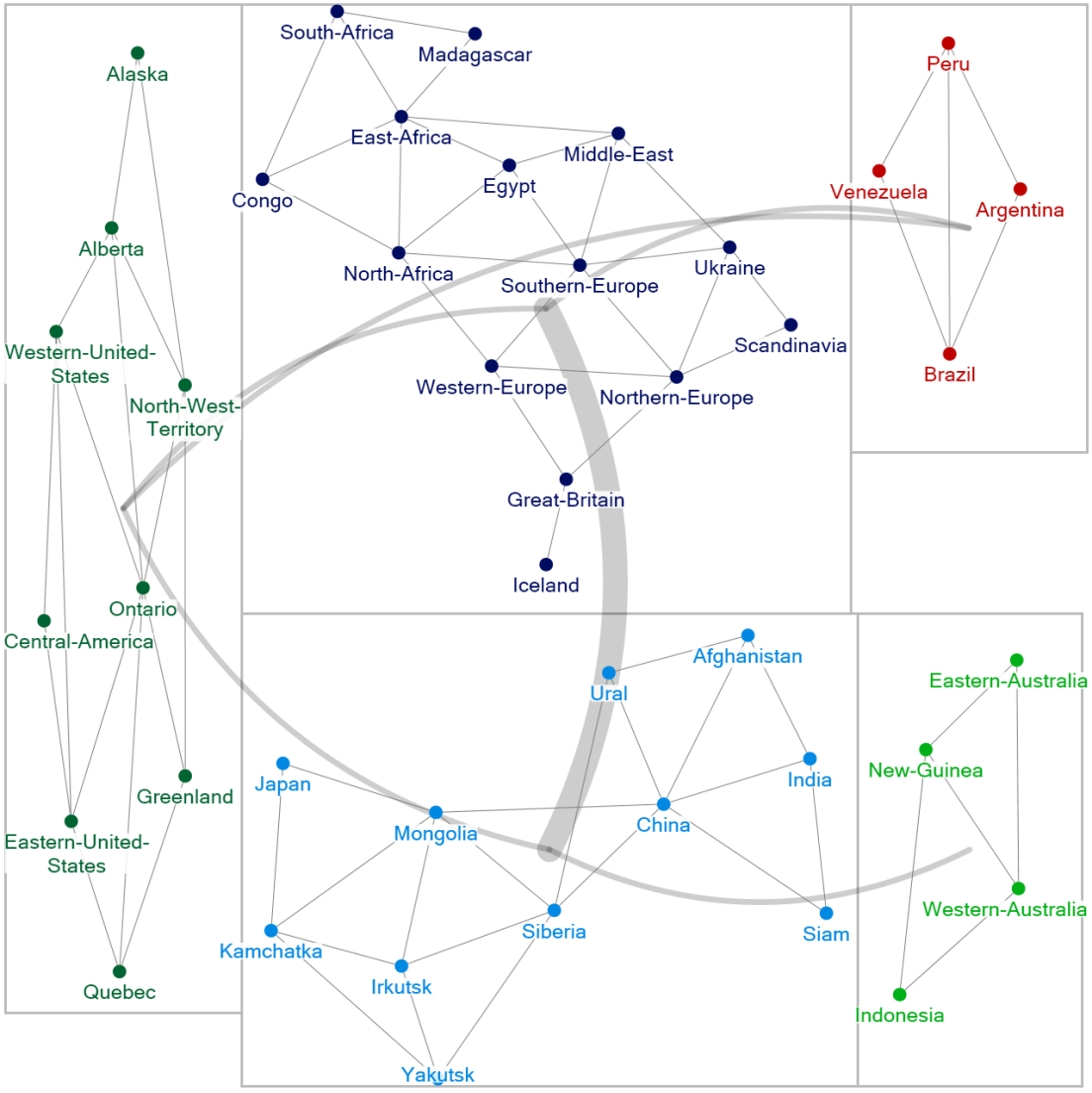


Figure 5.29: Risk network from Fig. 5.27, shown using the Croissant variant of the Croissant-Donut GIB layout with combined inter-group edges.

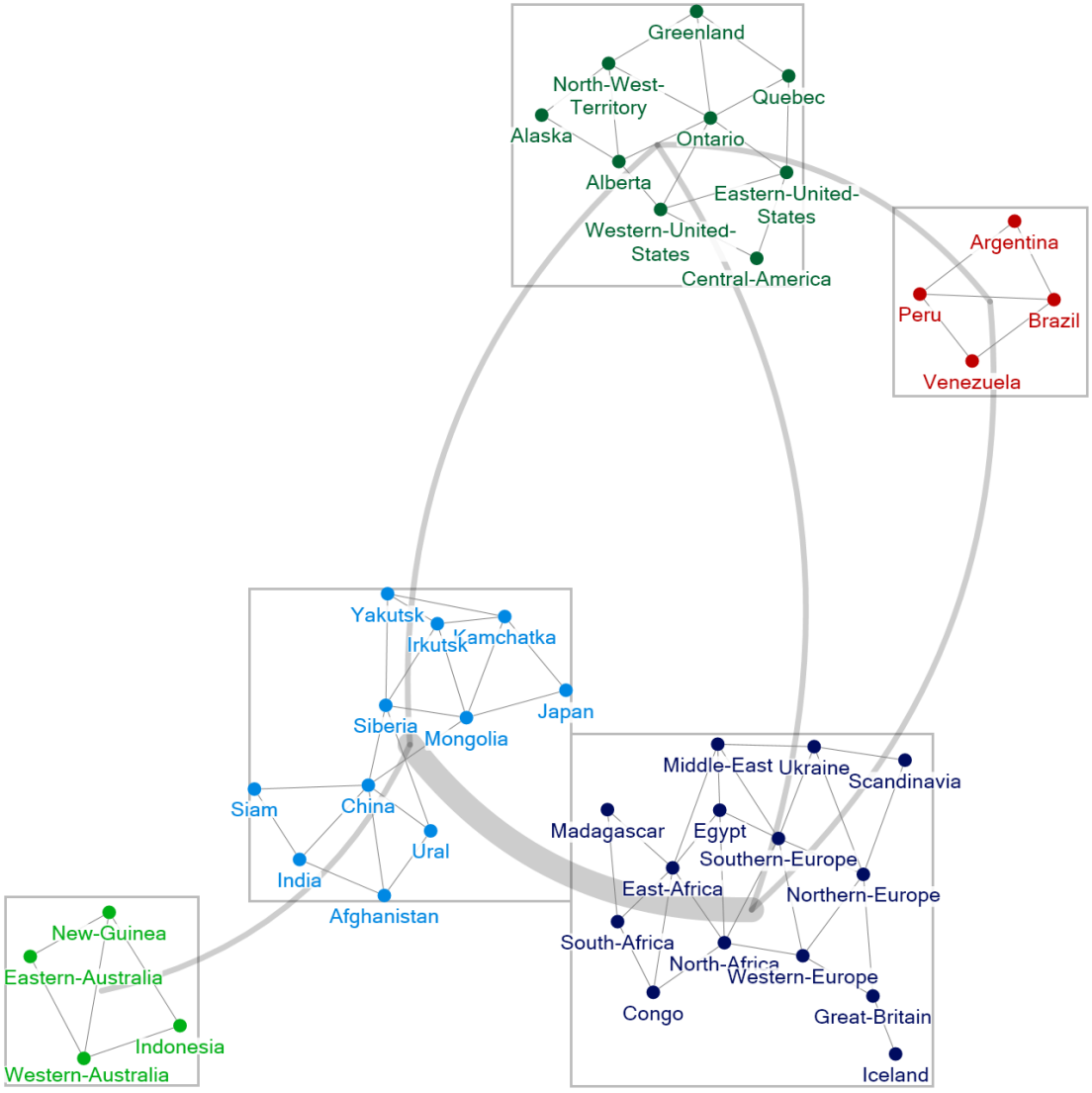


Figure 5.30: Risk network from Fig. 5.27, shown using the Force-Directed GIB layout with combined inter-group edges. The initial space-filling factor is 20%.

(Fig. 5.28) uses the space exceptionally well while maintaining small aspect ratios for the group boxes. The structure of each cluster is far more readable than in the original node-link visualization (Fig. 5.27). The combined inter-group edges show the strength of connection between purple EuroAfrica and light blue Asia – further solidifying our concerns about holding them in the game. The rest of the groups are only joined by a single route of attack. However, we can see the unfortunate placement of red South America next to light green Australia and light blue Asia – neither of which it as any connection to.

Moving to the Croissant-Donut GIB layout, we see that this network was assigned the Croissant variant (Fig. 5.29). There is some wasted space along the periphery, and the group box aspect ratios are worse than in the Treemap GIB layout (Fig. 5.28). On the plus side, light green Australia is only next to light blue Asia, its sole tie to the world, and red South America is by purple EuroAfrica which it connects to. South America also has a tie to dark green North America on the left, but is unfortunately placed on the other side of the visualization.

Finally, we look at the network using the Force-Directed GIB layout with an initial space-filling factor of 20%, shown in Fig. 5.30. Because of the reduced group box size, the labels are smaller and group internal structure is less readable. However, the ties between clusters are now explicitly clear based on their locations and the lack of meta-edge crossings or overlaps.

Overall, this case study illustrates the trade-offs inherent in the the three techniques while at the same time highlighting effective strategies for the game Risk. Clusters of countries and their internal legal movements are most clear using the Treemap GIB layout, while the Force-Directed GIB layout highlights the movements possible between the clusters. The Croissant-Donut GIB layout strikes a balance between these two extremes.

5.5.2 Finding Regional Innovation Clusters

One of the goals of urban planners is to understand the relationships behind innovation and how the ties between organizations, individuals, and funding agencies affect growth. Christopher Scott Dempwolf,[2] a researcher in the School of Architecture, Planning and Preservation at the University of Maryland, has been working to model innovation based on patent ties, federal and state funding, and physical locations. I introduced Dempwolf to NodeXL and helped guide several of his network analyses, including one of Pennsylvania innovations in 1990. He was keen on detecting technology and talent clusters which could be positively influenced. The network he collected included patent ties, federal funding from SBIR/STTR, and state funding via the DCED and Ben Franklin Technology Partners.

An initial visualization of this network is shown in Fig. 5.31, which uses the Harel-Koren FMS layout [HK02a], link bundling, and categorical coloring for node

[2] http://www.terpconnect.umd.edu/~dempy

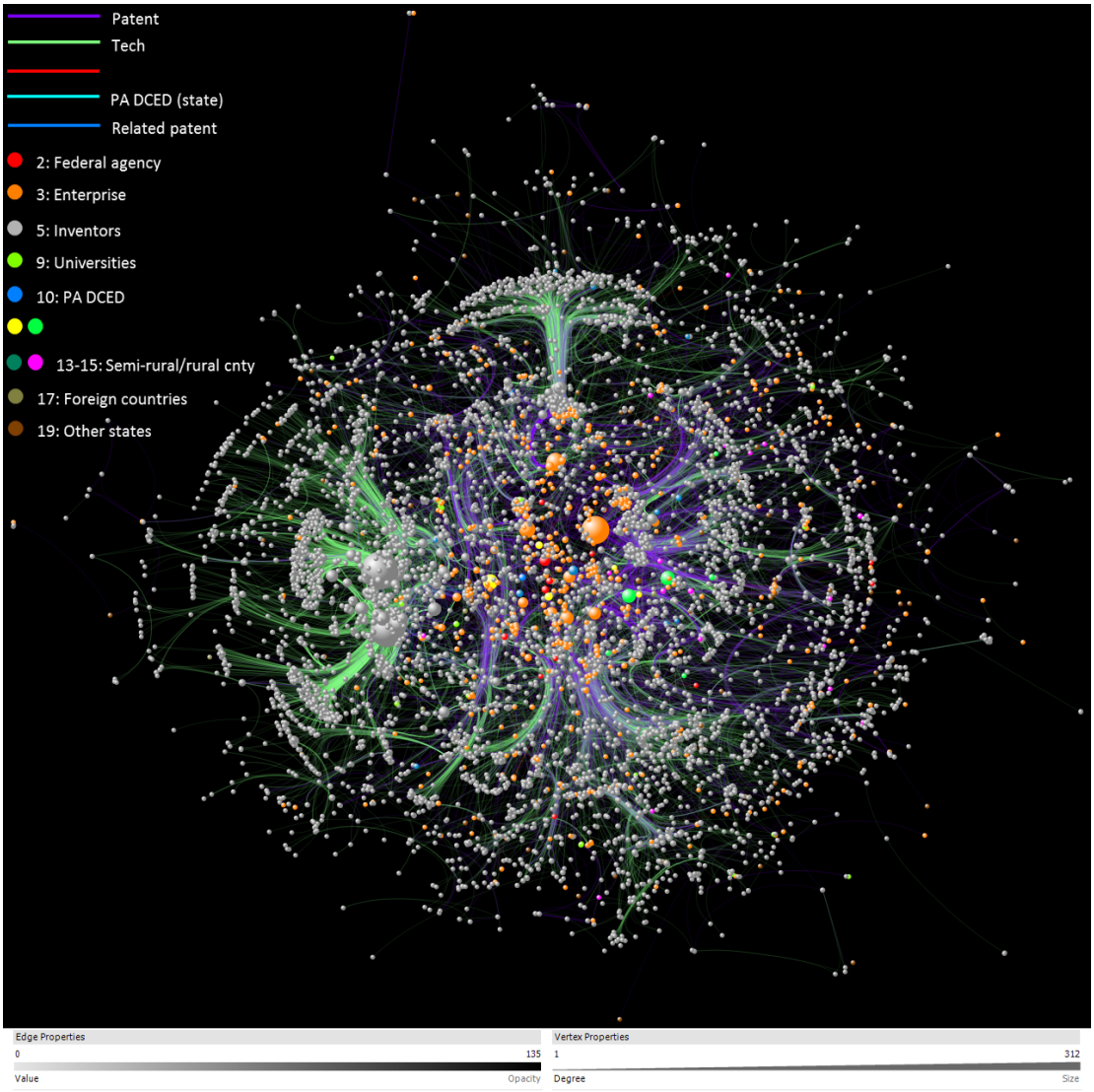


Figure 5.31: Pennsylvania innovation relationships during 1990 (main component) collected by Christopher Scott Dempwolf. Nodes are laid out using the Harel-Koren FMS layout [HK02a] and I used link bundling as well as categorical coloring for node and link types. Gray nodes represent inventors; orange are firms; red are federal agencies; royal blue are PA DCED / Ben Franklin agencies; lime are universities. Red ties (lines) are SBIR / STTR funding; purple ties are patent relationships; aqua ties are state funding; blue ties are explicit relationships between patents; light green ties are technology-based relationships.

and link types. While quite beautiful, this visualization is not particularly effective. Some large structures are easily distinguishable, like the cauliflower-shaped groups of gray inventors and a few large orange enterprises. However, the overall structures and relationships are hard to interpret.

Dempwolf was interested in technology and talent clusters, so to try to pick these features out of this large network I applied the Clauset-Newman-Moore clustering algorithm [CNM04]. The algorithm finds clusters of nodes that link to each other more frequently than outside the cluster, which, in this case, represents clusters of entities with similarities in patented technology. With a node-link visualization alone it can be challenging to see group membership, size, and aggregate relationships using solely the standard color or shape coding as in Fig. 5.32. I applied the Treemap GIB layout to make these features explicitly visible by laying out each detected cluster individually (Fig. 5.33).

In analyzing this visualization, we discovered many expected clusters around specific Pennsylvania counties and local enterprises. For example, the bottom-left cluster of Fig. 5.33 is the Pittsburgh metro area, containing the orange Westinghouse Electric. The Pittsburgh cluster is highly connected (via faint, small links) to the Montgomery county cluster to its right, another large metro area. An unexpected exception to the location grouping is the top-left pharmaceutical and medical cluster, composed of several companies, universities, HHS, and an interest-

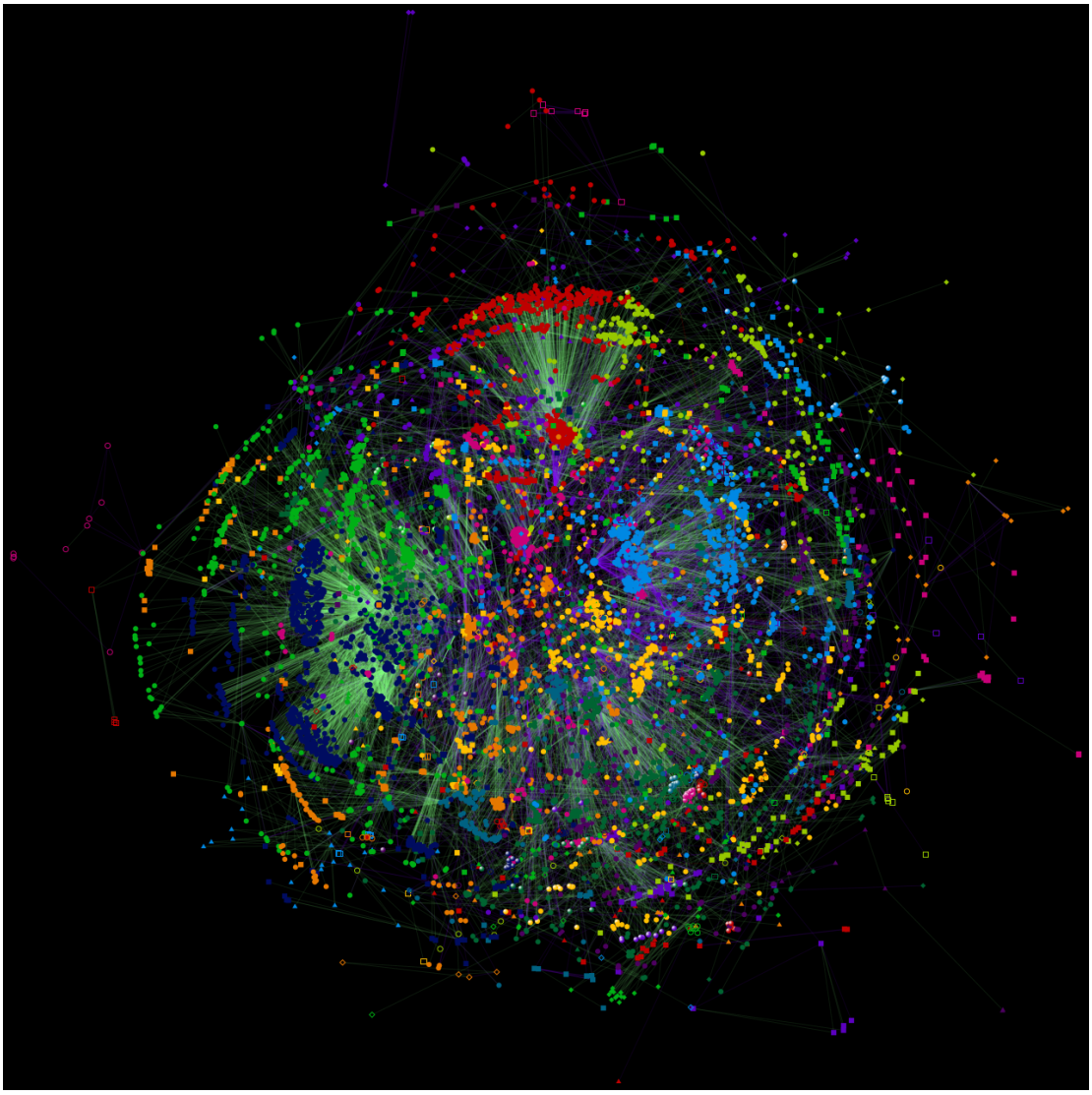


Figure 5.32: The innovation network from Fig. 5.31, with clusters found using the Clauset-Newman-Moore algorithm [CNM04] shown using node color and shape. Because of the dense, intermingled clusters it is difficult to understand the network and cluster structure. In this figure the edges are shown as straight lines.

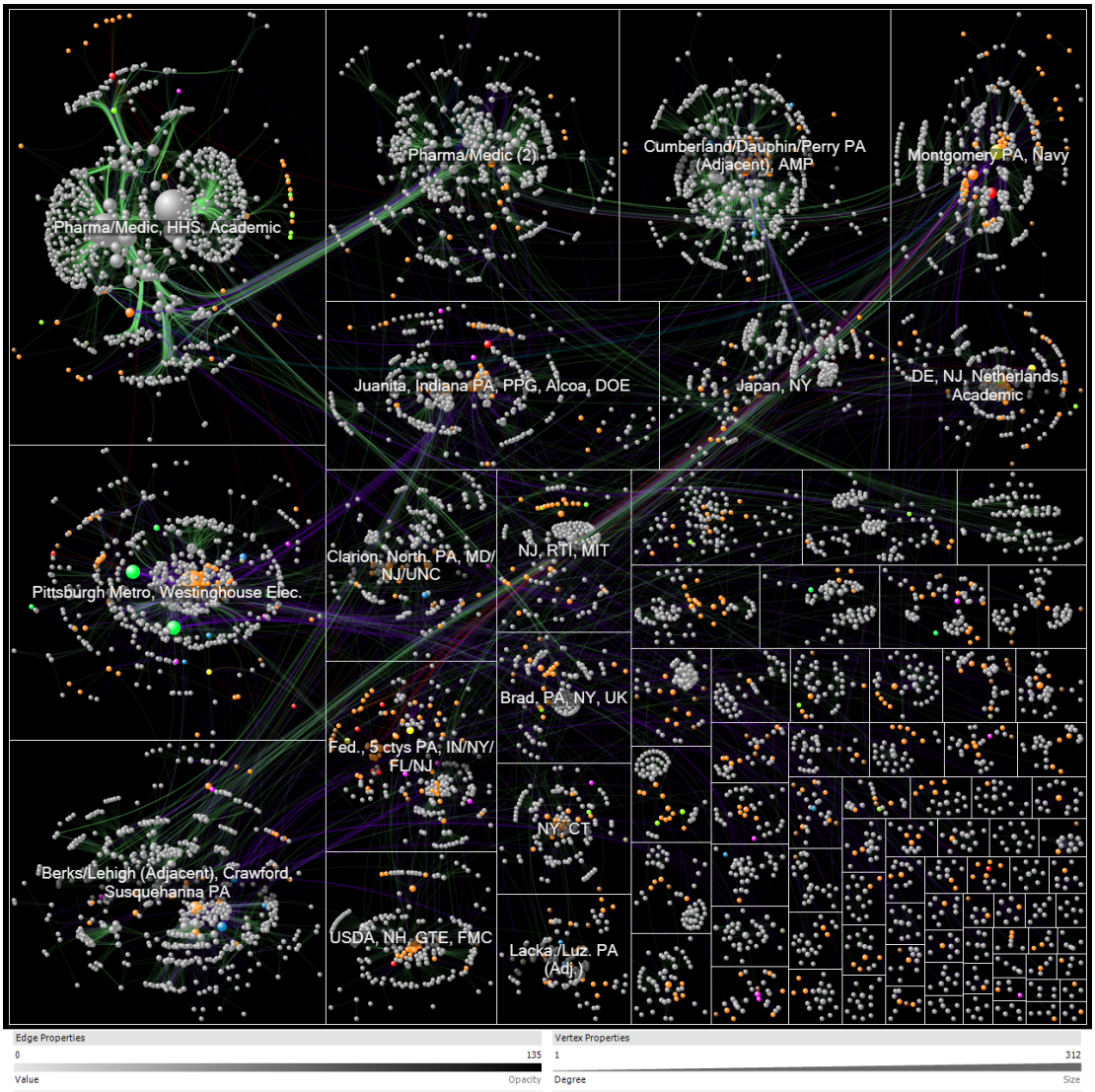


Figure 5.33: The innovation network from Fig. 5.31, with nodes grouped into boxes by the clusters found using the Clauset-Newman-Moore algorithm [CNM04], laid out using the Treemap GIB layout sized by their degree, and arranged inside boxes using the Harel-Koren FMS layout [HK02a]. Edge opacity is based on the tie strength and edges are bundled.

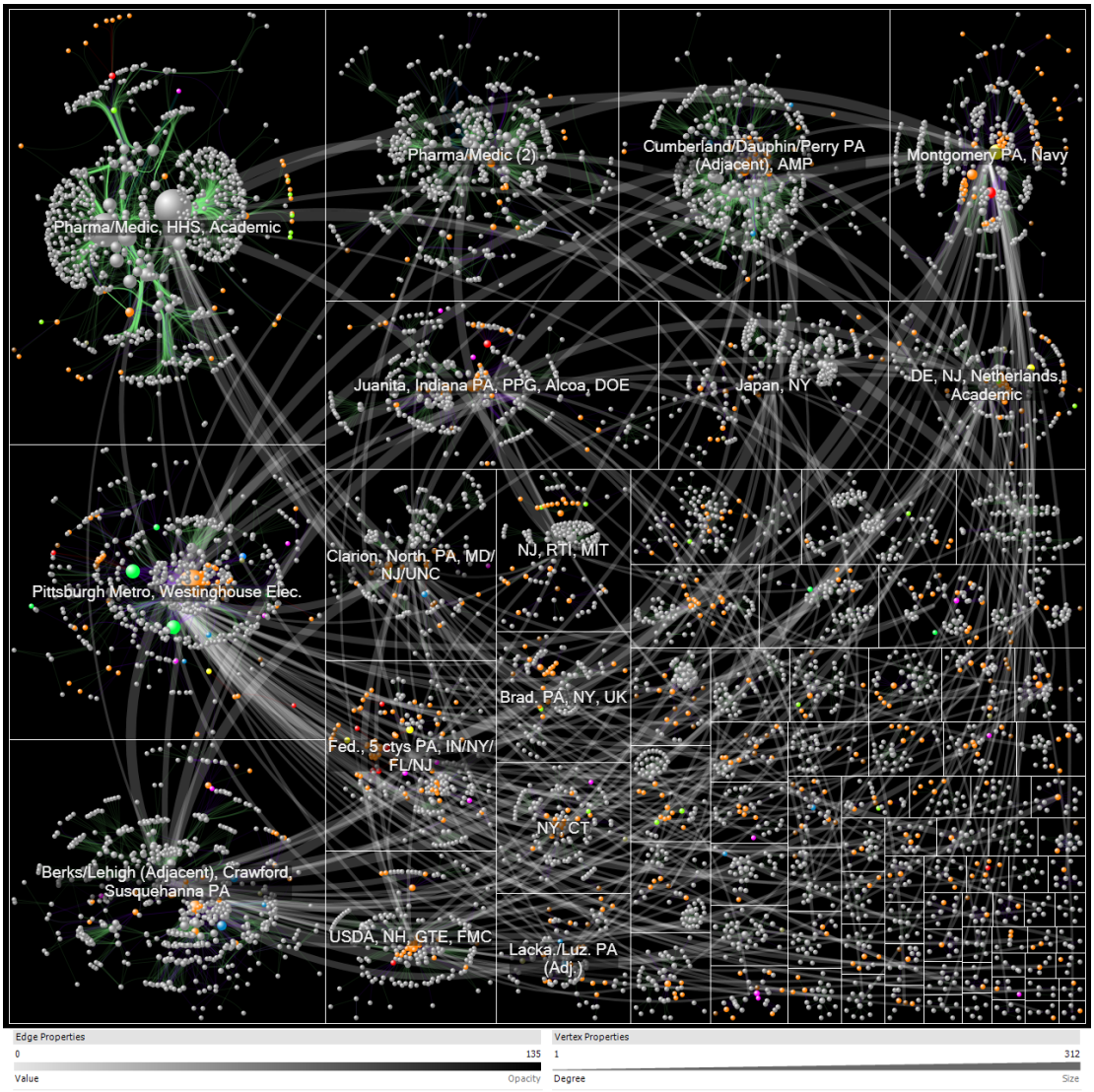


Figure 5.34: The visualization from Fig. 5.33 after replacing inter-group edges with meta-edges that represent the aggregate relationships between each pair of groups.

ing arrangement of inventors in several connected fans. Dempwolf was completely unaware of this cluster, which was immediately visible with the Treemap GIB layout. These sorts of meaningful structures were mostly hidden in the original visualization (Fig. 5.31).

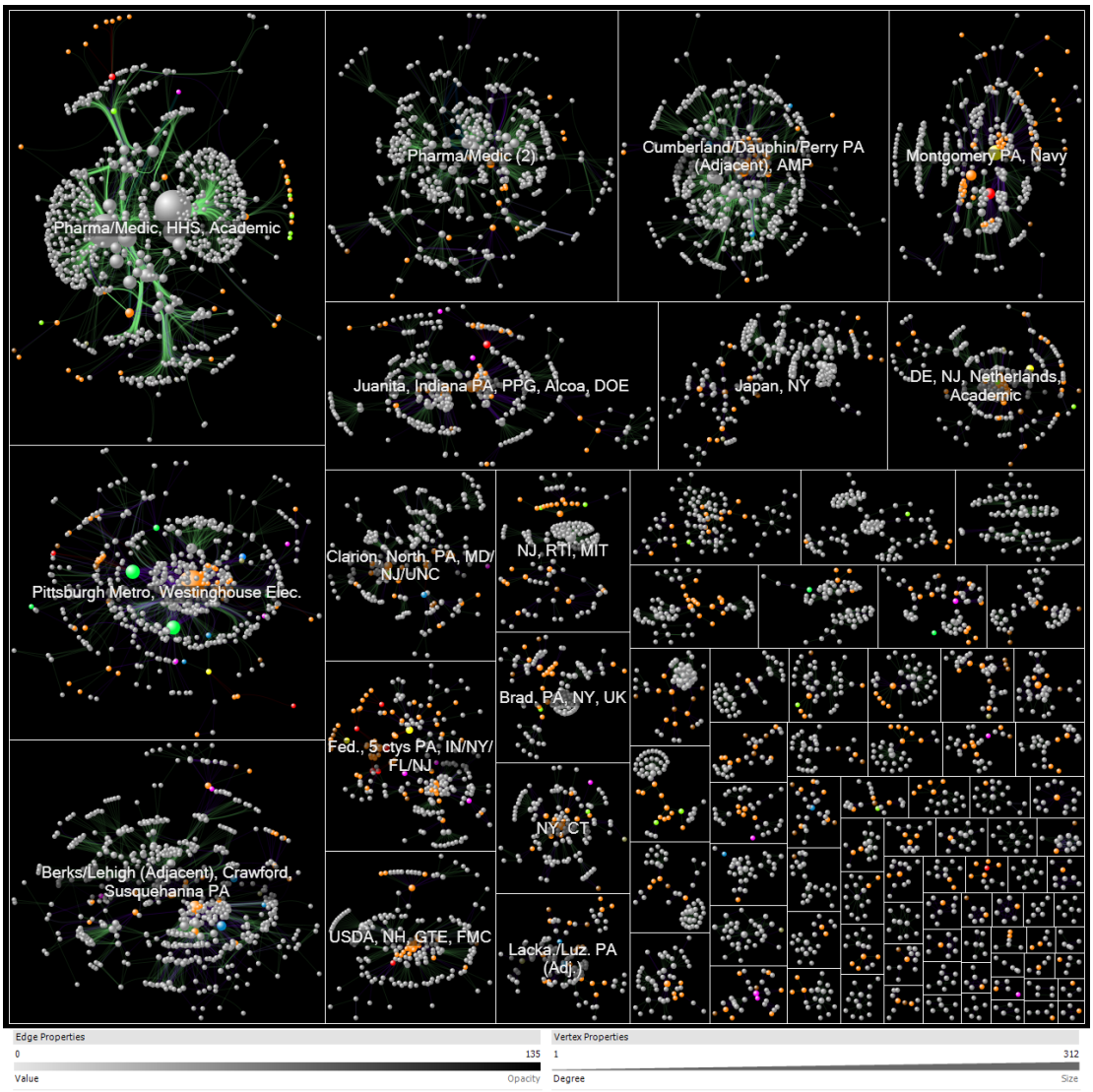


Figure 5.35: The visualization from Fig. 5.33 after hiding inter-group edges.

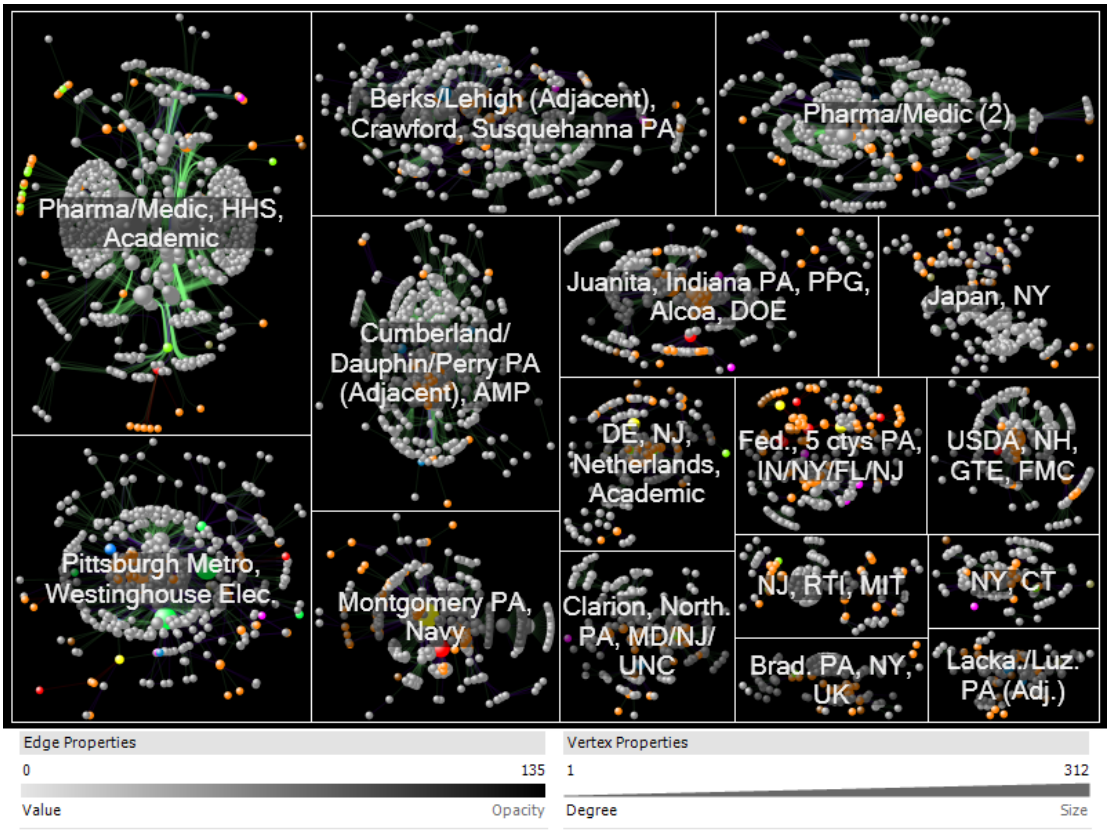


Figure 5.36: The visualization from Fig. 5.33 after hiding inter-group edges and filtering to only the largest groups.

However, the Treemap GIB layout can place highly connected groups of nodes far apart in the treemap, such as the two adjacent counties in Fig. 5.33 that are placed in the top-right and bottom-left corners. This makes it difficult to see aggregate relationships, with the edges stretched across many other groups they are not connected to. I attempted to show these aggregate relationships explicitly using a single aggregate edge between groups instead of the plethora of small ones, but the results were not encouraging (Fig. 5.34). In this example, there are too many connections between the various groups to be able to discern individual

ones. I hid all inter-group edges entirely (Fig. 5.35), which showed internal group structure more clearly, then drilled down to only the largest groups in the network to create Fig. 5.36. This kind of filtered, labeled visualization would be especially good for presenting the results.

Of course, I wanted to see how the other Group-in-a-Box variants handled this large, complex network. When I used the Fitted-Rectangles layout, our algorithms chose to use the Donut variant (Fig. 5.37). We can see the largest groups and their connections fairly well, but edges cross unrelated groups in some cases and many of the boxes have high aspect ratios. The smallest groups, which are shown as slices in the corners, have extremely high aspect ratios and should be filtered out. Alternatively, we could explore a combination with the Treemap GIB approach that subdivides corners when aspect ratios become to high.

My Force-Directed GIB approach, on the other hand, retains very good aspect ratios for the groups (Fig. 5.38). Moreover, the most tightly connected groups are placed near each other with the edges generally overlapping few other boxes. However, some of the largest groups are pushed to the periphery and thus their edges are drawn across unrelated groups. Some parameter tweaking may be necessary in the overlap reduction algorithm to avoid this. Also, there is much more wasted screen space than the Treemap GIB layout (Fig. 5.33) and the Croissant-Donut Donut GIB layout (Fig. 5.37).

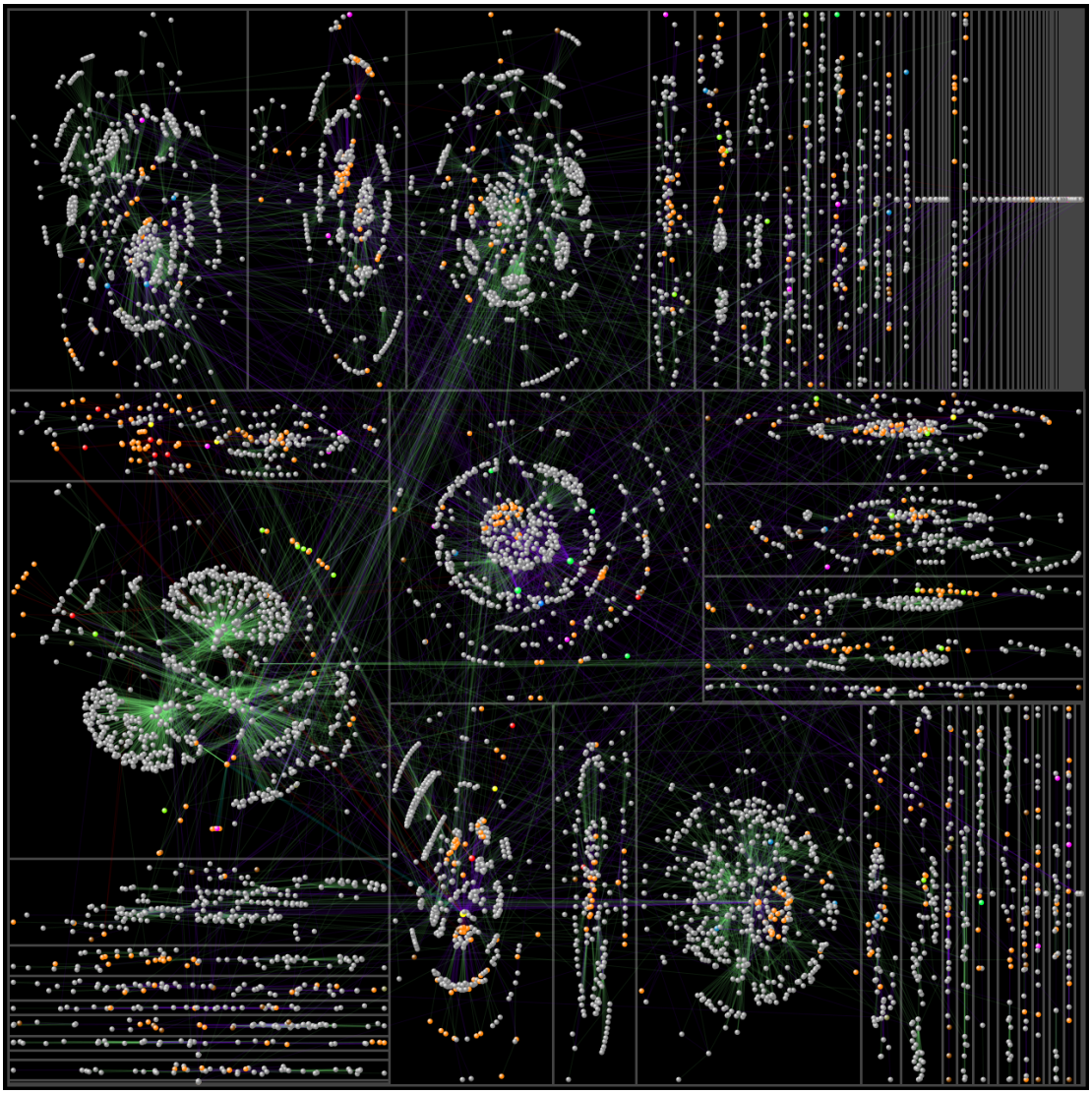


Figure 5.37: The visualization from Fig. 5.33, but using the Croissant-Donut Donut GIB layout instead of the Treemap. Inter-group edges are visible and straight. While we can see some of the groups well, many of the smaller groups in the corners have high aspect ratios.

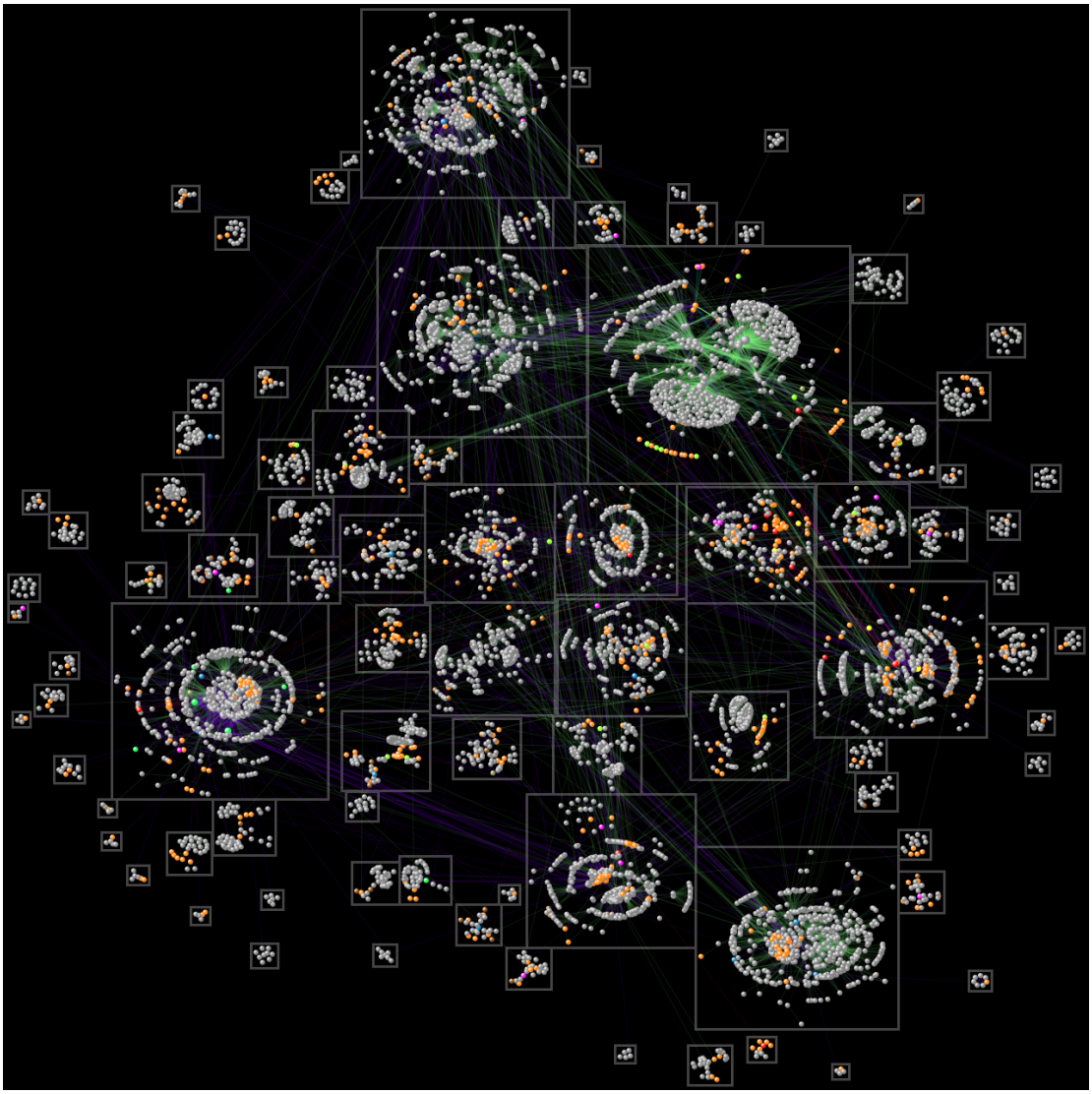


Figure 5.38: The visualization from Fig. 5.33, but using the Force-Directed GIB layout instead of the Treemap. Inter-group edges are visible and straight. All the groups have low aspect ratios, and aggregate connections between the large groups are more visible. The initial space-filling factor is 50%.

All told, Dempwolf found that these clusters accurately represented specific economic development opportunities that could be influenced to increase employment. According to him, "This approach gives you a list of firms to go talk to and specific things to talk with them about. It also identifies specific talent clusters. These are things that traditional industry cluster analysis has never done." More details of Dempwolf's use of NodeXL for identifying high-priority economic development targets are available in his slide deck[3], as well as his dissertation [Dem12].

5.5.3 Patient Discharge Summaries

I also applied the three Group-in-a-Box meta-layouts to the network of patients and concepts from their discharge reports, originally discussed in Section 4.3.5 as a case study for motif simplification (Chapter 4). After applying the Clauset-Newman-Moore topologic clustering algorithm [CNM04] to the network from Fig. 4.20, the standard color-coding approach produced the visualization shown in Fig. 5.39. The many densely connected clusters here are difficult to interpret. Note that standard clustering algorithms may not be as effective for analyzing networks with multiple node types, like this one of patients and concepts.

The Group-in-a-Box layouts, on the other hand, nicely segment these clusters. First, the Treemap GIB layout shown in Fig. 5.40 enables us to see the internal structure of each cluster. We have large clusters around our two egos in the net-

[3] http://portal.sliderocket.com/ATWBE/Using-SNA-to-find-and-manage-RICs

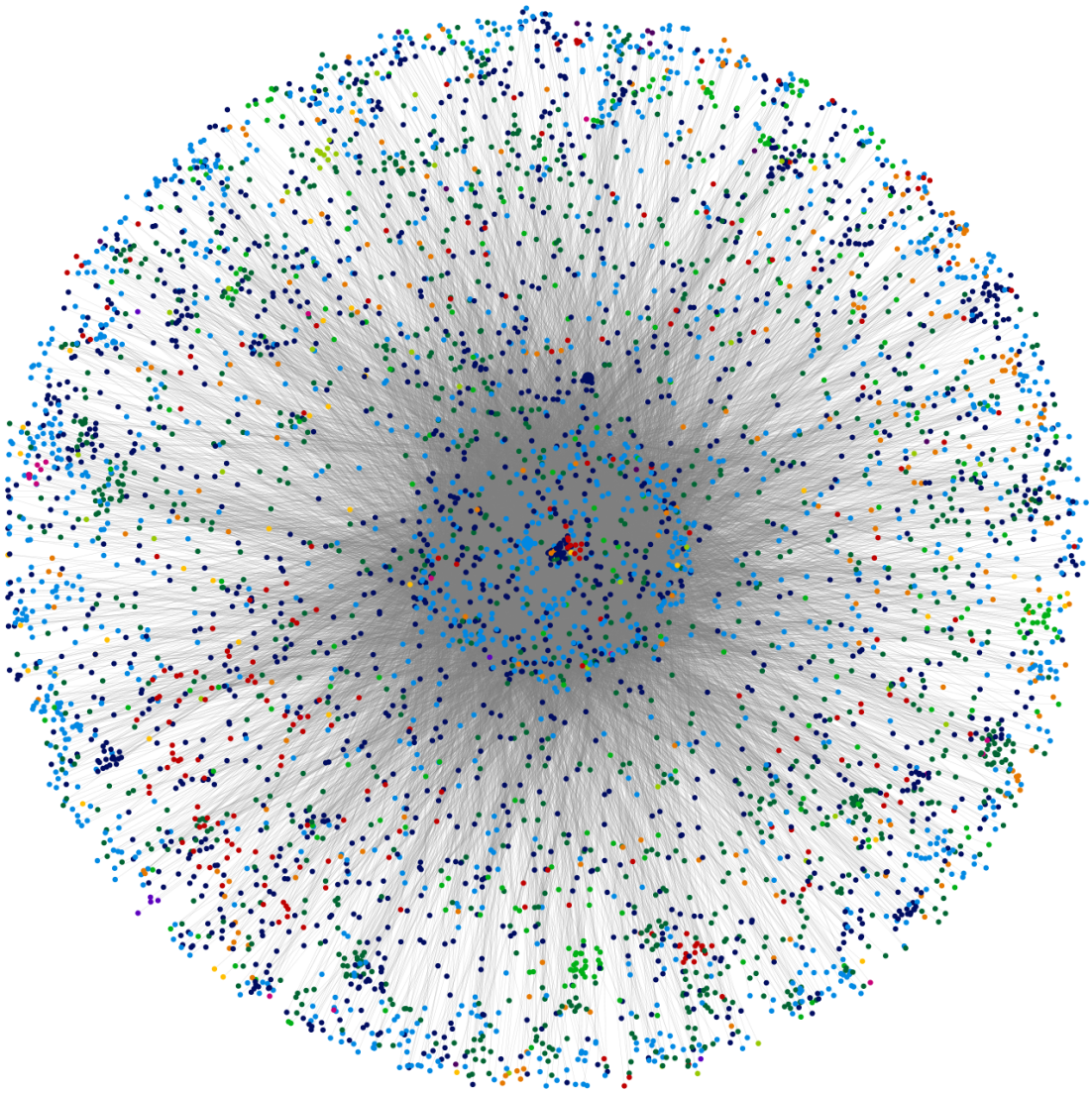


Figure 5.39: Patients and concepts related to the "hops5325" and "orch7323" medications from Fig. 4.20. Nodes are grouped using the Clauset-Newman-Moore topologic clustering algorithm [CNM04] and colored accordingly.

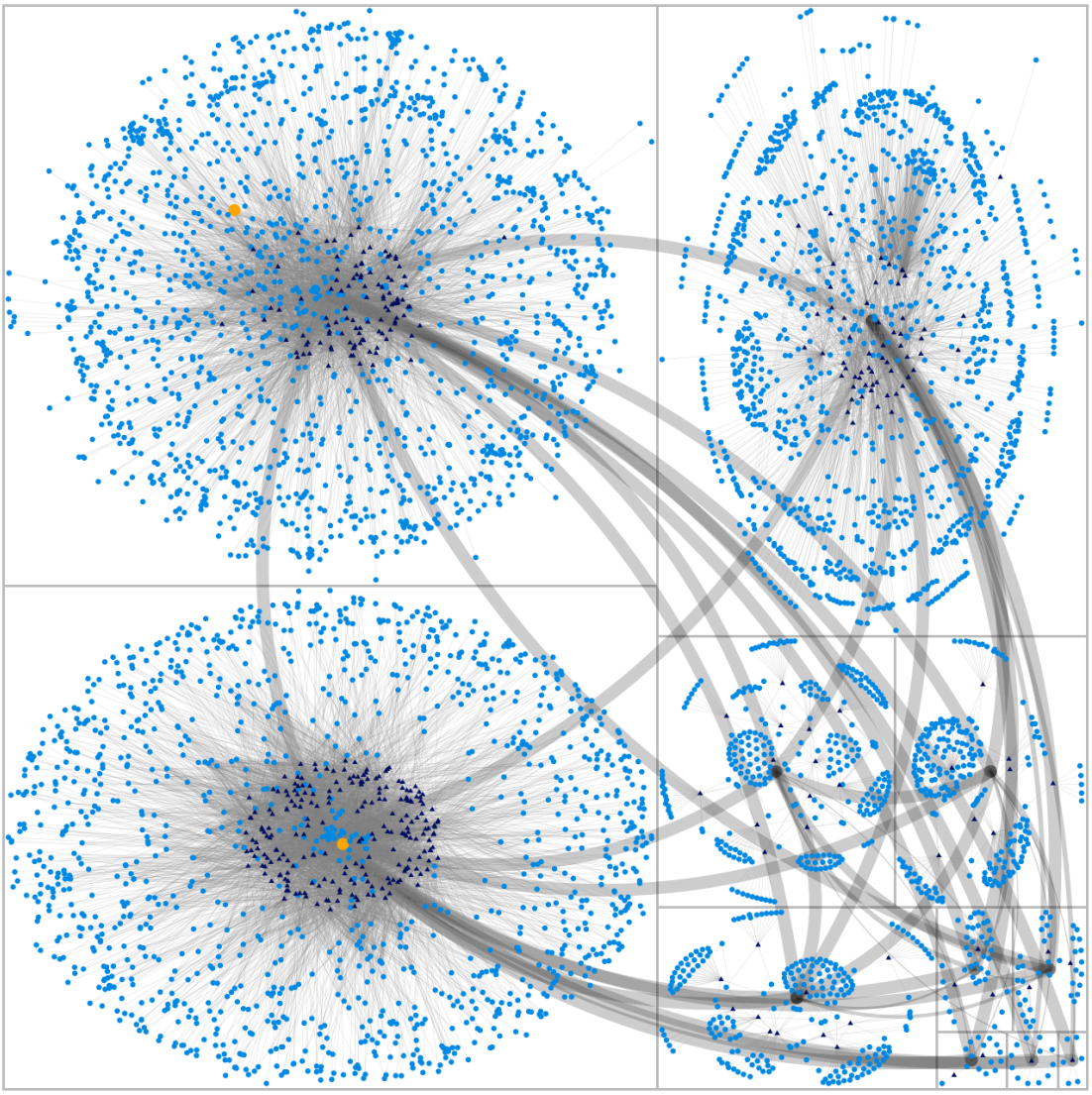


Figure 5.40: Patients, concepts, and clusters from Fig. 5.39, shown in the Treemap Group-in-a-Box layout. Our ego concepts, "hops5325" and "orch7323", are shown in orange in the largest clusters.

work, the concepts “hops5325” and “orch7323” shown in orange. There is another large cluster to the top-right, as well as several smaller ones. Each of these clusters consist of several patients and a range of concepts associated with them. However, the Treemap layout prevents us from seeing the ties between clusters easily.

The Croissant-Donut layout, in this case choosing the Croissant variant, is shown in Fig. 5.41. This layout does somewhat better at removing the overlap of the meta-edges between groups though has worse aspect ratios for the group boxes. The pure Force-Directed approach, shown in Fig. 5.42, does even better at showing the group ties and maintains square group boxes, though group internal structure is a bit less discernable than in the Treemap layout.

One interesting combination is to use one of the Group-in-a-Box layouts with the motif simplification techniques I presented in Chapter 4. I combined the node positions given by the Force-Directed GIB layout with the simplified motif glyphs, resulting in the visualization in Fig. 5.43. Due to technical limitations in the implementation, these approaches are not completely complimentary. For example, the edges between groups are shown and the group boxes have disappeared. However, the group and node positions are maintained. We can see which groups have large fan and connector motifs of similar concepts and could drill into them on a per-group basis. Future development, especially the inclusion of hierarchical or nested groups in NodeXL, could enable more effective combinations of these approaches.

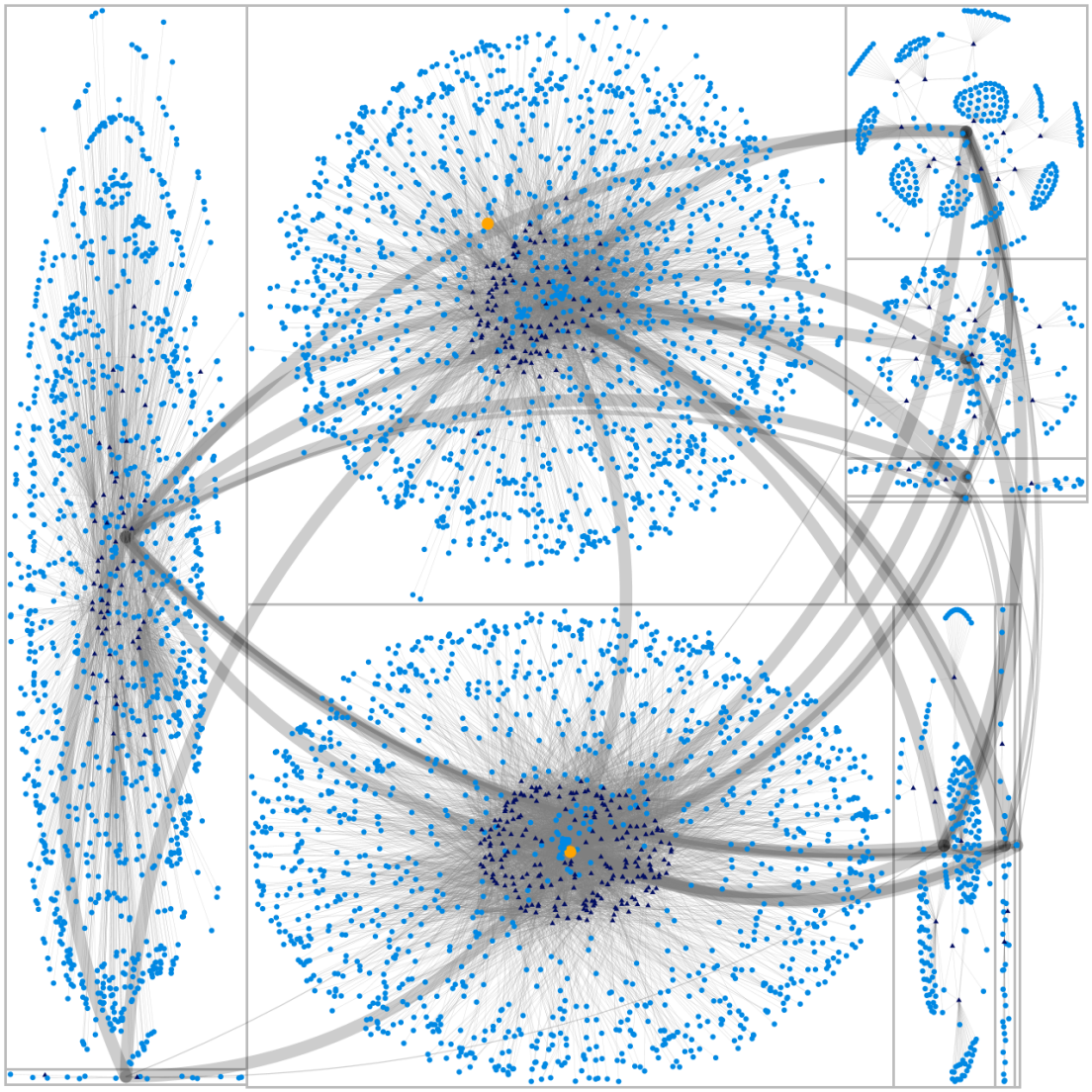


Figure 5.41: Patients, concepts, and clusters from Fig. 5.39, shown in the Croissant-Donut Group-in-a-Box layout. In this case the Croissant variant was chosen automatically. Our ego concepts, "hops5325" and "orch7323", are shown in orange in the largest clusters.

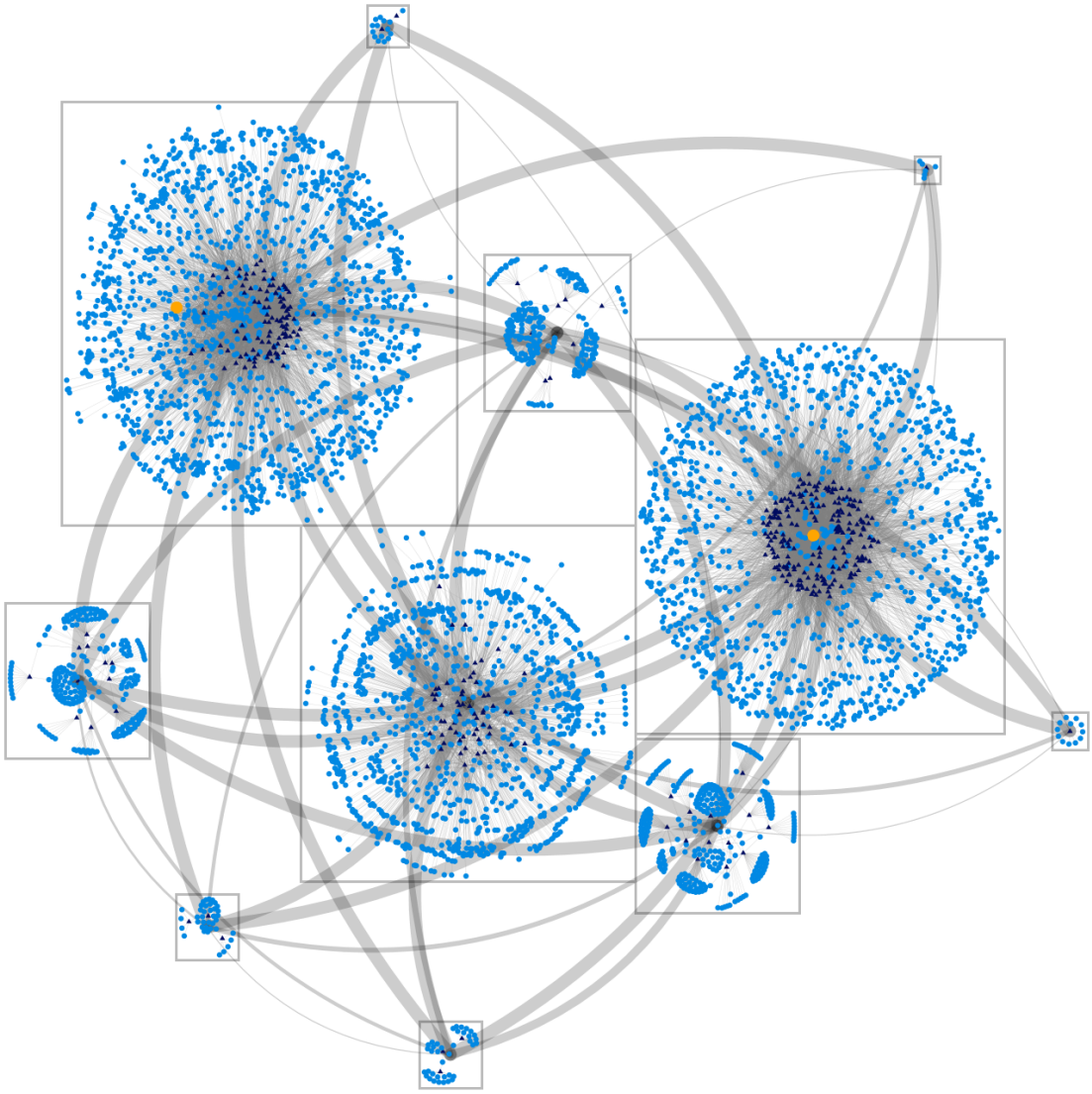


Figure 5.42: Patients, concepts, and clusters from Fig. 5.39, shown in the Force-Directed Group-in-a-Box layout. Our ego concepts, "hops5325" and "orch7323", are shown in orange in the largest clusters. The initial space-filling factor is 50%.

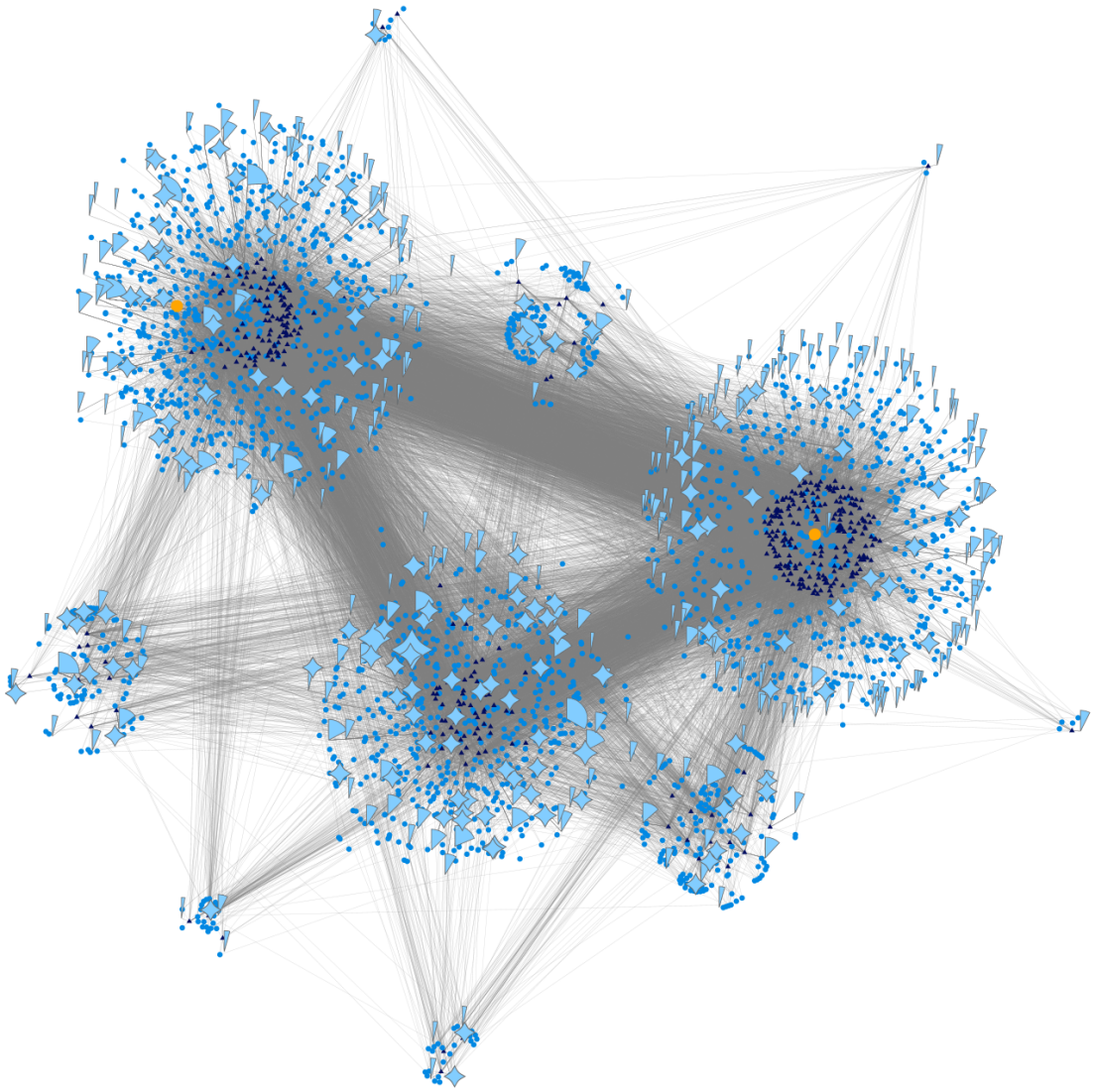


Figure 5.43: Patients, concepts, and clusters from Fig. 5.39, shown in the Force-Directed Group-in-a-Box layout but without the group boxes. The underlying edges are visible. The motif simplification technique from Chapter 4 is applied as well.

5.6 Experimental Results

In this section I compare the performance of the proposed Group-in-a-Box methods with the baseline ST-GIB on 309 Twitter networks downloaded from the NodeXL Graph Gallery [Smi+13]. I also describe an initial user study that was conducted to compare the usefulness of such GIB approaches. These studies were conducted by my students and me [Cha+13].

5.6.1 Pilot Study

The meta-layout methods proposed in this dissertation are based on the assumption that the existing ST-GIB layout is not good enough for understanding inter-group relations and that there is a need for methods that consider inter-group edges while arranging groups. To validate this hypothesis, we conducted an initial user study to compare the CD-GIB approach with the ST-GIB approach.

We recruited 9 participants who self reported that they have dealt with network data previously. The experiment followed a within subjects design, where subjects were given a set of tasks and asked to use the ST-GIB and the CD-GIB layouts to answer questions. The order of experimental conditions was counterbalanced by alternating the order in which the two layouts were presented. The tasks presented to the users were derived from questions that may arise about a network with regards to the relationship between the various groups. The tasks asked users to

count the number of outgoing combined edges from a list of groups, to find the group which had the maximum number of adjacent groups, to find the number of groups connected to a list of pair of groups, and to ascertain whether there was an edge between a series of pairs of groups. Following each task, users were asked to rate each layout on a scale from 0 to 9 based on the the layout's usefulness.

Based on this experiment, CD-GIB received an average score of 6.94 ± 1.47 and the ST-GIB layout received an average score of 4.61 ± 1.59. These results were encouraging as they demonstrated a need for better layout algorithms that would assist the user in understanding the relationships in a network better.

5.6.2 Readability Measures

Our initial evaluations of manually analyzing the results of the three algorithms were encouraging. However, for a more robust and formal evaluation, we quantify the usefulness of the a GIB method on the basis of the following network readability metrics. A good layout would occupy as much of the screen space as possible; have a Mean Group-Box Aspect Ratio close to 1.0 for a clearer intra-cluster visualization; and have a low Edge-Box-Overlap for a better inter-cluster visualization.

5.6.2.1 Edge-Box-Overlap(G)

Discernibility of inter-group edges depends on a number of factors. Most of these factors become critically important for particularly long edges which run from one

end of the screen to another. Visually following a long edge from source group-box to destination group-box can be cognitively challenging especially if the edge overlaps with several other group boxes 'on its way'.

Therefore, for a given inter-group edge, e, of a network or graph, G, we define the edge overlap, $Overlap(G, e)$, as the count of the number of group boxes (excluding the source and the destination group boxes) which intersect with the edge. A group box and an edge are said to be intersecting if the edge intersects with at least one of the four boundaries of the group box. For example, the edge overlap for the combined inter-group edge connecting G4 and G7 in Fig. 5.13 is 2 because it intersects boxes G1 and G2. Total Edge-Box-Overlap for the network, G, is then defined as:

$$\text{Edge-Box-Overlap(G)} = \frac{\sum_{e \in E} Overlap(G, e) \times w_e}{Max - Overlap(G)} \tag{5.1}$$

where

- E =Set of all inter-group edges in the network G
- w_e =Weight of an edge, e

When the inter-group edges are 'combined' in nature (see Section 5.4.4), we assume a straight line between the centers of the concerned groups and compute $Overlap(G, e)$ using this straight line to represent the inter-group edge, e. Edge-

Box-Overlap(G) is then computed by aggregating $Overlap(G, e)$ for all the combined inter-group edges in the network, G. Here, the weight of a combined inter-group edge is simply the sum of weights of the constituent inter-group edges.

For the sake of comparison, for a given network, we compute an upper bound to the Overlap, $Max - Overlap(G)$, as $IE \times (N - 2)$, where IE = Total number of inter-group edges times and N=Total number of groups in the network and use it to normalize the observed overlap.

5.6.2.2 Screen Space wasted

In the current problem setting, the size and shape of the screen, where the group boxes have to be arranged, is predetermined. The layout algorithms should, therefore, attempt to use as much of this space as possible. The space filling property is important because a layout which wastes more space basically assigns smaller areas to the group boxes (than a space-filling layout) and thus compromises on the clarity of intra-group cluster visualization. For example, the visualization in Fig. 5.30 is less space filling than that in Fig. 5.29. This happens because it assigns lesser screen area to group-boxes and so visualization of intra-group contents of, say EuroAfrica, is more difficult in Fig. 5.30 than in Fig. 5.29.

'Screen Space wasted' is defined as the percentage of screen space that was not occupied by any of the group boxes. Since the focus of this paper is arrangement of group boxes and not the nodes within the group, any white space inside a group

box is not considered as 'wasted'. Note, of course, that we are not truly wasting the space: we are often using to show aggregate topology.

5.6.2.3 Mean Group-Box Aspect Ratio

As mentioned earlier, thin elongated rectangles make analyzing their content difficult and so it is desirable for a GIB approach to produce 'squarified' group boxes that have aspect ratios closer to 1.0. Also, in the three GIB approaches compared here, a group's area is representative of its size. A typical user could exploit this property to compare group sizes based on their areas. Since visually comparing sizes of squares is easier than comparing sizes of rectangles, a better GIB algorithm should produce group boxes that are more 'square' in shape.

Given a clustered network laid out using a GIB approach, we measure this property using a mean of aspect ratios of the group boxes. Defining aspect ratio of a box as the ratio of its width and height, the Mean Group-Box Aspect Ratio can be expressed as:

$$\text{Mean Group-Box Aspect Ratio} = \frac{\sum_{i=1}^{N} a_i}{N} \tag{5.2}$$

where

- a_i = aspect ratio of the i^{th} group
- N = Total number of groups in the network

5.6.2.4 Time taken

This is defined as the time taken to layout the clustered network using the method under consideration, as determined by code surrounding the algorithm.

5.6.3 Dataset

We compared the performance of Squarified-Treemap, Croissant-Donut and Force-Directed GIBs on 309 Twitter networks. The networks each show the results of a search for tweets matching a certain word or hashtag. The nodes are Twitter users and the edges are created between any two users who mention, retweet, or reply to each other. These networks were collected by Marc Smith from Connected Action Consulting[4] and are published on the NodeXL Graph Gallery [Smi+13].

Table 5.1 describes some overall network metrics for the networks in our dataset. Since reporting values for individual networks was not feasible, I detail the mean, standard deviation, minimum, maximum and median values. All the networks were preprocessed to contain only the largest connected component, and the table reports its statistics. This was done to avoid the numerous uninteresting disconnected singleton groups that exist in many social network datasets. The networks were then clustered using the Clauset-Newman-Moore algorithm [CNM04].

[4] http://www.connectedaction.net/

Network Property	Mean±Standard Deviation	Minimum	Maximum	Median
Total number of Nodes	547.30±271.54	12.00	1462.00	541.00
Total number of Edges	7820.80±7982.11	30.00	40352.00	5438.00
Network Density ($\times 10^{-2}$)	1.25±1.24	0.07	9.04	0.83
Network Modularity	0.27±0.03	0.15	0.38	0.27
Average Geodesic Distance	3.04±0.69	1.72	7.31	3.00
Total number of Groups	11.38±5.42	2.00	30.00	10.00
Average Group size	52.52±35.34	4.00	236.50	43.38
Total number of inter-group edges	1630.83±2315.98	2.00	14858.00	898.00

Table 5.1: Overall network properties for the networks in our dataset.

Property/Measure	ST-GIB	CD-GIB	FD-GIB	CD-GIB Experiments	
				Donut always	Croissant always
Edge-Box-Overlap ($\times 10^{-2}$)	5.42	5.12	1.77	5.36	5.31
Screen Space Wasted	0.00	2.04	58.72	17.50	2.03
Time taken	811.00	744.00	951.00	765.00	739.00
Mean Group-Box Aspect Ratio	1.05	2.06	1.00	3.47	2.04

Table 5.2: Performance comparison of the two proposed approaches: CD-GIB and FD-GIB with the baseline ST-GIB layout. All figures reported above are median values computed for the complete dataset.

5.6.4 Results

Each network in our dataset, after clustering, were laid out using each of the three GIB layouts and the various performance measures described above were computed. The aspect ratio of the screen was kept at 1.0 for all experiments and the inter-group edges were combined. After arranging the boxes on the screen space, the nodes belonging to individual groups were laid out within the corresponding group box using the Harel and Koren FMS layout [HK01].

Table 5.2 presents the results of our experiments. For a given readability measure, the highlighed cells represent the best performing method among the three GIB approaches. The columns titled 'Donut always' and 'Croissant always' present intermediate results corresponding to the two layout possibilities for the CD-GIB method (Section 5.4.2). The actual results for the CD-GIB layout are listed in the column titled 'CD-GIB' after automatically selecting the appropriate layout as described in Section 5.4.5.3. I also performed Student's t-tests on these results and discuss the statistically significant ($p<0.01$) differences between treatments.

From Table 5.2, we can see that the FD-GIB leads to very little edge-box-overlap (1.77×10^{-2}) followed by CD-GIB (5.12×10^{-2}), while ST-GIB leads to maximum overlap of 5.42×10^{-2}. The statistical test revealed that the values obtained for FD-GIB were significantly different from others. However, the reduced overlap comes at the cost of an increased amount of space wasted. FD-GIB wastes

almost 59% of the screen space while ST-GIB wastes no space at all because of the use of highly space-filling treemap algorithm for laying out the group boxes. The space wasted by CD-GIB is 2% which is comparable to that of ST-GIB because like ST-GIB CD-GIB tries to 'pack' boxes next to each other. On the other hand, FD-GIB, lays out the boxes using one of the force-directed layouts which are not space-filling by nature. With regards to space wasted, all three methods were significantly different from each other according to the t-test results. The table also compares the three methods based on the time taken in milliseconds to lay out the complete network after clustering. We see that the time taken for all three methods are comparable with CD-GIB being the fastest (744ms); ST-GIB slightly slower with 811ms and FD-GIB taking 951ms. According to the t-test, the performance of FD-GIB was significantly different from others.

Finally, Table 5.2 compares the three methods based on the aspect ratio of their group boxes. Since each network contains several group boxes each with a different aspect ratio, we compute Mean Group-Box Aspect Ratio as defined above and compare the median values over the complete dataset. We see that the aspect ratio for ST-GIB and FD-GIB are almost 1.0 and the difference between them was not statistically significant. However, group boxes in CD-GIB approach suffer from poor aspect ratio (median value of 2.06) which was worse than that of FD-GIB and ST-GIB and this result was statistically significant. CD-GIB leads to poor

aspect ratio because unlike ST-GIB and FD-GIB, this approach does not try to produce squarified rectangles. It instead determines one dimension of the group boxes using the corresponding dimension of the free-space boxes (in which it is being placed) available around the 'donut hole' or the 'croissant hole'. Most of the free-space boxes are huge rectangles of white space. Hence, if the group contains small number of nodes, its area would be small, but one of its dimensions would be same as the dimension of the free-space box leading to thin elongated rectangles.

Table 5.2 also contains an intermediate result of comparing Donut and Croissant layouts on the same dataset. For this experiment, disregarding the paradigm presented in Section 5.4.5.3, all networks in the dataset were laid out using the Donut layout and the Croissant Layout separately. As seen from the table, the performances of Donut and Croissant are close in terms of overlap and time taken. Croissant outperforms Donut slightly in term of time taken while Donut beats Croissant for the overlap measure. For the other two measures, Croissant is statistically significantly better than Donut for screen wastage and group-box aspect ratio. However, comparing these columns with the 'CD-GIB' column, which is obtained by selecting either Donut or Croissant layout for each network based on our paradigm, we see that CD-GIB seems to be benefiting from the strength of both the alternatives. This justifies our use of the paradigm for Donut vs. Croissant selection heuristic.

5.7 Summary

This chapter discusses **meta-layouts**, which leverage disjoint node groupings in order to dissect a network into more manageable, yet meaningful subnetworks that are displayed individually. The first meta-layout, called the **Midichlorian-Directed Layout**, uses a standard force-directed layout algorithm that has been modified so that groups are less strongly attracted to each other. Thus, the groups in the network float apart and are more easily understood in isolation. However, this approach requires substantial screen space and in dense areas of the network groups can still overlap. This makes it difficult to measure group sizes and their aggregate relationships. Moreover, the high scaling required means that individual nodes are challenging to see, much less read the labels of.

To improve on this situation we developed three **Group-in-a-Box (GIB) layouts** that segment a network using the results of a topologic clustering or attribute grouping. Each group is laid out individually in a rectangular region of the screen, and we size each region according to the number of nodes it contains. The first layout, the **Treemap GIB layout** created by the NodeXL team [Rod+11], uses a squarified treemap algorithm [BHJVW00] to subdivide the screen space into group boxes with low aspect ratios, as shown in Fig. 5.9. The layout is completely space-filling, but can cause edge readability problems when large groups are positioned at opposite corners as in the innovation network in Fig. 5.33.

The second layout, the **Croissant-Donut GIB Layout**, maintains much of the space-filling property of the Treemap but can show the group relationships more clearly. It comes in two variants: the **Donut** and the **Croissant**. In the Donut variant, the most connected group is placed in the center of the visualization and other groups are wrapped around its periphery (Fig. 5.13). Alternatively, the Croissant variant puts the most connected group at the top and places the other groups around its three sides (Fig. 5.15). The Donut layout is more effective at showing many small groups, while the Croissant is better for a few large groups. Our code chooses which of the two to use automatically depending on the distribution of group sizes. The Croissant-Donut layouts fill most of the visualization space while showing relationships more clearly, but aspect ratios can get especially high for small groups.

Finally, I developed a **Force-Directed GIB Layout** that positions groups according to their aggregate relationships, followed by an overlap removal step that ensures the boxes do not intersect (Fig. 5.30). The overlap removal algorithm maintains the relative positions of groups while minimizing the additional space required. The resulting visualization requires a substantial amount of screen space, but uses the extra space to clearly show the relationships between groups. At the same time, the low aspect ratios of the group boxes helps offset their smaller size for showing internal group structure.

We have also developed a few ways to automatically choose which to use depending on the network and group properties. By nesting the Group-in-a-Box layouts I can handle disconnected components better than other approaches. Moreover, for certain numbers of groups and distributions of group sizes I pick the best layout for the user. Finally, I present several case studies and an experimental study to help validate the effectiveness of these layout techniques.

Each of these Group-in-a-Box layouts have been implemented and made publicly available in NodeXL [Smi+10]. While the Croissant-Donut and Force-Directed GIB approaches have only recently been added, the Treemap layout has been available since 2010 and is used extensively by users. Looking at the NodeXL Graph Gallery [Smi+13], most of the visualizations presented and almost all of those by Marc Smith (from Connected Action Consulting and leader of the NodeXL project) use the Treemap GIB layout. Dr. Smith intends to transition to the Force-Directed approach immediately for his work. This demonstrates the utility of these techniques for segmenting real networks into manageable, meaningful pieces – especially in web environments where display space is limited and overviews are particularly useful. Moreover, the improved defaults for placing disconnected components will help all users of NodeXL, which has been downloaded more than 166,000 times and is used extensively for introductory network analysis courses.

Chapter 6

Measuring Network Visualization Readability

6.1 Introduction

The results of applying force-directed layout algorithms can vary greatly depending on the size and topology of the network, and the layout generated is highly dependent on the algorithm used. Each algorithm attempts to find an optimal layout of the network, often according to a set of **readability metrics (RMs)** or heuristics. Readability metrics are measures of how understandable the network drawing is, based on artifacts such as the number of edge crossings or overlapping nodes in the drawing [DS09]. Traditionally these RMs have been called **aesthetic criteria** [PL96; Pur02], though several recent papers describe network visualizations in terms of readability instead of aesthetics ([GFC04; HBF08; Bon+09]). I call them readability metrics because of the ambiguity implied by the word "aesthetic". I am not concerned as much with how visually pleasing a particular network drawing is; instead I am interested in how well it communicates the underlying data. However, some of the most informative visualizations are also the most beautiful.

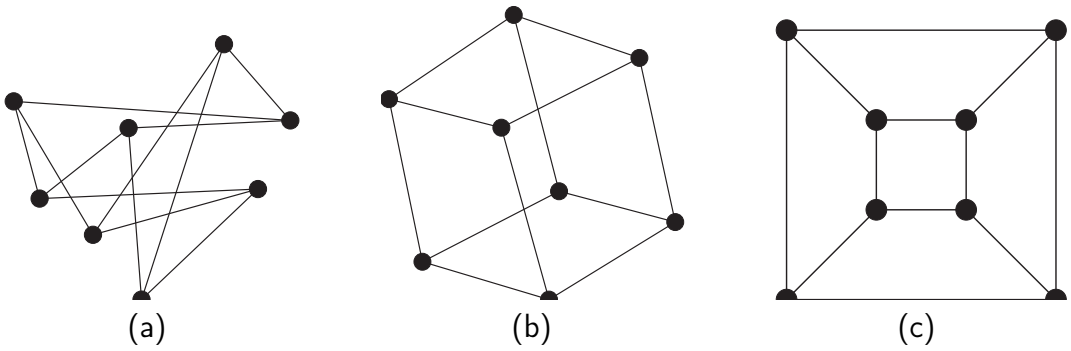


Figure 6.1: Different visualizations of the same network with many (a), few (b), and no (c) edge crossings.

Optimizing the layout for specific readability metrics, or RMs, can lead to much more understandable drawings. For example, Figs. 6.1 and 6.2 show how reducing edge crossings can lead to more straightforward representations. Optimizing for RMs has been shown to promote many common analysis tasks, though it does not guarantee the resulting drawing is understandable. The particular RMs that the layout algorithms optimize intentionally or indirectly through heuristics may not be the correct ones for the tasks users are trying to accomplish. There are often substantial trade-offs in task performance when different RMs are optimized, and can result in ineffective, unintelligible, or even misleading drawings. For example, after reducing the number of edge crossings in a large drawing the spatial layout is oftentimes substantially distorted, and it can alter a viewer's perception of the importance and centrality of individual nodes (see Section 6.2 and Fig. 6.6d for an example of this effect).

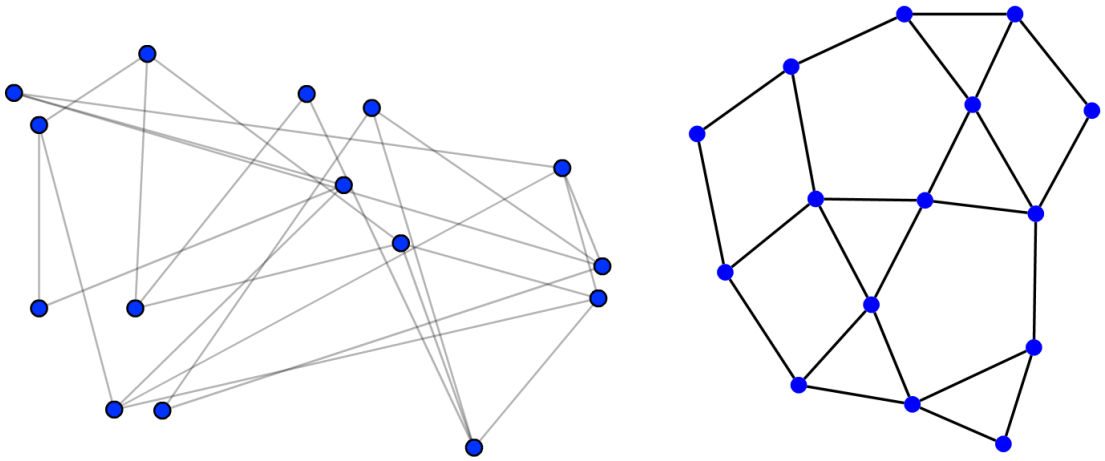


Figure 6.2: In the Planarity online game (www.planarity.net), users start with a planar network: one that can be embedded in two dimensions using straight edges with no crossings. Given a random network layout like (a) users try to manually eliminate crossings. The goal is to create a planar drawing like (b), which is the same network run through NodeXL's [Smi+10] Harel-Koren FMS layout [HK02a].

Additionally, as the optimization of many RMs is NP-hard [Bat+98], these techniques often produce suboptimal network drawings. The International Symposium on Graph Drawing has met annually for two decades working to improve automated network layout algorithms and RMs, among other things, but I believe that state of the art automated layout algorithms alone are insufficient to consistently produce understandable network drawings. Additional post-processing algorithms can improve the layout, but are limited in how much they can modify the layout. The layout algorithms available to end users depends on the network analysis tool being used, and post-processing techniques are rarely included and have difficulties with evolving networks.

Users can be made aware of the common problems RMs measure, or even quantitative values for RMs to optimize manually. However, current RMs only provide overall measures for the drawing without any means for focusing user attention on the problem areas. Users are not provided with any indication of where to start their manual improvements and how effective they have been. Seasoned network analysts develop an ingrained understanding of proper layout techniques and will adjust the spatial layout accordingly, but novice users are left to fend for themselves. Even expert users have difficulty applying their layout techniques to networks over a few hundred nodes. Furthermore, users may not be aware of the optimization trade-offs of particular metrics and how it affects task performance.

Part of my dissertation work was to develop new readability metrics to measure the effectiveness of node-link visualizations, including a set of novel **node & edge readability metrics** that provide more localized identification of where improvement is needed. As there are trade-offs when optimizing readability metrics, I provide a survey of the related literature studying these trade-offs and the effect of specific metrics on user task performance. I also provide the design and implementation of an interactive optimization technique that provides users with visual metric feedback, helping them optimizing their drawings. This work aims to raise user awareness of network visualization readability issues, and applying these techniques will guide users in creating more effective node-link visualizations.

Instead of focusing only on purely automated network layout, I advocate raising user awareness of the importance of readability metrics for their network drawings and providing users with computer-assisted layout manipulation tools. Taking up where the automated layout leaves off, my tool gives users real-time feedback as to how their movement of nodes affect the RMs and provide local placement suggestions for the RMs users wish to optimize. I believe that this approach will provide users, and network analysts in particular, tools and guidelines that will allow them to create more understandable network drawings that more accurately highlight features of interest like communities within social networks.

To enable this I detail several new readability metrics on a [0,1] continuous scale. Additionally, I define novel **node & edge readability metrics** to provide more localized identification of where improvement is needed. The metrics can be used by a user to motivate improvement of the network drawing, either by hand, through immediate feedback techniques, or automatic improvement by feeding RM results back into a layout algorithm. I describe the trade-offs inherent in optimizing individual metrics as well as recommended metric optimizations for particular tasks. Several of the RMs and the interactive improvement techniques are implemented in SocialAction, a research network analysis tool that combines statistics with network analysis [PS06; PS08a; PS08b]. I have also begun integrating the metrics and improvement technique in NodeXL [Smi+10], a network

analysis template for Excel 2007/2010/2013, in order to direct users towards poor areas of the drawing and provide real-time readability metric feedback as users manipulate nodes and edges. The interaction functionality includes ranking and highlighting of nodes and edges by their metrics.

6.1.1 Chapter Overview

Specifically, the contributions of this chapter are:

- New global readability metrics to help understand different aspects of network visualization readability,
- Local readability metrics for individual nodes and edges to help users identify problem areas and fix them,
- A method for user-assisted layout improvement that provides real-time metric feedback to users in a ranked list and with a color scale,
- Implementations of readability metrics and the layout improvement technique in SocialAction and NodeXL, and
- A survey of work on readability metrics and evaluations of their effectiveness on various network analysis tasks.

This chapter is divided into several sections as follows. First, I describe the idea behind the user-assisted layout improvement technique and the SocialAction

implementation in Section 6.2. This includes two case studies of the effectiveness of the approach. Next, I cover the NodeXL implementation in Section 6.3. Then I go into detail about specific readability metrics in Section 6.4 including a survey of their history and evaluations of their effectiveness. Finally I conclude in Section 6.5.

6.2 Readability Metrics in SocialAction

Several readability metrics (RMs) exist that measure the suitability of a network drawing as a whole, providing a single quantitative measure for the entire drawing. While these metrics can aid users in understanding that there is a problem, they do not highlight where the problems are occurring. To do so, we can provide additional attributes for both nodes and edges in the network that describe how these individual components affect the global understanding. I call these **node readability metrics** and **edge readability metrics**, or node RMs and edge RMs for short. This is an extension of the idea of individual node and edge metrics espoused in [HMM00]. Several of my metrics are detailed in Section 6.4, along with their individual motivations, including: **node-node overlap**, **edge crossing**, and **node-edge overlap**.

I have implemented a prototype of the RM framework inside of SocialAction, a tool that uses attribute ranking and multiple coordinated views to help users systematically explore various statistical measures for social network analysis [PS06;

PS08a; PS08b]. In SocialAction, users can rank nodes and edges using ordered lists of the chosen attribute and simultaneously visually code the node-edge drawing using the ranking. Nodes remain in their original positions as users change the ranked attributes, which prevents the users from losing their mental map of the network. By combining multiple coordinated views with rapid transitions between statistical social network analysis measures and additional node and edge attribute rankings, SocialAction affords network analysts a quick understanding of the network properties. Extreme-valued nodes and edges are highlighted particularly effectively through the combination of ranked lists and visual coding.

I leveraged this attribute ranking system by incorporating preliminary node and edge RMs into SocialAction as node and edge attributes. Like any statistical measure or additional attributes in the dataset, users can now rank nodes and edges based on their individual RMs, highlighting problem areas in the network drawing. This allows them to rapidly flip between RM rankings and identify areas that would benefit from hand-tuning of the layout.

Users can then utilize the interactive features of SocialAction which allow them to drag nodes or groups of nodes to new positions, attempting to manually optimize the RMs. Node and edge RMs are computed in real-time for the nodes being dragged, and many global RMs can be selectively updated with these local computations to shortcut the computational complexity a complete recalculation

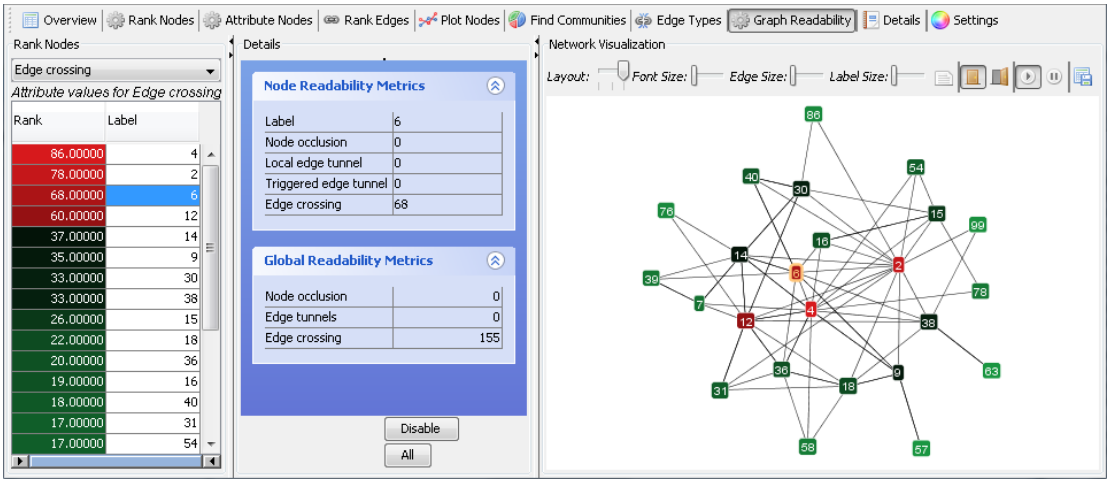


Figure 6.3: SocialAction with the integrated Network Drawing Readability Metric framework rapidly shows problem areas in the network drawing highlighted in red and listed in a ranked table. It is currently showing a subset of the reply relationships within the Alberta Politics discussion newsgroup, and the network drawing has been optimized for the node occlusion and edge tunnel readability metrics. The steps in SocialAction's Systematic Yet Flexible framework are shown along the top. The Network Readability panel (middle-left) shows node or edge readability metrics as well as global ones. The Rank Nodes panel at the far left ranks nodes by the edge crossing readability metric and provides the color scale for the Network pane.

requires. This allows users to see how their movement of nodes affects both global and node RMs simultaneously, both in a Network Readability panel as well as real-time updating of the ranked list and color scale of the node-edge drawing. Moreover, users can switch between individual RMs and statistical measures while maintaining the same network layout and preserving any hand tuning they have already accomplished.

Fig. 6.3 shows the SocialAction interface displaying a node-link visualization of reply relationships within a subset the Alberta Politics discussion newsgroup for

which the node occlusion and edge tunnel readability metrics have been minimized. Across the top are the steps in SocialAction's Systematic Yet Flexible framework, which allows for a guided and all-encompassing while still flexible approach to social network analysis, along with the Attribute Nodes panel for categorical coloring and the Network Readability panel (shown along the middle-left). The Network Readability panel shows the node or edge readability metrics for the selected items, as well as global readability metrics. The Rank Nodes panel (far left) shows a ranking of nodes by the edge crossing readability metric in decreasing order, with a filtering slider at the bottom. The large Network panel shows the node-edge drawing with color coding of nodes by their ranking in the Rank Nodes panel, with nodes having many edge crossings colored bright red. These are candidates for movement or resizing to reduce the number of edge crossings.

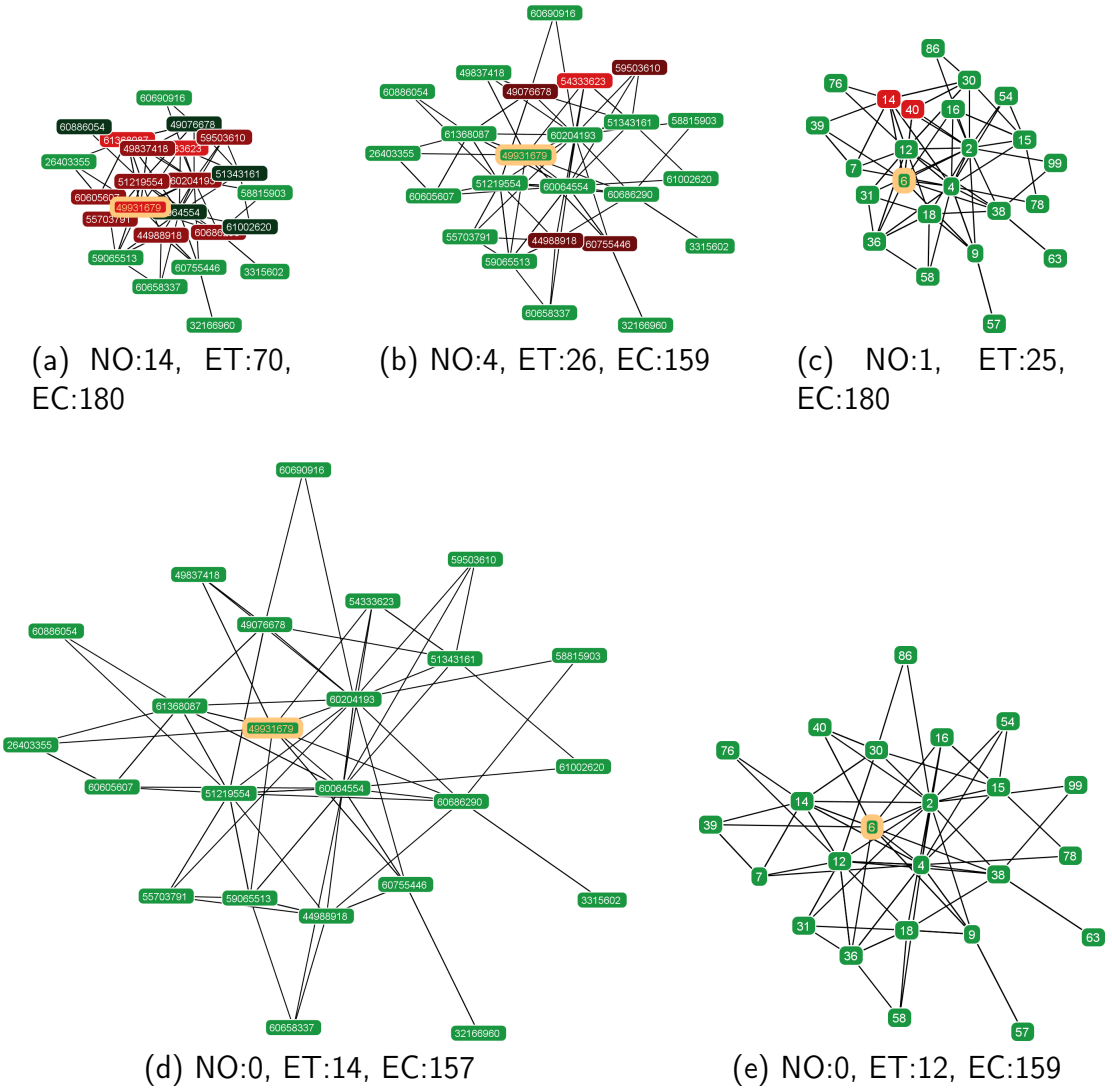
6.2.1 Case Study: Alberta Politics Newsgroup

The following figures demonstrate manual optimization of a network drawing. Underneath each figure are counts for the number of node occlusions (NO), edge tunnels (ET), and edge crossings (EC). Counts can usually be made available as tooltips, but for the RMs to be useful they must be independent of the network size, and are thus scaled to the continuous range from [0,1]. This requirement is made evident from the global count of 2954 edge crossings in the Alberta Politics

discussion group network. Also note that figures which show a progression of drawings being optimized for a RM may change color scale, as the worst nodes become better. This relative scale is better at highlighting maximal existing metric values.

Users can manipulate their drawings in order to minimize node occlusion using the node RM for it as a guide (Fig. 6.4). Coloring is scaled by the node RM, with bright red drawing user attention to areas of high occlusion. By relaxing the layout slider in SocialAction we can eliminate node occlusion entirely for this subset of the Alberta Politics dataset (Figs. 6.4a, 6.4b and 6.4d). This increases the default spring length used by the layout algorithm, allowing clusters of nodes to spread out and resulting in a larger drawing. Some networks, especially dense ones, may require manual tweaking. Another way to minimize occlusion is to reduce the size of labels. One way is to move from a full label to a distinctive yet concise one (Figs. 6.4c and 6.4e, though numeric ones are difficult to remember). Other ways include minimizing text margins in the nodes or font size.

To reduce the number of edge tunnels in the drawing, users can rank and color by the node RM for local edge tunnels. Figs. 6.5a and 6.5b show a user removing edge tunnels by tuning node placement. This is easier for loosely connected nodes but can be difficult in dense areas. To reduce edge tunnels, we may have to increase the number of edge crossings. For manually tweaking the position of poorly connected nodes the local edge tunnel RM seems more useful. However,



(a) NO:14, ET:70, EC:180

(b) NO:4, ET:26, EC:159

(c) NO:1, ET:25, EC:180

(d) NO:0, ET:14, EC:157

(e) NO:0, ET:12, EC:159

Figure 6.4: Ranking and coloring with the node occlusion node RM shows areas of high occlusion in red. To reduce occlusion we can relax the layout by increasing default spring lengths ((a), (b), (d)). Note that this is not the same as merely increasing the size of the drawing: the adjustment of the parameters of the layout algorithm results in a somewhat different layout as well. We can also use shorter unique, trimmed, or simplified labels ((c) & (e)), in addition to hand-tuning node position as a final step. Note that color scales may change between figures as the worst nodes become better. Counts listed are node occlusion (NO), edge tunnels (ET), and edge crossings (EC).

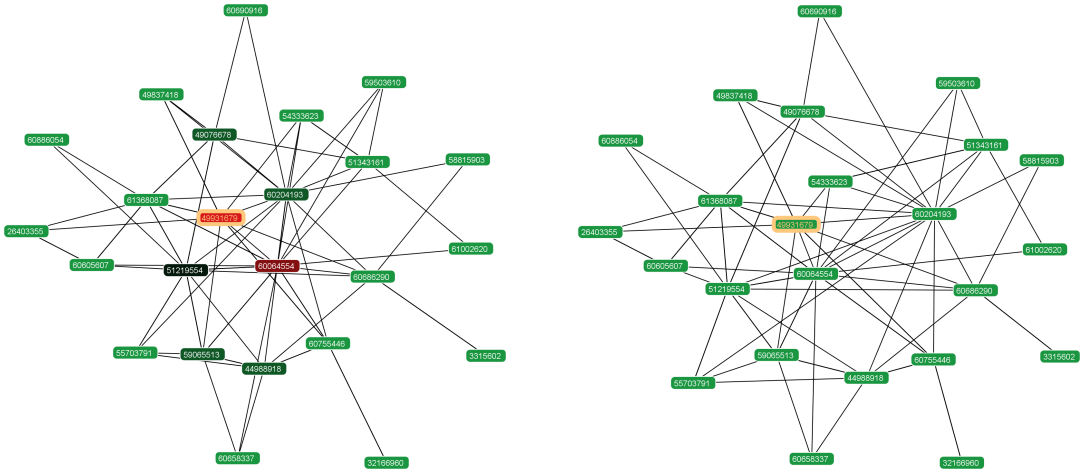
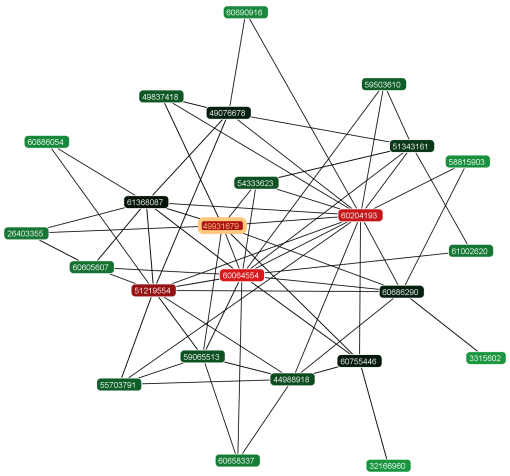


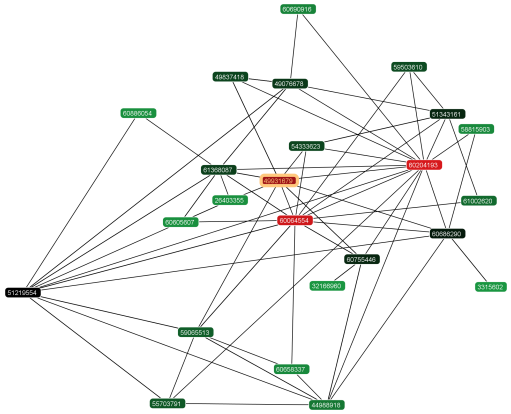
Figure 6.5: Using the node RM for edge tunnels, users can see areas with edge tunnels in red (a) and manually adjust the layout to remove them (b).

the triggered edge tunnel RM is better suited for moving highly connected nodes as it shows the effect a node has on its region of the drawing. As with node occlusion, one way of reducing edge tunnels is to shrink nodes.

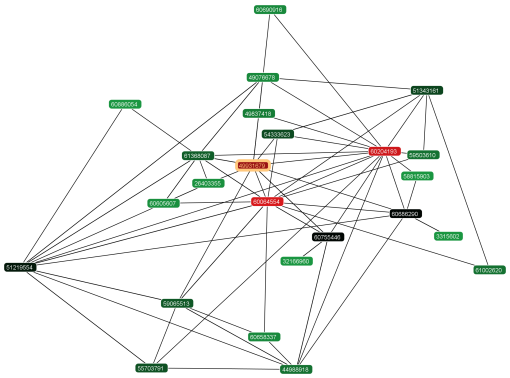
Similarly, Figs. 6.6b to 6.6d show a user removing edge crossings using the node RM for it. This is often a harder RM to minimize, as it is not always obvious how moving a node will eventually affect the total count. The process often involves trial and error, as well as multiple passes through each region of the drawing. Moreover, most social networks are not planar networks and can't be represented without edge crossings. One of the easiest approaches is to pull tightly connected nodes near the edge farther out as in Fig. 6.6c, so that less central nodes can be placed between its connected edges. This has the unfortunate effect of significantly



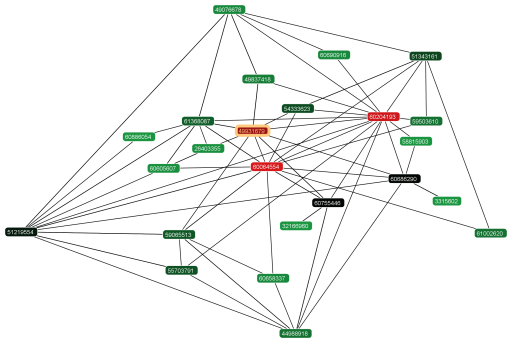
(a) Edge crossing NO:0,ET:0,EC:155



(b) Edge crossings removed (1/3) NO:0,ET:0,EC:114



(c) Edge crossings removed (2/3) NO:0,ET:0,EC:90



(d) Edge crossings removed (3/3) NO:0,ET:0,EC:85

Figure 6.6: Likewise, the node RM for edge crossings shows users areas with lots of crossings (a) and lets them hand tune the layout to reduce them ((b)–(d)). Fig. 6.1 gives a prime example for how minimizing edge crossings can greatly improve the readability of a drawing. Unfortunately, minimizing the number of edge crossings for less structured networks often results in an asymmetric drawing like (d) in which the centrality and angular resolution of many nodes is reduced, decreasing their perceived importance. For larger, less structured networks a balance must be struck between the number of edge crossings and the impact of further minimization on the spatial layout of the drawing. Note that color scales may change between figures as the worst nodes become better. Metrics listed are node occlusion (NO), edge tunnels (ET), and edge crossings (EC).

worsening the angular resolution and spatial layout RMs, which can make the node seem less important or central than it is.

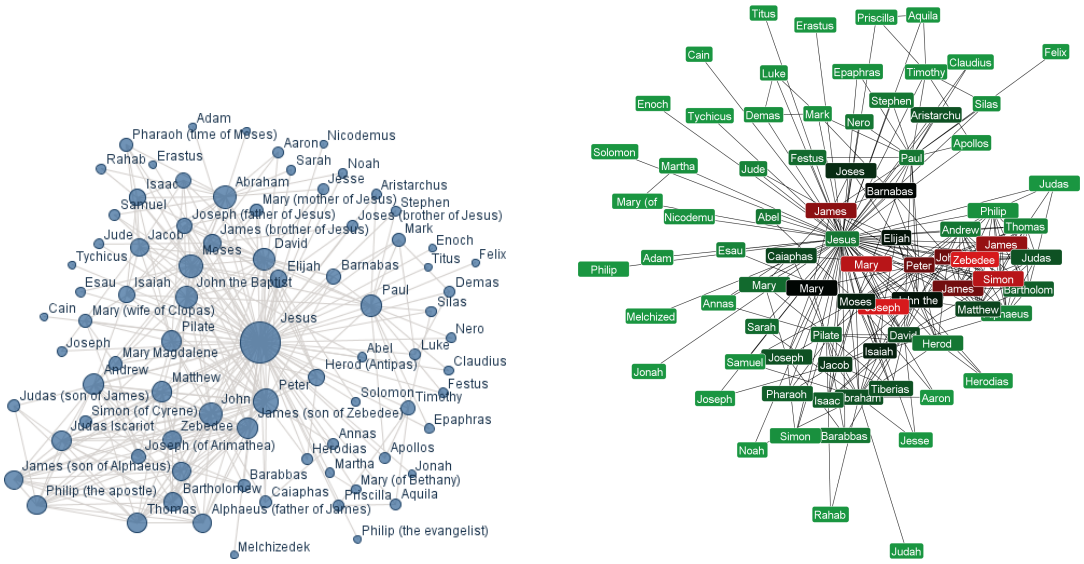
Improving individual RMs can be beneficial for other RMs as well, though often there are tradeoffs between them users may have to weigh. Which RMs should be improved thus depends on what users are trying to convey with their drawings. Thus, it is imperative that users of network drawing software be made aware of which RMs their layout algorithms attempt to optimize and the effects various layout techniques have on how much of the underlying data is effectively conveyed.

6.2.2 Case Study: New Testament Name Co-Occurrence

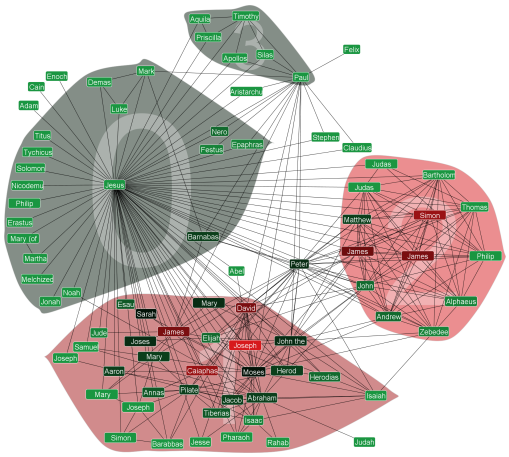
In 2008 The New York Times published a node-link visualization of the co-occurrence of names appearing in the New Testament,[1] shown in Fig. 6.7a. It used a force-directed layout drawn by IBM's ManyEyes tool.[2] While interesting, I believed that the drawing had substantial readability problems that could be improved by using my metrics.

After loading the same dataset into SocialAction, the default force-directed layout rendered a quite similar drawing (Fig. 6.7b). After applying topological clustering using Newman's fast heuristic [New04] and showing the clusters using convex hulls, much of the underlying structure could be discerned. I further im-

[1] http://www.nytimes.com/imagepages/2008/08/31/business/31novelCA02ready.html
[2] http://www-958.ibm.com/software/data/cognos/manyeyes/



(a) (b) NO:23,ET:283,EC:2104



(c) NO:0,ET:154,EC:2032

Figure 6.7: Name co-appearance network from the New Testament. (a) is the original New York Times/ManyEyes visualization, while (b) shows the same network in SocialAction [PS06]. (c) shows the clusters found by Newman's fast heuristic [New04] using convex hulls, and I optimized the layout using the node-node overlap and edge crossing metrics.

proved on this layout by optimizing for the node-node overlap and edge crossing metrics, resulting in the drawing in Fig. 6.7c.

One advantage of this new drawing (Fig. 6.7c) is that the separate clusters of individuals are much easier to discern than in the original drawing. It is also much easier to understand pivotal relationships that bridge the groups, like Peter. Moreover, there are no overlapping labels, though the zoom is lower. The main disadvantage of this drawing is in the kind of visceral reaction people may have to the movement of the Jesus node towards the periphery, with its group of connected singletons in the top left. Studies have shown that reducing the angular resolution of high-importance nodes like Jesus do not significantly impact task performance, however these kinds of modifications can substantially impact user perception of less important nodes.

6.3 Readability Metrics in NodeXL

I have begun implementing the readability metrics and automatic improvement technique inside NodeXL [Smi+10]. Fig. 6.8 shows the NodeXL interface with the readability metrics dialog in the foreground. The dialog allows the user to select which global, node, and edge metrics to calculate. Then the user can calculate the metrics on demand and optionally have NodeXL continue updating the metrics incrementally as the user manipulates the node-link visualization in the graph

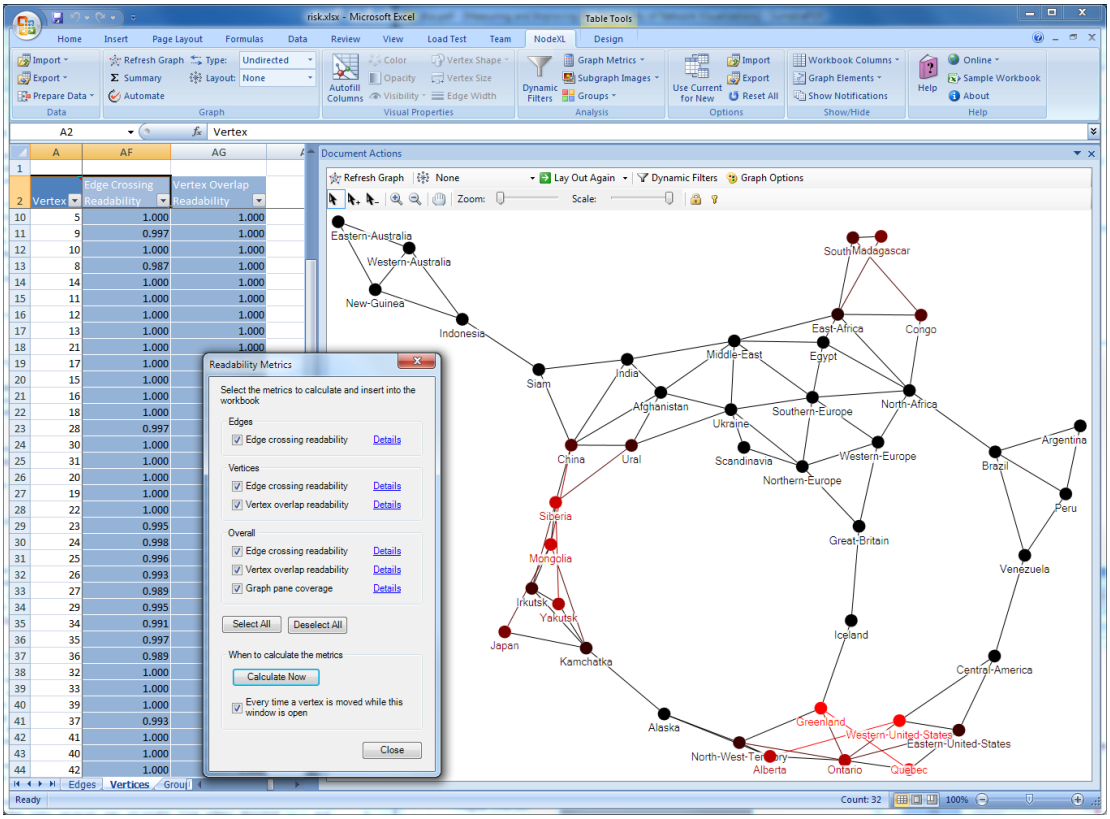


Figure 6.8: NodeXL showing the readability metrics dialog box (foreground), the nodes in the worksheet with their associated edge crossing and node overlap metric columns, and the graph pane where nodes and edges are colored by the edge crossing metric on a red-black scale. Nodes causing the most edge crossings are colored in bright red, as are edges with the most crossings. The network shown represents the legal moves in the board game Risk (see Section 5.5.1 for details).

pane on the right. On the left side we can see the node worksheet, which has two additional columns populated for the calculated edge crossing and node overlap metrics. In this case, the edge crossing metric column has been used to color the nodes on a red-black scale to highlight nodes that cause edge crossing problems. Similarly, the edge worksheet (not visible) has column for edge crossings as well which was used to color the edges in the node-link visualization. With these tools

the user can immediately find the problem areas and make manual improvement with real-time color feedback.

6.4 Specific Readability Metrics

This section discusses several specific readability metrics (RMs), including the motivation for their use and the formulas I have created to quantify them. For more background and an introduction to my approach, see Sections 6.1 and 6.2. The following sections each deal with a specific metric I considered, and Section 6.4.18 gives a brief overview of additional RMs that I have not yet implemented but appear valuable.

As per [Pur02], each RM is scaled appropriately to a continuous scale from [0,1] where 1 indicates the positive maximum of the RM. This allows us to assign graph readability requirements to particular drawings based on the content and information we want the impart. For example, a journal may recommend 0% node occlusion, <2% edge tunneling, and <5% edge crossing to publish a node-link visualization, while having different suggestions for UML diagrams or other kinds of graphs. However, there are many useful graph drawings that violate these limits and they should not be eliminated based solely on the RMs.

In these formulas I use a notation similar to that of [Pur02], where the graph has n nodes and m edges, indexed using subscripts. Using a technique called **bends**

promotion [Pur02], we can convert a polyline edge into several new straight line edges denoted m' and replace the bends in the edges with new nodes denoted n'.

6.4.1 Node-Node Overlap $\aleph_n$

Euclid defined a point as that which has no part. Historically, graph layout algorithms were designed around these **abstract graphs** [LE02], with nodes taking up little or no space [WS79; Mis+95; LEN05]. However, **practical graphs** like sociograms or UML diagrams represent nodes using text, shapes, colors, pictures, and size [LE02]. Classical algorithms can thus frequently result in nodes with non-zero width and height overlapping one another in the graph drawing.

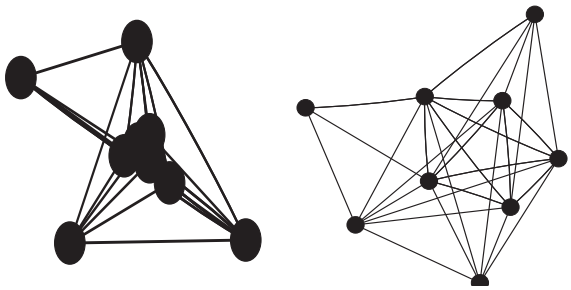
This **node-node overlap**, also called overplotting, is contrary to accepted graph readability guidelines [Sug02], including those for trees [WS79] and UML diagrams [Eic03]. Moreover, areas of the drawing with high occlusion make it very difficult for the viewer to get an accurate count of the number of individual nodes in a cluster to get a sense of its scale. These problems can be reduced somewhat, but not entirely, through the use of a halo or fog effect around nodes to help distinguish them from each other.

Many force-directed layout algorithms include node-node repulsive forces or equivalent constructs, including variants of the spring embedder [Ead84] such the popular Fruchterman-Reingold force-directed algorithm [FR91] and more scal-

able gravitational N-Body approaches like provided by Prefuse [HCL05] using the Barnes-Hut force calculation algorithm [BH86]. However, force-directed approaches cannot usually guarantee all overlaps will be removed while the area and shape of the drawing are preserved because they rely on overly large repulsive forces or post-processing [GH09]. One notable exception is [HK02b].

There have also been many algorithms developed for removing node-node overlaps using post-processing after an initial layout algorithm. These include variants of the force-scan method [EL92; Mis+95; LE02; Hay+02; HL03; LEN05], constrained optimization [Mar+03; DMS06; DMS07], and force-directed approaches [LMR98; GN98; Hua+07]. One of the most effective approaches appears to be the PRoxImity Stress Model (PRISM) algorithm of Gansner and Hu [GH09], which is discussed in detail in Section 5.4.3.3 in the context of removing group box overlap in a Group-in-a-Box layout and compared to Dwyer, Marriott, and Stuckey's solve_VPSC algorithm [DMS06; DMS07].

One option proposed by Li, Eades, and Nikolov [LEN05] is varying the edge lengths in a standard force-directed layout. While this preserves the orthogonal ordering well, it has scaling issues and can require excessive space [GH09]. An alternative is the Voronoi cluster busting algorithm of Lyons, Meijer, and Rappaport [LMR98] and used by Gansner and North [GN98] for their layout. This algorithm iteratively forms a Voronoi diagram for the layout and moves nodes to the center



(a) Tight layout (b) Relaxed layout

Figure 6.9: We can eliminate the node occlusion that makes the central overlapping group in Fig. 6.9a so hard to understand by zooming out and increasing the the spring lengths of the layout algorithm (Fig. 6.9b).

of their Voronoi cells. This roughly maintains the network shape, but loses much of the layout structure and again expands to take up a lot of screen space [GH09]. Another interesting approach by Imamichi et al. [Ima+09] for 3D visualizations assumes labels extend from spherical nodes, models these masses with a set of spheres, and solves the sphere packing problem. This allows for arbitrary rotation and translation, but is not as suitable to 2D rectangles.

Despite two decades of research into algorithms for node-node overlap removal, most widely used network visualization tools fail to properly reduce occlusion. Examples include Pajek [BM98], a common social network analysis tool, as well as our NodeXL [Smi+10]. In a recent user study [HHE06c] the authors had to hand tune the diagrams produced by Pajek to avoid occlusion. Fig. 6.9 shows how node occlusion can be eliminated by zooming out and increasing default spring lengths, at the cost of decreasing perceived clustering.

Node Occlusion Readability Metrics: I am not aware of any suitable existing readability metrics for node occlusion. I suggest a global RM proportional to the number of uniquely distinguishable items in the graph drawing, where an item can be either a node or a connected mass of overlapping nodes. On a continuous scale from 0 to 1, 1 indicates that every node is uniquely distinguishable from its neighbors (possibly including a spacing requirement) and 0 indicates that all nodes in the graph drawing are overlapping, creating one large connected mass. Similarly, a node RM can be proportional to the ratio of the node's representation area (possibly including a spacing requirement) that is obscured by other nodes. Naturally there is no edge RM for node occlusion, however node occlusion is usually grouped in the literature with edge tunneling (Section 6.4.8), which provides additional RMs.

6.4.2 Global Readability Metric $\aleph_n$

$$a = area\left(\bigcup_{j=1}^{n} bounds(n_j)\right) \tag{6.1}$$

$$a_{\max} = \sum_{j=1}^{n} area(bounds(n_j)) \tag{6.2}$$

$$\aleph_n = \frac{a}{a_{\max}} \tag{6.3}$$

6.4.3 Node Readability Metric $\aleph_n^{n_j \in N}$

The **regularized intersection** of rectangles P and Q, denoted $P \cap^* Q$, is the closure of the interior of the standard intersection $P \cap Q$. Regularization is used to remove lower-dimensional "dangling" components (for instance, lines in 2D drawings) [Mou04].

$$a_j = area\left(\bigcup_{k=1}^{n} bounds(n_j) \cap^* bounds(n_k)\right) \tag{6.4}$$

$$\aleph_n^{n_j} = 1 - \frac{a_j}{area(bounds(n_j))} \tag{6.5}$$

6.4.4 Edge Crossing $\aleph_c$

The number of **edge crossings** or intersections is the most widely accepted RM in the literature. In 1953, Moreno [Mor53] wrote, "The fewer the number of lines crossing, the better the sociogram." Edge crossings is listed as an important general RM in many books on graph drawing, including [Bat+98; Sug02; War04], as well as for automated UML diagram layout [Eic03]. As with the Node-Node Overlap metric, the effect of edge crossings can be somewhat mitigated with a halo, fog, or border effect around the edges to help distinguish them from each other. Substantial work has also been done in the design of graph drawing algorithms that specifically reduce the number of edge crossings, such as [STT81; ES90; FR91; CP96; DH96; Mut97].

Purchase's seminal RM comparison user study identified edge crossings as having the greatest impact on human understanding of general graphs of the five RMs she studied [Pur97]. This finding has been empirically validated in [PCJ96; Pur98; PCA02]. These studies focus on edge tracing tasks like finding the length of the shortest path between two nodes, though use a global count of the number of edge crossings. [War+02] suggests the number of edge crossings along the relevant edges is more important than a global measure. Additional evidence for the importance of edge crossing comes from [KA02], which deals with visualizing ordered sets. Moreover, user preference studies identify minimizing edge crossings as the most important RM for UML diagrams [PAC02; PCA02] as well as for node-link visualizations [HHE06a], and when given the option of improving on an initial force-directed or random layout, users created graph drawings with 60% fewer edge crossings on average [HR08]. [KA02] theorizes that crossed lines could be salient properties which distract the user's visual system from the relationships the drawing was designed to convey.

However, [Mut97] suggests that allowing some edge crossings can sometimes result in more readable graph drawings and recent literature points to restricting edge crossing angles being almost as effective as reducing edge crossings (Section 6.4.9). Furthermore, recent research on node-link visualizations comparing edge tracing tasks like finding groups to node importance tasks indicates that while reducing

edge crossings improves edge tracing task performance and user preference, it has little effect on node importance tasks [HHE06b; HHE05; HHE07]. This was further verified in eye tracking studies [Hua06; Hua07b; HEH08]. They postulate that this indicates the effects of edge crossings can vary depending on the situation. Further discussion of the cognitive load imposed by edge crossings quantified using eye tracking is in [Kö4; HHE06c; Hua07a; HEH08]. Fig. 6.1 demonstrates how reducing edge crossings can lead to a much more understandable drawing.

6.4.5 Global Readability Metric $\aleph_c$

I take from [Pur02] the global RM for edge crossings ($\aleph_c$) based on c, the number of pairwise edge crossings in the drawing. Scaling by an approximate upper bound for the number of crossings in the drawing, I can produce a metric over $[0, 1]$.

$$c_{all} = \sum_{i=1}^{m'}(i-1) = \frac{m'(m'-1)}{2} \tag{6.6}$$

$$c_{impossible} = \frac{1}{2}\sum_{j=1}^{n'} deg(n_j)(deg(n_j)-1) \tag{6.7}$$

$$c_{mx} = c_{all} - c_{impossible} \tag{6.8}$$

$$\aleph_c = 1 - \begin{cases} \frac{c}{c_{mx}} & \text{if } c_{mx} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.9}$$

Here, $deg(n_j)$ is the degree of node n_j. First, I calculate c_{all}, the number of crossings if every pair of edges intersect. Of those, I remove $c_{impossible}$, the impossible intersections of edges connected to the same node in a straight-line drawing. This leaves us with c_{mx}, a (probably high) upper bound to the number of crossings in the drawing. Scaling c by c_{mx} and subtracting from 1 I get the global RM for edge crossings $\aleph_c$. I can report all c crossings of m' edges in $O(m' \log m' + c)$ time and $O(m')$ space [Mul91] rather than testing all c_{mx} pairs. c_{mx} can be computed in $O(n')$ time, though only needs to be calculated once. If the graph topology is dynamically changing, only those nodes with modified degree ($\Delta n'$) need to be used to recalculate $c_{impossible}$ in $O(\Delta n')$ time and the added or removed edges must be fed back into the calculation of c. Similarly, if the layout is dynamically changing, then c must be updated for all edges whose location has changed. See [Mou04] for a discussion of various algorithms for line segment intersection reporting. The ability to use precomputed results to only test the modified edges for intersections naturally depends on the choice of algorithm, though some like [Mul91] are iterative and seem particularly suited for the addition of new edges.

6.4.6 Edge Readability Metric $\aleph_{c^{e_i}}^{e_i \in E}$

My edge RM for edge crossings ($\aleph_{c^{e_i}}^{e_i \in E}$) is defined for any edge e_i based on the number of pairwise edge crossings c^{e_i} between it and any other edge in the drawing.

Scaling as before, I can produce a metric over $[0, 1]$. With this metric I can identify the edges with the most crossings in the drawing.

$$c_{all}^{e_i} = m' - 1 \tag{6.10}$$

$$c_{impossible}^{e_i} = deg(src(e_i)) + deg(tar(e_i)) - 2 \tag{6.11}$$

$$c_{mx}^{e_i} = c_{all}^{e_i} - c_{impossible}^{e_i} \tag{6.12}$$

$$= m' - deg(src(e_i)) - deg(tar(e_i)) + 1 \tag{6.13}$$

$$\aleph_{c^{e_i}}^{e_i \in E} = 1 - \begin{cases} \frac{c^{e_i}}{c_{mx}^{e_i}} & \text{if } c_{mx}^{e_i} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.14}$$

$c_{all}^{e_i}$ is the number of edges e_i could intersect in the drawing, of which I can remove the impossible intersections $c_{impossible}^{e_i}$. Edges that have the same source or target node as e_i ($src(e_i)$ and $tar(e_i)$, respectively) cannot intersect e_i in a straight-line drawing. Thus I have $c_{mx}^{e_i}$, an upper bound to the number of edges crossing e_i. Scaling c^{e_i} by $c_{mx}^{e_i}$ and subtracting from 1 I get the edge RM for edge crossings.

6.4.7 Node Readability Metric $\aleph_{c^{n_j}}^{n_j \in N}$

My node RM for edge crossings ($\aleph_{c^{n_j}}^{n_j \in N}$) is defined for any node n_j based on c^{n_j}, the sum of the number of crossings its connected edges have (triggered crossings).

Again, I scale to a continuous metric scale of $[0, 1]$. This allows us to identify the nodes whose positions are the cause of many edge crossings.

$$c^{n_j} = \sum_{e_i \in edges(n_j)} c^{e_i} \tag{6.15}$$

$$c_{mx}^{n_j} = \sum_{e_i \in edges(n_j)} c_{mx}^{e_i} \tag{6.16}$$

$$= \sum_{e_i \in edges(n_j)} m' + 1 - deg(src(e_i)) - deg(tar(e_i)) \tag{6.17}$$

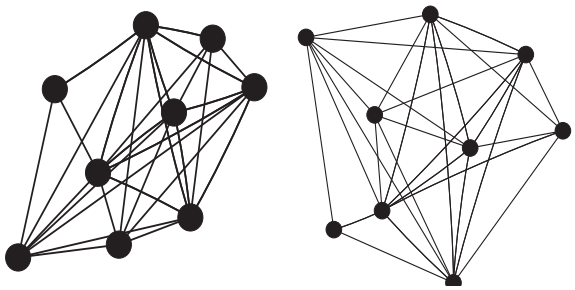
$$= \sum_{e_i \in edges(n_j)} m' + 1 - deg(n_j) - deg(adj(n_j, e_i)) \tag{6.18}$$

$$= deg(n_j)(m' + 1 - deg(n_j)) \tag{6.19}$$

$$- \sum_{e_i \in edges(n_j)} deg(adj(n_j, e_i)) \tag{6.20}$$

$$\aleph_{c^{n_j}}^{n_j \in N} = 1 - \begin{cases} \frac{c^{n_j}}{c_{mx}^{n_j}} & \text{if } c_{mx}^{n_j} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.21}$$

Here $edges(n_j)$ is the set of all edges connected to node n_j. I define an upper bound to the number of edge crossings of connected edges $c_{mx}^{n_j}$ as the sum of the individual edge upper bounds $c_{mx}^{e_i}$ from the edge RM. For all connected edges, I can pick the current node n_j as either the source or the target, and use the adjacent node along edge e_i, denoted $adj(n_j, e_i)$, as the other. As $deg(n_j) = |edges(n_j)|$, I get the formula for $c_{mx}^{n_j}$. Again scaling c^{n_j} by $c_{mx}^{n_j}$ and subtracting from 1 I get the



(a) Original layout (b) After removing edge tunnels

Figure 6.10: In Fig. 6.10a it is difficult to tell which edges connect to which nodes because of the number of edge tunnels. By zooming out and hand tuning the layout (Fig. 6.10b) we can completely eliminate edge tunnels (but not crossings).

node RM for edge crossings.

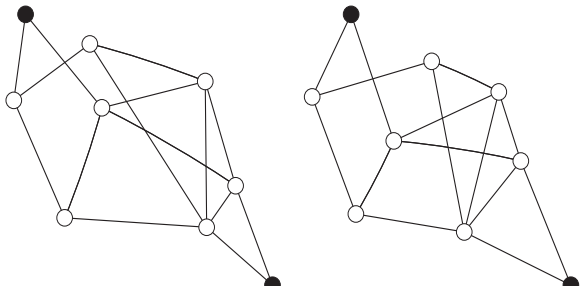
6.4.8 Edge Tunnel

There is little literature dealing with nodes occluding edges and vice versa, and it is often lumped together with node occlusion (Section 6.4.1). Because of the limited definitions available for this RM, I will call the specific case of a node occluding an edge an **edge tunnel**. The reverse can be called an **edge bridge**, but as many modern graph drawing tools (e.g. SocialAction [PS06], NodeXL [Smi+10]) draw nodes with higher priority than edges I am ignoring this case.

Both cases are accounted for by the simulated annealing graph drawing algorithm from [DH96], which incorporates the distance between every node and edge in a fine-tuning step. [Sug02] calls avoiding edge tunnels a basic rule, and for UML diagrams, [Eic03] specifies that nodes should not be too close to edges un-

less they are connected or a more important RM forces their proximity. However, many algorithms do not take this into account, including [LEN05] and the commonly used Fruchterman-Reingold algorithm [FR91]. Even tools using algorithms that remove edge tunnels are not guaranteed to do so. The excellent user study [War+02] used 200 generated graph drawings with 42 nodes each, of which the results from 7 graph drawings had to be excluded from the final analysis because of unexpected edge tunnels that implied nonexistent connections. The standard users of graph drawing tools are more likely to overlook such problems than RM researchers. Fig. 6.10 shows how zooming out and hand tuning a layout to reduce edge tunnels allows for a much clearer picture of the network topology.

Edge Tunnel Readability Metrics: The global RM for edge tunnels can be built upon the global RM for edge crossings (Section 6.4.4), comparing the number of edge tunnels in the graph drawing to an appropriate upper bound. A simple edge RM is thus an appropriate scale of the number of edge tunnels that edge has. **Local edge tunnels** is defined as a node RM for the number of edges that tunnel under that node. An second node RM for **triggered edge tunnels**, the edge tunnels of all edges connected to that node, can be specified in terms of the combined edge RMs for those edges.



(a) Original layout (b) After making edge crossings more perpendicular

Figure 6.11: In edge tracing tasks such as finding the length of the shortest path between the bottom right and top left nodes in Fig. 6.11a, increasing the edge crossing angles approaching 90 degrees (Fig. 6.11b) improves user path finding performance.

6.4.9 Edge Crossing Angle $\aleph_{eca}$

The impact of **edge crossing angles** was first introduced as a global RM by [War+02], which is based on a neurophysiological view of the user. Ware et al. claim rapid early-stage neural processing causes certain features to "pop out" to users, and that these neurons are coarsely tuned when examining angles, roughly between +/- 30 degrees. Though they did not find the impact of edge crossing angles to be significant, they did find that another angular measure, path continuity, was. This neurophysiological view supplies an explanation for the results of [HE05; Hua06; Hua07b; Hua07a; HHE08], which use an eye tracking user study to verify that the angle of edge crossings has a significant impact on user response time for edge tracing tasks. Moreover, response time significantly decreased as the cross-

ing angle tended towards 90%, though tended to level off or even slightly increase beyond 70%. This is attibuted to extra back-and-forth eye movements around acute crossings. However, as the size of the graph increases creating longer searching paths, the impact of even near-perpendicular crossings can build up and become significant [Hua07b]. See Fig. 6.11 for a demonstration of how more perpendicular edge crossing angles promote path finding tasks.

Edge Crossing Angle Readability Metrics: I believe the global RM for angular resolutioncan be modified to incorporate the average deviation of edge crossing angles from the ideal angle of ∼70 degrees instead. [War+02] uses the average cosine crossing angle as their global RM metric, and my planned experiments with these metrics may suggest that modification as well. The associated edge RM follows simply by removing the sum over all nodes and the relevant scaling. The node RM is somewhat harder to define, though it can be based on the combining the edge RMs for the node's connected edges.

6.4.10 Angular Resolution (min) $\aleph_{arm}$

The angular resolution RM refers to the minimum or average angle formed by all the edges incident to an individual node. This section discusses both but defines the minimum metric. [STT81] and [For+93] dealt with this early on, and [Pur02] defines a minimum angle metric called $\aleph_m$. [Pur97] found this metric had no effect

on path finding tasks, but it was found significant for recognizing actor status by [HHE06b].

6.4.11 Global Readability Metric $\aleph_{arm}$

$$d = \frac{1}{n}\sum_{j=1}^{n} d^{n_j} \tag{6.22}$$

$$= \frac{1}{n}\sum_{j=1}^{n} \left| \frac{\vartheta_j - \theta_{j_{min}}}{\vartheta_j} \right| \tag{6.23}$$

$$\aleph_{arm} = 1 - d \tag{6.24}$$

6.4.12 Node Readability Metric $\aleph_{arm}^{n_j \in N}$

$$d^{n_j} = \left| \frac{\vartheta_j - \theta_{j_{min}}}{\vartheta_j} \right| \tag{6.25}$$

$$\vartheta_j = \frac{360^\circ}{deg(v_j)} \tag{6.26}$$

$$\aleph_{arm}^{n_j} = 1 - d^{n_j} \tag{6.27}$$

6.4.13 Angular Resolution (avg) $\aleph_{ara}$

This metric is similar to the minimum angular resolution RM discussed in Section 6.4.10 and is described there.

6.4.14 Global Readability Metric $\aleph_{ara}$

$$d = \frac{1}{n}\sum_{j=1}^{n} d^{n_j} \tag{6.28}$$

$$= \frac{1}{n}\sum_{j=1}^{n}\left(\frac{1}{deg(n_j)}\sum_{i=1}^{deg(n_j)}\left|\frac{\vartheta_j - \theta_{i,(i+1)\%deg(n_j)}}{\vartheta_j}\right|\right) \tag{6.29}$$

$$\aleph_{ara} = 1 - d \tag{6.30}$$

6.4.15 Node Readability Metric $\aleph_{ara}^{n_j \in N}$

$$d^{n_j} = \frac{1}{deg(n_j)}\sum_{i=1}^{deg(n_j)}\left|\frac{\vartheta_j - \theta_{i,(i+1)\%deg(n_j)}}{\vartheta_j}\right| \tag{6.31}$$

$$\aleph_{arm}^{n_j} = 1 - d^{n_j} \tag{6.32}$$

where ϑ_j is the same as in Section 6.4.10.

6.4.16 Visualization Coverage Metric $\aleph_{vc}$

The **visualization coverage** or **ink** metric denoted $\aleph_{vc}$ is my attempt to quantify the amount of screen space used by the visual items in a visualization compared to the entire space available. It is formulated as the area occupied by all visual items divided by the area of the screen space. The objective of this metric is to measure the amount of theoretically available screen space, so as to quantify the reduction

in in ink presented to the user after filtering (Section 3.3.1) or motif simplification (Chapter 4). It can also measure the reduction in ink by using aggregate edges (or no edges) between groups in the Group-in-a-Box layouts (Chapter 5).

Here I use a notation of a network or graph G with $|G.nodes|$ nodes and $|G.edges|$ edges and a network visualization $V(G)$. Each individual node $n \in G.nodes$ and edge $e \in G.edges$ is indexed using subscripts (e.g., n_i, e_j). For any node, edge, or visualization k, $bounds(k)$ indicates a bounding shape b for that item in the visualization, and $area(b)$ denotes the area of that bounding shape. The visualization coverage metric $\aleph_{vc}$ is defined as follows:

$$b_n = \bigcup_{n \in \text{G.nodes}} \text{bounds}(n) \tag{6.33}$$

$$b_e = \bigcup_{e \in \text{G.edges}} \text{bounds}(e) \tag{6.34}$$

$$a = \text{area}(b_n \cup b_e) \tag{6.35}$$

$$na_{\max} = \underset{n_i \in \text{G.nodes}}{\text{argmax}}\ \text{area}(\text{bounds}(n_i)) \tag{6.36}$$

$$ea_{\max} = \underset{e_j \in \text{G.edges}}{\text{argmax}}\ \text{area}(\text{bounds}(e_j)) \tag{6.37}$$

$$a_\Delta = \max(na_{\max}, ea_{\max}) \tag{6.38}$$

$$a_{\max} = \text{area}(\text{bounds}(V(G))) \tag{6.39}$$

$$\aleph_{vc} = \frac{a - a_{\Delta}}{a_{\max}} \tag{6.40}$$

First, a union is computed of all the node bounding shapes and edge bounding shapes in the visualization, including all meta-nodes and meta-edges. In order for the metric to have a range of $[0, 1]$, this area a must have the maximum node or edge area a_{Δ} subtracted from it. This quantity is then divided by the total visualization area.

6.4.17 Group Overlap

In Algorithm 7, I describe an algorithm for counting the number of overlaps between groups (sets) of nodes in the network and the remaining nodes. It first computes a convex hull for each group, then finds the number of nodes outside the group that overlap with the convex hull. The objective is to measure how the original layout of the group affects users' perceptions of group membership, and how an alternate layouts improve on these perceptions. This measure is applicable to both motif simplification (Chapter 4) and meta-layout (Chapter 5).

I believe that convex hulls are more appropriate for this measure than alternatives like concave hulls because (1) convex hulls more accurately model the way users perceive regions of the network, and (2) it is more efficient to find inter-

Algorithm 7 Calculate the number of group-node overlaps for each group

```
1: groups = set of all groups, where each group is a set of points (x_i, y_i)
2: hullCounts = [];
3: for all g ∈ groups do
4:     count = 0
5:     hull = grahamScan(g)
6:     for all node ∈ G.nodes | node ∉ g do
7:         if intersects(hull, node) then
8:             count = count + 1
9:     hullCounts.add(count)
   return hullCounts
```

sections between convex polygons than simple polygons [Mou04]. Additionally, colored convex hulls are often used to show network group structure (e.g., [PS06]).

Two functions are called in Algorithm 7 which we assume are defined elsewhere. The first, $grahamScan(S)$, is the Graham scan algorithm[3] for computing a convex hull of a finite set of points in $O(n \log n)$ time, where n is the number of points (nodes), in this case $|g|$. The second, $intersects(a, b)$, computes the intersection of two convex polygons in $O(\log n)$ time, where n is the count of the nodes in a and b [DK83; Mou04].

The time complexity of Algorithm 7 is derived below, where $|node_j|$ is the number of sides of the polygon representing a particular node $node_j$. The other uses of $|s|$ indicate the size of the enclosed set s. E.g., $|g_i|$ is the number of nodes

[3] http://en.wikipedia.org/wiki/Graham_scan

in the set g_i.

$$time = time_a + time_b \tag{6.41}$$

$$time_a = \sum_{i=1}^{|groups|} |g_i| \log |g_i|, \text{ where } g_i \in groups \tag{6.42}$$

$$time_b = |groups||nodes| \log(\max_i(|\operatorname{hull}(g_i)|) + \max_j(|node_j|)) \tag{6.43}$$

As $\forall i, |\operatorname{hull}(g_i)| \leq |g_i| \leq \max_i(|g_i|) \leq |nodes|$, and as $\max_j(|node_j|)$ is a constant for the highest degree polygon used as a node shape,

$$time_a \leq |groups| \max_i(|g_i|) \log \max_i(|g_i|) \tag{6.44}$$

$$\leq |groups||nodes| \log \max_i(|g_i|) \tag{6.45}$$

$$time_b = O(|groups||nodes| \log \max_i(|g_i|)) \tag{6.46}$$

$$time = O(|groups||nodes| \log \max_i(|g_i|) \tag{6.47}$$

Thus, the time complexity of Algorithm 7 is given in Eq. (6.47). As $|groups| \leq |nodes|$ and $\max_i(|g_i|) \leq |nodes|$, another (much worse) upper bound would be $|nodes|^2 \log |nodes|$.

6.4.18 Additional Readability Metrics

There are many potential RMs that can be taken into account to produce effective graph drawings, and each impacts how understandable the final product is and how successfully it imparts the author's message. Many that I am investigating for standardization and inclusion in my framework are briefly discussed below.

Node Size: The size of nodes in the graph drawing can significantly affect node occlusion, edge tunneling, and the ability of users to see shapes and colors as well as read labels. I suggest outlining four size constraints depending on the amount of information to be displayed. Displaying the location of the node only requires representing a point, while adding properties like color and shape to indicate additional attributes requires more space to be identifiable. Nodes must be even larger yet in order to display meaningful text labels within the node, which are dealt with more in the following two RMs.

Node Label Distinctiveness: In many graph drawings node labels must be truncated to limit node occlusion and edge tunneling. As it is important to have uniquely identifiable and meaningful labels, users should attempt to remove common prefixes (e.g. "Department of" in an organization network). A RM for assessing the distinctiveness of individual labels in the drawing would draw attention to these problems, but must be flexible enough to accommodate unexpected prefixes. A potential solution might be found through the use of suffix trees.

Text Legibility: Similarly, the text must be sized and formatted appropriately so that it is readable in the final drawing. If this is not possible, the text should be removed to reduce node occlusion, edge tunneling, and the size of the graph. A common measure for this is the angle subtended by the text from the users point of view, though this may be difficult to translate into a RM.

Node Color & Shape Variance: As users have substantial difficulty interpreting a graph drawing using too many distinct shapes or colors to represent attributes, a RM should be defined that indicates the difficulty of keeping those combinations in memory. This might limit the publication of drawings with excessive shape and color coding.

Edge Bends: [ES90] stated that edges in a graph drawing should be as straight as possible. While the examples here deal with only straight-line drawings, edges with bends can be very useful for some types of graphs like UML diagrams. [Pur02] defines a RM for edge bends, while [Pur97] found that they have an impact on path finding tasks.

Path Continuity: How continuous a path is is inversely related to the number and size of its bends. [War+02] defines continuation at a node as "the angular deviation from a straight line of the two edges on the shortest path which emanate from the node." The sum of these deviations provides the basis for a path continuity RM. Their user study found path continuity to be significant for path finding tasks.

Geometric-path tendency: A path between two nodes in a graph drawing can "become harder to follow when many branches of the path go toward the target node" [Hua07b]. This is known as the geometric path tendency. Though a RM is not obvious, developing one may result in graph drawings better suited for edge tracing tasks.

Orthogonality: [Pur02] defines a RM for orthogonality using measures for the extent nodes and edges in the graph drawing follow the points and lines of an imaginary Cartesian grid. Orthogonality is important for some kinds of drawings, especially those of UML class diagrams [PAC02] and other hierarchical structures. However, it is unimportant and can even be misleading for node-link visualizations, as by placing nodes along imaginary lines the visualization implies to viewers that horizontally or vertically adjusted nodes are related [KA02]. Node and edge RMs for orthogonality would likely be of limited use.

Symmetry: [LNS85] observed that a graph drawing is "good" when it displays as many symmetries as possible. This was verified by [Pur97] and a RM for axial symmetry is provided by [Pur02]. Like for orthogonality, node and edge RMs for symmetry are of limited value.

Spatial Layout & Grouping: The spatial layout of nodes in a graph drawing has a substantial impact on the ability of users to ascertain the importance of actors in the network as well as identifying groups or communities of them [MBK97]. A

RM for this might compare how effectively the visual grouping of nodes in the graph drawing conveys groupings found via a community algorithm that operates only on the structure of the graph.

Edge Length: The most common algorithms for node-link visualization layout are the many variations of the spring embedder [Ead84], which attempt to reduce the variance of intra-node distances in the graph drawing. However, [HR08] found that users prefer to space clusters of nodes proportional to number of connecting edges between them. This might lend credence to a RM that analyzes the strength of relationships between clusters and compares that to the actual visible separation, though optimizing the RM would be difficult when using spring or force based layout algorithms.

Path Branches: The number of edges branching from shortest paths within the graph drawing can also have an affect on path finding tasks [War+02]. A global RM might compute the number of branches along each shortest path in the graph drawing as a measure of the general difficulty of edge tracing tasks.

6.5 Summary

My user studies, case studies, and experiments demonstrate the utility of motif simplification and Group-in-a-Box layouts for network visualization, but I am also interested in improving the effectiveness of general node-link visualizations. By

quantifying the readability of a layout we can guide analysts in making improvements and in the future feed the results in automatic layout algorithms. Past work provides definitions for several **global readability metrics**, which measure detrimental features like edge crossings and rate the layout as a whole. However, a single value is not enough to direct users to problem areas of the layout, which I address by introducing **local readability metrics** for individual nodes and edges. Moreover, I introduce several new global metrics to detect readability problems like node-node overlap and edges tunneling under nodes (node-edge overlap).

I leverage these metrics in a new method for user-assisted layout improvement. By computing the metrics in real-time as users manipulate the layout, I provide immediate visual feedback to users as they optimize their visualization, showing how they are affecting readability. As there are trade-offs when optimizing specific readability metrics, I include a survey of the related literature studying each of these metrics and their effect on user task performance. My evaluations indicate that these readability metrics help users create more effective node-link visualizations, and I plan to release both the metrics and layout improvement tool as part of NodeXL [Smi+10]. These metrics and the improvement technique were additionally implemented as part of SocialAction [PS06; PS08a; PS08b], though I have not made this code publicly available due to the research prototype nature of SocialAction.

This work aims to raise user awareness of network visualization readability issues, and applying my optimization technique will guide users in creating more effective network visualizations. I believe that many currently published networks could be substantially improved with a few modest refinements based on these readability metrics. While no set of requirements can fully capture all effective network drawings, I believe that applying select RMs for the task at hand will improve most network authors' output. These principles will need refinement to deal with large networks where node aggregation, edge bundles, and cluster markers may be necessary to allow users to make scalable comparisons.

Chapter 7

Conclusion and Future Directions

7.1 Conclusion

My dissertation contributes techniques for understanding and improving the readability of node-link network visualizations. First, I present motif simplification, a technique for reducing the complexity of node-link visualizations. With motif simplification, common repeating network motifs are replaced with easily understandable motif glyphs that require less space, are easier to understand, and reveal hidden relationships. While users must learn the visual language of motifs and glyphs, there is a dramatic payoff in the usability and readability of the visualization. I contribute design guidelines for motif glyphs; designs of glyphs to replace the high-payoff fan, connector, and clique motifs common in networks; as well as algorithms to identify these motifs. I have also developed a free and open source reference implementation, made publicly available as part of NodeXL [Smi+10], and I present results from a controlled study of 36 participants that demonstrates the benefit of motif simplification for many common network analysis tasks.

An important part of network analysis is understanding the community structures that are present, and highlighting these features can provide immediate insights during an exploration. Standard approaches for showing communities using color, shape, convex hulls, or layout algorithms do not sufficiently expose community membership, internal structure, and inter-community relationships. I address this problem with three meta-layouts that subdivide complex networks based on their community structure. The first, the Midichlorian-Directed Layout, uses a force-directed layout to visually separate clusters. The other two Group-in-a-Box (GIB) layouts display each community laid out individually within its own box, sized according to the number of nodes therein. The Fitted-Rectangles GIB layout arranges the boxes to optimize the space used while still showing inter-community relationships. The Force-Directed GIB layout, alternatively, arranges community boxes based on their aggregate ties at the cost of additional space. My implementation in NodeXL [Smi+10] automatically chooses the most appropriate Group-in-a-Box layout to best show disconnected components and different numbers or sizes of communities. Several case studies and an experimental study of 309 Twitter networks demonstrate the utility of the proposed layouts, especially for presenting the aggregate relationships between communities.

Third, my dissertation contributes a set of global and local readability metrics to help users understand and improve their node-link network visualizations. The

global metrics can be used to evaluate the effectiveness of a particular layout of the node-link visualization. Additionally, the local metrics are implemented within the analysis tool to help users identify problem areas in the visualization using color coding, and the metrics and associated colors are updated in real time as users manipulate the visualization. This provides them with immediate feedback as to how they are affecting the visualization's readability. The basics of this technique are implemented in NodeXL [Smi+10] and SocialAction [PS06], another tool for network analysis. This work provides an improved understanding of node-link visualization readability, the trade-offs when optimizing for specific tasks, and techniques users can use to improve their visualization. My hope is that it will encourage developers to take network visualization readability into account when designing analysis tools, as well as help educate users about these issues.

The three techniques I present can be used together or individually to help create more effective visualizations, especially for novice users. For example, a user could apply a Group-in-a-Box layout to highlight the clusters in the network, which are then laid out individually using motif simplification. The user could then use the interactive readability metric improvement tool to optimize the layout for presentation. The reference implementation of these techniques in NodeXL [Smi+10] will be particularly useful for novice users, as NodeXL is frequently used for teaching introductory courses on network analysis. It is my hope that these strategies for

improving visualizations of large, complex networks will demonstrate that progress is possible, and will provide several starting points for other researchers exploring additional ways to visualize networks.

7.2 Future Directions

This dissertation opens up several interesting avenues of research on node-link network visualizations. Below I detail specific opportunities for leveraging my work on motif simplification, Group-in-a-Box meta-layouts, and readability metrics to handle even larger and more complex datasets.

7.2.1 Motif Simplification

My studies indicate that motif simplification is an effective way of reducing node-link visualization complexity, but it does pose several challenges and opens up many avenues for future work. These include better education amd explanation of the motifs and their associated glyphs, but also additional techniques for showing more of the underlying network information and scaling to larger datasets. At the cost of having larger and more complex glyphs, additional details like directed edges, approximate topology, and node attribute distributions can be exposed.

7.2.1.1 Visual Complexity and Education

The visual complexity of multiple glyphs can require time for users to understand and train their eyes/mind to recognize them. As such, I have tried to keep the visual lexicon as small as possible, for example by using the same connector motif glyph for any number of anchors instead of creating many variants (Fig. 4.3). I also made several changes after the initial pilot study to improve user perception, including changing the crescent connector motif glyph to a more effective tapered diamond glyph (Fig. 4.2). However, my task-based study showed that users had still had difficulties with topology-based tasks when using motif simplification (Section 4.5). Part of this can possibly be attributed to the loss of edge information that occurred before I started using sized meta-edges between motifs (e.g., Figs. 4.9 to 4.11), which I have not yet tested.

I believe the main issue, though was that participants were only given a few minutes to understand the basics of node-link network diagrams as well as any translations between motifs and their associated glyphs. While participants had a legend available to them throughout the study, it did not seem to be enough to ensure users understood the translations. User education is likely the most promising way to improve the glyph performance, either through additional preliminary training or time spent using the techniques and becoming comfortable with the translations. Currently in NodeXL there may be the extra effort required

to learn the motif concepts and interpret the glyphs, which may deter some users, but simplification is a user choice which can be reversed at any time.

Another option would be to use more heavyweight glyphs that expose more of the underlying information to the user. Several of these approaches are discussed below for showing edge directionality, approximate motifs, arbitrary motifs, and attribute distributions. However, I have tried to strike a balance between showing the underlying information and maintaining a small visual lexicon, as well as keeping glyphs small and understandable at a distance. Heavyweight glyphs expose more, but at a substantial cost of visual clutter and space required.

7.2.1.2 Edge Directionality

Many networks have the added complexity of edge directionality, which is important for some tasks like determining information flow and trust analysis. For tasks on directed networks like path-finding, the underlying edge directionality needs to be taken into account in the glyph design so as to show these flows. I began working on this problem and developed an effective technique for subdividing fan glyphs without requiring any labels or annotations to show directionality. An example of this is shown in Fig. 7.1, with extra arrows around the edges that are not part of the glyphs.

The example directed fan motif in Fig. 7.1 is divided into three representatively sized sectors, each representing a different directionality of edges: towards the

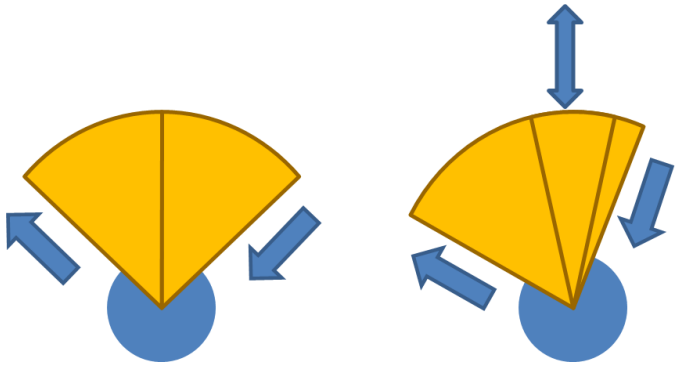


Figure 7.1: Examples of how to show edge directionality in a fan motif glyph. The arrows around the fans are not part of the glyph, and are only presented here to highlight which sector corresponds to which direction of edges.

head node, towards leaf nodes, or in both directions (reciprocated ties). The directionality of each sector can be shown with small arrows inside the sectors, but this requires a much larger glyph to be readable at a distance. Instead, I chose to arrange the sectors at different angles around the head node. The left glyph in Fig. 7.1 shows only edges pointing in one direction and that are not reciprocated. Both sectors are aligned vertically, with the incoming edge sector growing clockwise from vertical and the outgoing sector growing counter-clockwise from vertical. If only one direction of edges exist, say those pointing from the head to leaves, only that sector would be drawn. This technique for growing the sectors in different directions from vertical makes the directionality of the edges immediately clear, without requiring labels.

If there are reciprocated edges I propose the right glyph in Fig. 7.1. In this glyph there is a third sector for the reciprocated edges in the center, which grows evenly

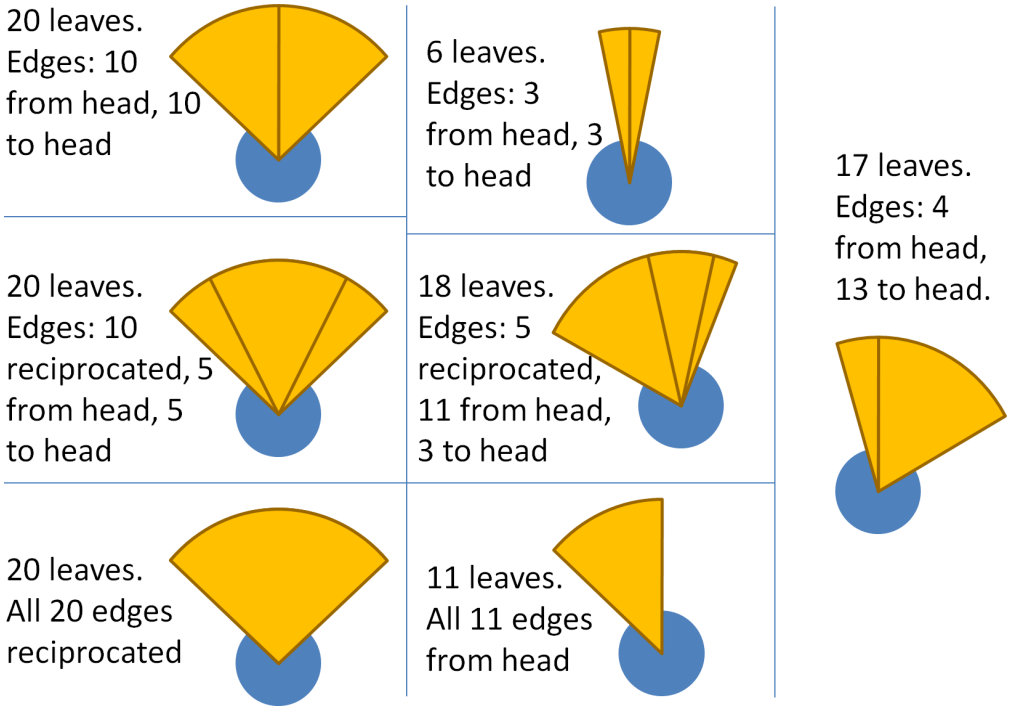


Figure 7.2: Variants of the directed fan motif glyph with different numbers leaf nodes and number of directed edges in each of the three types (from head, to head, and reciprocated).

in both directions from vertical. Then, instead of the solo-directed edges growing from vertical they grow from the edge of the central glyph. If there are no edges in one of the three sectors, it is not drawn at all and there is no extra border, again maintaining the directionality information solely in vertical alignment. Several variants for different configurations of edges are shown in Fig. 7.2. With this design the original size information of the fan glyph can be retained and directionality shown, all without labels.

This kind of subdivided design worked well for the fan glyphs, but is not as easy for things like the connector and clique motifs. Connectors are especially difficult

because they can have virtually any number of anchors, and thus combinations of edge directionalities. While a 2-Connector will have $3^2 = 9$ different combinations of edges (in-in, in-out, in-reciprocated, etc.), a 3-Connector will have $3^3 = 27$ and a 4-Connector $3^4 = 81$. When we get to the 70-Connector that showed up in the medical records example (Section 4.3.5), there are 2.5×10^{33} different combinations to show! It seems like displaying all the potential flows through a connector will be challenging. Instead, each meta-edge can be subdivided into three proportionally sized parts to show some of the directionality information. However, at this point we are creating a new heavyweight encoding for every glyph and it becomes difficult to keep the visual lexicon small, which is why I decided not to pursue this route. Cliques could be somewhat easier, but would likely require embedding a flow visualization or asymmetric adjacency matrix inside the motif glyph. This would be similar to the approach presented by NodeTrix [HFM07].

7.2.1.3 Approximate Topology

One of the best ways to scale up motif simplification to larger networks is to use approximate topology for the simplification instead of requiring exact motifs. We then return to the problem of displaying this ambiguity to the user, the basis for the exact motif simplification approach in the first place. Moreover, we have the problem of detecting these "fuzzy" motifs or functionally equivalent bits of the network. Almost-cliques are perhaps the most studied motif of the bunch and are

used as the basis for some clustering algorithms. Instead of showing the present edges in an almost-clique like normal, it could be better to instead show the absence of specific edges in the motif. These absences can be represented as light cuts across a regular polygon glyph that shows a complete clique. Alternatively, an adjacency matrix can be embedded in the clique glyph, again either showing the underlying edges (as in NodeTrix [HFM07]) or showing their absence.

Fan and connector motifs are perhaps a bit trickier to show. The presence of additional edges in a fan motif, connecting the leaf nodes, or in a connector motif, linking the span nodes, could be shown using various styles or textures for the glyph components. For example, connections between the fan leaves could be shown with a curved outer line for the sector like in the basic glyph, but when there are no connections that sector has a jagged appearance. One approach for finding "fuzzy" fan motifs would be to look for any trees in the network, which could be detected by iteratively applying the linear time algorithm detailed in Algorithm 2. These trees would have to be simplified into a staggered glyph to show the depth of its various parts, and in that case maintaining the area scaling to show node count would be difficult.

7.2.1.4 Arbitrary Motifs

A similar problem is how to detect and represent arbitrary motifs for the user. These motifs can be user-specified, like those of known interest to biologists, or

automatically generated using motif census tools. See Section 2.4 for an extensive discussion of motif census techniques, as well as the current state-of-the-art techniques for displaying the resulting motifs. Current motif census and visualization approaches are used in bioinformatics, but only find small motifs with little simplification payoff and do not create truly simplified displays. The main problem to solve would be automatically generating effective and distinguishable glyphs.

Motif simplification would be more generally applicable if we can develop a technique for detecting new kinds of motifs automatically and suggesting ones that will have a high payoff if simplified. A motif census tool could be created that makes recommendations for specific motif simplifications to target based on readability metrics for the original and reduced visualizations. One heavyweight display approach would be to embed small node-link visualizations of some representative topology inside the meta-nodes. The latest version of Cytoscape [Sha+03] will now show an exact subnetwork visualization inside a meta-node, but it would be better to automatically create a small, representative version to display.

7.2.1.5 Attribute Distributions

The current motif glyphs show a single aggregate measure of the underlying node attributes, such as their average, on the same color scale as used for the nodes. While this provides some information, it is not enough to identify unusual outliers or distributions of attribute values. With a more heavyweight glyph, this distri-

bution could be shown with small box-and-whisker charts or the like. Perhaps a bit simpler would be to use proportionally sized stripes of color to show categorical attributes or bins of attributes. Alternatively, the glyph could be subdivided into distinct sized sections for each attribute bin. While these approaches would highlight underlying attributes better, they do come at a substantial cost of screen space and visual complexity.

7.2.1.6 Overlap Handling

While the underlying topology of an individual motif is unambiguous, in some cases the choice of which motifs to simplify can lead to different overviews. The fan and connector motifs prevent ambiguous overlap, but clique motifs can overlap each other substantially. I use a heuristic that picks the largest non-overlapping clique to simplify. A more effective, but computationally hard, approach would be to rate each motif by desirability and find the optimal set of motifs by solving the NP-complete set-packing problem [Kar72]. This could result in overall better simplifications, as well as more confidence in having meaningful results.

7.2.1.7 Layout Algorithms

One of the common results of motif simplification is having the simplified network be rather dense. Most layout heuristics do not handle dense networks as well as sparse ones, though it is computationally easier than running on the original

network. Moreover, especially with the heavyweight glyphs I discuss above, it becomes important to take the glyph size and shape into account. We could apply an overlap removal post-processing step as in Section 5.4.3.3, but it is better to take the node size and shape into account in the layout algorithm. This algorithm should also take the aggregate strength of any meta-edges into account to ensure that things like tightly linked anchors of a connector motif are brought close together.

7.2.1.8 Interaction Techniques

An interesting interactive technique that could be leveraged is semantic zooming, where more details are revealed as the user zooms in on the network. Similar to Google Maps, features are revealed only when they do not add undue complexity to the display. Instead of expanding and collapsing glyphs on demand, glyphs would be expanded automatically when there is enough screen space available to present them well. This could be combined with "fuzzy" summarization [NRS08] or backbone-generation [Won+08] techniques to get further reductions in complexity, at the cost of losing some information about the topology. All these overview approaches would be especially effective for web-based network visualizations, which have a space premium and significant performance issues with even small networks.

7.2.2 Group-in-a-Box Layouts

The Group-in-a-Box layouts I have discussed could benefit from several improvements. First, better automatic parameter selection and layout choice techniques could get users to good results faster without trial and error. Moreover, better layout algorithms could be applied to get the initial group positions. Finally, additional interaction techniques could let users explore the groups in the network individually.

7.2.2.1 Automatic Parameter Selection

Currently, the initial space-filling factor used in the Force-Directed Group-in-a-Box layout (Section 5.4.3.1) is hard-coded in NodeXL at 50%. A more effective approach might iteratively lower that value if the box overlap removal step caused too much movement of the group boxes. Alternatively, the layout could run several times to correct for mistakes like the groups being placed in poor positions initially, which can cause substantial overlap or degenerate cases like a single line.

7.2.2.2 Layout Algorithm Improvements

The layout algorithm I currently use for the Force-Directed Group-in-a-Box layout is the Harel-Koren FMS layout [HK02a]. One problem with the implementation is that it does not take the aggregate meta-edge strength into account yet when positioning the group boxes, an issue I plan to address as soon as I have time. A

more substantial step would be to try this approach using other effective layout algorithms like the high-dimensional embedding (HDE) approach of Harel and Koren [HK02c] or the algebraic multigrid method (ACE) of Koren, Carmel, and Harel [KCH03]. The FM^3 algorithm [HJ05; Hac05] seems to produce particularly good results, but may be slower and difficult to implement. HDE, ACE, and FM^3 should all be able to utilize the meta-edge weight between groups.

7.2.2.3 Evaluation

My students and I are currently conducting an empirical evaluation of the Group-in-a-Box layouts on thousands of Twitter scrapes (Section 5.6), but more work is definitely needed to quantify how useful these meta-layouts are. Additional task-based studies could help quantify the benefits of the Group-in-a-Box approach and any potential pitfalls that have not been exposed through my case studies and explorations.

7.2.2.4 Automatic Layout Choice

In some cases, I choose which Group-in-a-Box layout to use based on the number of connected components, groups, and certain group properties (Section 5.4.5). Despite this, it would be good to extend this work to completely automate the Group-in-a-Box layout choice. One way to do this would be to run each layout, quantify its utility using readability metrics, and choose the best one. Alter-

natively, studies like our empirical analysis of Group-in-a-Box layouts on Twitter networks may provide sufficient data to automatically choose the best layout based on network and group statistics. Similarly, the best clustering algorithm for a network could be found by comparing how effective each clustering algorithm is when the results are displayed in the Force-Directed Group-in-a-Box layout. This would be quantified by using the readability metrics.

7.2.2.5 Interaction Techniques

Instead of displaying all the groups on the screen at the same time, interactive techniques could help users drill into particular groups. The original Treemap tool[1] and now Spotfire [Spo] allow users to drill into a Treemap interactively, showing only one box on a level. This same kind of interactive drill-down can be applied to any of the Group-in-a-Box layouts, and would be especially effective for hierarchical clusterings. An alternate technique like continuously variable zoom [Dil+94] would let users see one group in more of the screen space, while minimizing other groups to take up less space.

7.2.3 Readability Metrics

There are several ways forward for work on the readability metrics. Initially, there is a need for local node and edge versions of current global metrics that I did

[1] http://www.cs.umd.edu/hcil/treemap/

not cover as part of my work. As more metrics are developed, they should be evaluated for user task performance and integrated into a visual taxonomy for the user, which can then be used to help users choose the metrics to optimize. These optimizations could be done manually with color-coding assistance like I do now, but also using a snap-to-local-maxima or fully automatic approach.

7.2.3.1 Additional Local Metrics

There are many existing global readability metrics that I have not created local node and edge versions for, many of which are these are listed in Section 6.4.18. The development of additional local metrics would provide users with more ways to understand the effectiveness of their node-link visualizations, as well as ways to improve those visualizations. While there are many studies looking at the utility of metrics like edge crossings (Section 6.4.4), many metrics are not as well studied. With any new metrics, it becomes important to quantify how well it maps to user task performance.

7.2.3.2 Metric-Task Taxonomy and User Interface

It would be useful to document the results of new metric studies, as well as the large corpus of studies I detail in Section 6.4, in a metric-by-task taxonomy that can be presented visually to the user. While NodeXL will currently let users select which metrics to optimize, the user may not be aware of which metrics they should use

for particular tasks. This taxonomy interface would let users select a path-finding task, for example, and be given the appropriate metrics to optimize.

7.2.3.3 Automatic Metric Optimization

Once a metric-by-task user interface exists, we can then enable the user to select several of the relevant metrics to optimize. While my current implementations only show the user highlighting for one metric at a time, we could create a linear or weighted combination of the metrics to display. More interestingly, we could feed this combined metric into a snap-to-local-maxima tool, or even an automatic layout algorithm that finds the perfect layout for that user-defined energy function.

Simulated annealing [Met+53; KGV83] may be a good approach for a fully automated layout. Simulated annealing is an optimization strategy originating in statistical mechanics [Met+53] that has since been rewritten more generally [KGV83], and can be applied to many classical combinatorial problems. Surveys of the method and uses of simulated annealing can be found in [Haj85; Joh+91; JP87; LA87]. Earlier work has used simulated annealing for network layouts with a hard-coded energy function, based on metrics such as evenly-spaced nodes, uniform edge lengths, edge crossings, edge tunnels [DH96] or even to show group members proximally [Bar+08]. We could build on this to optimize the user-defined energy function that was created using the metric-by-task user interface. The running time of this approach would likely be slow ($O(N^2E)$), with memory required about

$O(max(N^2, E^2))$, but would produce "perfect" layouts for a given set of metrics.

7.3 Summary

Network data structures have been used extensively in recent years for modeling entities and their ties for many diverse disciplines. Analyzing networks involves understanding the complex relationships between entities as well as any attributes, statistics, or groupings associated with them. The omnipresent node-link visualization excels at showing network topology and features simultaneously, but many node-link visualizations are not easily readable or difficult to extract meaning from because of inherent network complexity or size. Moreover, for every network there are many potential unintelligible or even misleading visualizations.

In this dissertation I discuss strategies to help users create more effective node-link visualizations, all implemented in the NodeXL network analysis tool [Smi+10]. I first introduce a technique called motif simplification that leverages the repeating patterns or motifs in a network to reduce visual complexity and increase readability. I then discuss meta-layout algorithms that take attribute- or topology-based groupings into account, so as to more clearly show the ties within groups and the aggregate relationships between groups. Finally, I detail readability metrics to quantify the effectiveness of node-link visualizations, localize areas needing improvement, and be fed into assistive layout tools.

Each of these thrusts of my work opens up new avenues of research on network visualization. The motif simplification work can be expanded to show additional topology and attribute information, as well as arbitrary patterns in the network. My Group-in-a-Box layouts would benefit from advanced layout algorithms, in addition to automatic parameter and layout selection techniques. Finally, future work could develop local node and edge readability metrics for existing global metrics, and implement a visual metric-by-task taxonomy tool that would feed into automatic layout algorithms.

Bibliography

[Ada+04] Alex T Adai, Shailesh V Date, Shannon Wieland, and Edward M Marcotte. *LGL: Creating a map of protein function with an algorithm for visualizing very large biological networks*. In: *Journal of Molecular Biology* 340.1 (2004), pp. 179–190. DOI: 10.1016/j.jmb.2004.04.047 (cit. on pp. 39, 40).

[Ada06] Eytan Adar. *GUESS: a language and interface for graph exploration*. In: *CHI '06: Proc. SIGCHI Conference on Human Factors in Computing Systems*. 2006, pp. 791–800. DOI: 10.1145/1124772.1124889 (cit. on pp. 21, 23).

[AG05] Lada A. Adamic and Natalie Glance. *The political blogosphere and the 2004 U.S. election: Divided they blog*. In: *LinkKDD '05: Proc. 3rd International Workshop on Link Discovery*. 2005, pp. 36–43. DOI: 10.1145/1134271.1134277 (cit. on p. 3).

[AH04] James Abello and Frank van Ham. *Matrix Zoom: A visual interface to semi-external graphs*. In: *INFOVIS '04: Proc. IEEE Symposium on Information Visualization*. INFOVIS '04. 2004, pp. 183–190. DOI: 10.1109/INFVIS.2004.46 (cit. on p. 24).

[AHK06] James Abello, Frank van Ham, and Neeraj Krishnan. *ASK-GraphView: a large scale graph visualization system*. In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006), pp. 669–676. DOI: 10.1109/TVCG.2006.120 (cit. on p. 35).

[Ari08] Aleks Aris. *Visualizing and exploring networks using Semantic Substrates*. PhD thesis. University of Maryland, Department of Computer Science, 2008 (cit. on pp. 1, 21).

[AWS92] Christopher Ahlberg, Christopher Williamson, and Ben Shneiderman. *Dynamic queries for information exploration: an implementation and evaluation*. In: *CHI '92: Proc. SIGCHI Conference on Human Factors in Computing Systems*. 1992, pp. 619–626. DOI: 10.1145/142750.143054 (cit. on p. 56).

[Bar+08] Aaron Barsky, Tamara Munzner, Jennifer Gardy, and Robert Kincaid. *Cerebral: Visualizing multiple experimental conditions on a graph with biological context.* In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* 14.6 (2008), pp. 1253–1260. DOI: 10.1109/TVCG.2008.117 (cit. on pp. 6, 142, 288).

[Bat+94] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Algorithms for drawing graphs: An annotated bibliography*. In: *Computational Geometry* 4 (1994), pp. 235–282. DOI: 10.1016/0925-7721(94)00014-X (cit. on p. 33).

[Bat+98] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph drawing: Algorithms for the visualization of graphs*. Ed. by Laura Steele. Prentice Hall, 1998 (cit. on pp. 4, 33, 228, 249).

[Bez+10] Anastasia Bezerianos, Fanny Chevalier, Pierre Dragicevic, Niklas Elmqvist, and Jean-Daniel Fekete. *GraphDice: A system for exploring multivariate social networks*. In: *EuroVis '10: Proc. 2010 Eurographics/IEEE Symposium on Visualization*. 2010. DOI: 10.1111/j.1467-8659.2009.01687.x (cit. on pp. 26, 28).

[BFN85] Carlo Batini, L. Furlani, and Enrico Nardelli. *What is a good diagram? A pragmatic approach*. In: *ER '85: Proc. 4th International Conference on the Entity-Relationship Approach to Software Engineering*. 1985, pp. 312–319 (cit. on p. 33).

[BH86] Josh Barnes and Piet Hut. *A hierarchical O(N log N) force-calculation algorithm*. In: *Nature* 324.6096 (1986), pp. 446–449. DOI: 10.1038/324446a0 (cit. on pp. 154, 246).

[BHJ09] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. *Gephi: An open source software for exploring and manipulating networks*. In: *ICWSM '09: Proc. International AAAI Conference on Weblogs and Social Media*. 2009 (cit. on pp. 21, 23).

[BHJVW00] Mark Bruls, Kees Huizing, and Jarke J. Van Wijk. *Squarified Treemaps*. In: *Proc. Joint Eurographics and IEEE TCVG symposium on Visualization*. 2000, pp. 33–42 (cit. on pp. 160, 223).

[Bla+09] Jorik Blaas, Charl Botha, Edward Grundy, Mark Jones, Robert Laramee, and Frits Post. *Smooth graphs for visual exploration of higher-order state transitions*. In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009), pp. 969–976. DOI: 10.1109/TVCG.2009.181 (cit. on p. 26).

[Blu+08] Ryan Blue, Cody Dunne, Adam Fuchs, Kyle King, and Aaron Schulman. *Visualizing real-time network resource usage*. In: *VizSec '08: Proc. 5th international workshop on Visualization for Computer Security*. 2008, pp. 119–135. DOI: 10.1007/978-3-540-85933-8_12 (cit. on pp. 59–61).

[BM98] Vladimir Batagelj and Andrej Mrvar. *Pajek - Program for large network analysis*. In: *Connections* 21 (1998), pp. 47–57 (cit. on pp. 21, 23, 247).

[BMK96] Jim Blythe, Cathleen McGrath, and David Krackhardt. *The effect of graph layout on inference from social network data*. In: *GD '95: Proc. 3rd International Symposium on Graph Drawing*. GD '95. 1996, pp. 40–51. DOI: 10.1007/BFb0021783 (cit. on pp. 2, 4, 46).

[Bon+09] Elizabeth M. Bonsignore, Cody Dunne, Dana Rotman, Marc Smith, Tony Capone, Derek L. Hansen, and Ben Shneiderman. *First steps to NetViz Nirvana: Evaluating social network analysis with NodeXL*. In: *CSE '09: Proc. 2009 International Conference on Computational Science and Engineering*. Vol. 4. 2009, pp. 332–339. DOI: 10.1109/CSE.2009.120 (cit. on pp. 50, 226).

[Bra+99] Ulrik Brandes, Patrick Kenis, Jï£¡rg Raab, Volker Schneider, and Dorothea Wagner. *Explorations into the visualization of policy networks*. In: *Journal of Theoretical Politics* 11.11 (1999), pp. 75–106. DOI: 10.1177/0951692899011001004 (cit. on p. 4).

[Bru12] Tom Brughmans. *Thinking through networks: a review of formal network methods in archaeology*. English. In: *Journal of Archaeological Method and Theory* (2012), pp. 1–40. DOI: 10.1007/s10816-012-9133-8 (cit. on p. 3).

[BWK00] Michelle Q. Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. *Guidelines for using multiple views in information visualization*. In: *AVI '00: Proc. 2000 working conference on Advanced Visual Interfaces*. 2000, pp. 110–119. DOI: 10.1145/345513.345271 (cit. on p. 31).

[Cao+11] Nan Cao, Gotz, D., Sun, J., and Huamin Qu. *DICON: Interactive visual analysis of multidimensional clusters*. In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 2581–2590. DOI: 10.1109/TVCG.2011.188 (cit. on pp. 43, 44).

[CBB00] Bill Cheswick, Hal Burch, and Steve Branigan. *Mapping and visualizing the internet*. In: *Proc. 2000 USENIX Annual Technical Conference*. 2000, pp. 1–12 (cit. on p. 3).

[CC00] Chaomei Chen and Mary P. Czerwinski. *Empirical evaluation of information visualizations: An introduction*. In: *International Journal of Human-Computer Studies* 53.5 (2000), pp. 631–635. DOI: 10.1006/ijhc.2000.0421 (cit. on p. 45).

[Cha+13] Snigdha Chaturvedi, Zahra Ashktorab, Cody Dunne, Rajan Zacharia, and Ben Shneiderman. *Group-in-a-Box layouts for visualizing network communities and their ties*. Under submission. 2013 (cit. on pp. 20, 144, 145, 162, 213).

[CM85] William S. Cleveland and Robert McGill. *Graphical perception and graphical methods for analyzing scientific data*. In: *Science* 229.4716 (1985), pp. 828–833. DOI: 10.1126/science.229.4716.828 (cit. on p. 82).

[CNM04] Aaron Clauset, Mark E. J. Newman, and Cristopher Moore. *Finding community structure in very large networks*. In: *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics* 70 (6 2004), p. 066111. DOI: 10.1103/PhysRevE.70.066111 (cit. on pp. 11, 12, 140, 147, 189, 190, 197–199, 206, 207, 218).

[CP96] Michael K. Coleman and D. Stott Parker. *Aesthetics-based graph layout for human consumption*. In: *Software: Practice and Experience* 26.12 (1996), pp. 1415–1438. DOI: 10.1002/(SICI)1097-024X(199612)26:12<1415::AID-SPE69>3.3.CO;2-G (cit. on p. 249).

[Dem12] Christopher Scott Dempwolf. *Network models of regional innovation clusters and their impact on economic growth*. PhD thesis. University of Maryland, College Park, 2012 (cit. on pp. 3, 206).

[DH96] Ron Davidson and David Harel. *Drawing graphs nicely using simulated annealing*. In: *TOG: ACM Transactions on Graphics* 15.4 (1996), pp. 301–331. DOI: 10.1145/234535.234538 (cit. on pp. 34, 249, 255, 288).

[Dil+94] John Dill, Lyn Bartram, Albert Ho, and Frank Henigman. *A continuously variable zoom for navigating large hierarchical networks*. In: *Proc. IEEE SMC '94*. Vol. 1. 1994, pp. 386–390. DOI: 10.1109/ICSMC.1994.399869 (cit. on p. 286).

[DK83] David P. Dobkin and David G. Kirkpatrick. *Fast detection of polyhedral intersection*. In: *Theoretical Computer Science* 27.3 (1983), pp. 241–253. DOI: 10.1016/0304-3975(82)90120-7 (cit. on p. 263).

[DMS06] Tim Dwyer, Kim Marriott, and Peter Stuckey. *Fast node overlap removal*. In: *GD '05: Proc. 13th International Symposium on Graph Drawing*. 2006, pp. 153–164. DOI: 10.1007/11618058_15 (cit. on pp. 175, 176, 246).

[DMS07] Tim Dwyer, Kim Marriott, and Peter Stuckey. *Fast node overlap removal–correction*. In: *GD '06: Proc. 14th International Symposium on Graph Drawing*. 2007, pp. 446–447. DOI: 10.1007/978-3-540-70904-6_44 (cit. on pp. 175, 176, 246).

[DS09] Cody Dunne and Ben Shneiderman. *Improving graph drawing readability by incorporating readability metrics: A software tool for network analysts*. Human-Computer Interaction Lab Tech Report HCIL-2009-13. University of Maryland, 2009 (cit. on p. 226).

[DS13] Cody Dunne and Ben Shneiderman. *Motif simplification: improving network visualization readability with fan, connector, and clique glyphs*. In: *CHI '13: Proc. SIGCHI Conference on Human Factors in Computing Systems*. CHI '13. 2013, pp. 3247–3256. DOI: 10.1145/2470654.2466444 (cit. on pp. 20, 54, 76).

[Dun+12a] Cody Dunne, Nathalie Henry Riche, Bongshin Lee, Ronald A. Metoyer, and George G. Robertson. *GraphTrail: Analyzing large multivariate, heterogeneous networks while supporting exploration history*. In: *CHI '12: Proc. SIGCHI Conference on Human Factors in Computing Systems*. 2012, pp. 1663–1672. DOI: 10.1145/2207676.2208293 (cit. on pp. 7, 60–62).

[Dun+12b] Cody Dunne, Ben Shneiderman, Robert Gove, Judith Klavans, and Bonnie Dorr. *Rapid understanding of scientific paper collections: Integrating statistics, text analytics, and visualization*. In: *JASIST: Journal of the American Society for Information Science and Technology* 63.12 (2012), pp. 2351–2369. DOI: 10.1002/asi.22652 (cit. on pp. 68–70).

[Ead84] Peter Eades. *A heuristic for graph drawing*. In: *CN: Congressus Numerantium* 42 (1984), pp. 149–160 (cit. on pp. 5, 245, 268).

[Eic03] Holger Eichelberger. *Nice class diagrams admit good design?* In: *SoftVis '03: Proc. 2003 ACM Symposium on Software Visualization*. 2003, pp. 159–216. DOI: 10.1145/774833.774857 (cit. on pp. 33, 245, 249, 255).

[EL92] Peter Eades and Wei Lai. *Algorithms for disjoint node images*. In: *ACSC '92: Proc. 15th Australian Computer Science Conference*. 1992, pp. 253–265 (cit. on p. 246).

[ES11] David Eppstein and Darren Strash. *Listing all maximal cliques in large sparse real-world graphs*. In: *SEA '11: Proc. 10th International Symposium on Experimental Algorithms*. Vol. 6630. 2011, pp. 364–375. DOI: 10.1007/978-3-642-20662-7_31 (cit. on p. 93).

[ES90] Peter Eades and Kozo Sugiyama. *How to draw a directed graph*. In: *Journal of Information Processing* 13.4 (1990), pp. 424–437 (cit. on pp. 249, 266).

[Fek+03] Jean-Daniel Fekete, David Wang, Niem Dang, Aleks Aris, and Catherine Plaisant. *Overlaying graph links on Treemaps*. In: *Information Visualization Symposium Poster Compendium*. INFOVIS. 2003, pp. 82–83 (cit. on p. 26).

[Fek04] Jean-Daniel Fekete. *The InfoVis Toolkit*. In: *INFOVIS '04: Proc. IEEE symposium on Information Visualization*. INFOVIS '04. 2004, pp. 167–174. DOI: 10.1109/INFVIS.2004.64 (cit. on p. 22).

[For+93] Michael Formann, Torben Hagerup, James Haralambides, Michael Kaufmann, Frank Thomson Leighton, Antonios Symvonis, Emo Welzl, and Gerhard J. Woeginger. *Drawing graphs in the plane with high resolution*. In: *SIAM Journal on Computing* 22.5 (1993), pp. 1035–1052. DOI: 10.1137/0222063 (cit. on p. 258).

[FR91] Thomas M. J. Fruchterman and Edward M. Reingold. *Graph drawing by force-directed placement*. In: *SPE: Software: Practice and Experience* 21.11 (1991), pp. 1129–1164. DOI: 10.1002/spe.4380211102 (cit. on pp. 5, 107, 108, 149, 245, 249, 256).

[Fre+10] Manuel Freire, Catherine Plaisant, Ben Shneiderman, and Jen Golbeck. *ManyNets: An interface for multiple network analysis and visualization*. In: *CHI '10: Proc. SIGCHI Conference on Human Factors in Computing Systems*. 2010, pp. 213–222. DOI: 10.1145/1753326.1753358 (cit. on pp. 27, 28, 35).

[FSW06] Danyel Fisher, Marc Smith, and Howard T. Welser. *You are who you talk to: Detecting roles in Usenet newsgroups*. In: *HICSS '06: Proc. 39th Annual Hawaii International Conference on System Sciences*. 2006, p. 59.2. DOI: 10.1109/HICSS.2006.536 (cit. on p. 3).

[GFC04] Mohammad Ghoniem, Jean-Daniel Fekete, and Philippe Castagliola. *A comparison of the readability of graphs using node-link and matrix-based representations*. In: *INFOVIS '04: Proc. IEEE Symposium on Information Visualization*. INFOVIS '04. 2004, pp. 17–24. DOI: 10.1109/INFVIS.2004.1 (cit. on pp. 24, 46, 126, 128, 226).

[GGK04] Pawel Gajer, Michael T. Goodrich, and Stephen G. Kobourov. *A multi-dimensional approach to force-directed layouts of large graphs*. In: *Computational Geometry: Theory and Applications* 29.1 (2004), pp. 3–18. DOI: 10.1016/j.comgeo.2004.03.014 (cit. on p. 41).

[GH09] Emden Gansner and Yifan Hu. *Efficient node overlap removal using a proximity stress model*. In: *GD '08: Proc. 16th International Symposium on Graph Drawing*. 2009, pp. 206–217. DOI: 10.1007/978-3-642-00219-9_20 (cit. on pp. 173, 175, 176, 178–180, 246, 247).

[GK07] Joshua Grochow and Manolis Kellis. *Network motif discovery using Subgraph Enumeration and Symmetry-Breaking*. In: *RECOMB '07: Proc. 11th iInternational conference on Research in Computational Molecular Biology*. 2007, pp. 92–106. DOI: 10.1007/978-3-540-71681-5_7 (cit. on pp. 37, 86).

[GN02] Michelle Girvan and Mark E. J. Newman. *Community structure in social and biological networks*. In: *PNAS: Proc. National Academy of Sciences of the United States of America* 99.12 (2002), pp. 7821–7826. DOI: 10.1073/pnas.122653799 (cit. on pp. 140, 147).

[GN98] Emden Gansner and Stephen North. *Improved force-directed layouts*. In: *GD '98: Proc. 6th International Symposium on Graph Drawing*. 1998, pp. 364–373. DOI: 10.1007/3-540-37623-2_28 (cit. on pp. 175, 176, 246).

[Gov+11a] Robert Gove, Cody Dunne, Ben Shneiderman, Judith Klavans, and Bonnie Dorr. *Evaluating visual and statistical exploration of scientific literature networks*. In: *VL/HCC '11: Proc. 2011 IEEE Symposium on Visual Languages and Human-Centric Computing*. 2011, pp. 217–224. DOI: 10.1109/VLHCC.2011.6070403 (cit. on p. 68).

[Gov+11b] Robert Gove, Nick Gramsky, Rose Kirby, Emre Sefer, Awalin Sopan, Cody Dunne, Ben Shneiderman, and Meirav Taieb-Maimon. *NetVisia: Heat map & matrix visualization of dynamic social network statistics & content*. In: *SocialCom '11: Proc. 2011 IEEE 3rd International Conference on Social Computing*. 2011, pp. 19–26. DOI: 10.1109/PASSAT/SocialCom.2011.216 (cit. on p. 66).

[Hac05] Stefan Hachul. *A potential-field-based multilevel algorithm for drawing large graphs*. PhD thesis. Universität zu Köln, 2005 (cit. on pp. 172, 285).

[Haj85] Bruce Hajek. *A tutorial survey of theory and applications of simulated annealing*. In: *CDC '85: Proc. 24th IEEE Conference on Decision and Control*. Vol. 24. 1985, pp. 755–760. DOI: 10.1109/CDC.1985.268599 (cit. on p. 288).

[Hay+02] Kunihiko Hayashi, Michiko Inoue, Toshimitsu Masuzawa, and Hideo Fujiwara. *A layout adjustment problem for disjoint rectangles preserving orthogonal order*. In: *Systems and Computers in Japan* 33.2 (2002), pp. 31–42. DOI: 10.1002/scj.1104 (cit. on p. 246).

[HB05] Jeffrey Heer and Danah Boyd. *Vizster: Visualizing online social networks*. In: *INFOVIS '05: Proc. IEEE Symposium on Information Visualization*. INFOVIS '05. 2005, pp. 32–39. DOI: 10.1109/INFVIS.2005.1532126 (cit. on pp. 155, 157).

[HBF08] Nathalie Henry, Anastasia Bezerianos, and Jean-Daniel Fekete. *Improving the readability of clustered social networks using node duplication*. In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* 14.6 (2008), pp. 1317–1324. DOI: 10.1109/TVCG.2008.141 (cit. on p. 226).

[HCL05] Jeffrey Heer, Stuart K. Card, and James A. Landay. *Prefuse: A toolkit for interactive information visualization*. In: *CHI '05: Proc. SIGCHI Conference on Human Factors in Computing Systems*. 2005, pp. 421–430. DOI: 10.1145/1054972.1055031 (cit. on pp. 5, 22, 154, 157, 246).

[HDS10] Derek Hansen, Cody Dunne, and Ben Shneiderman. *Analyzing social media networks with NodeXL*. Proc. 27th Annual Human-Computer Interaction Lab Symposium. 2010 (cit. on p. 2).

[HE05] Weidong Huang and Peter Eades. *How people read graphs*. In: *APVis '05: Proc. 2005 Asia-Pacific Symposium on Information Visualisation*. 2005, pp. 51–58 (cit. on p. 257).

[HEH08] Weidong Huang, Peter Eades, and Seok-Hee Hong. *Beyond time and error: A cognitive approach to the evaluation of graph drawings*. In: *BELIV '08: Proc. 2008 conference on BEyond time and errors: novel evaLuation methods for Information Visualization*. 2008, pp. 1–8. DOI: 10.1145/1377966.1377970 (cit. on p. 251).

[Hen+07] Nathalie Henry, Howard Goodell, Niklas Elmqvist, and Jean-Daniel Fekete. *20 years of four HCI conferences: A visual exploration*. In: *International Journal of Human-Computer Interaction* 23.3 (2007), pp. 239–285. DOI: 10.1080/10447310701702402 (cit. on p. 3).

[Her+99] Ivan Herman, M. Scott Marshall, Guy Melançon, D. J. Duke, Maylis. Delest, and J.-P. Domenger. *Skeletal images as visual cues in graph visualization*. In: *Data Visualization '99: Proc. joint Eurographics and IEEE TVCG Symposium on Visualization*. 1999, pp. 13–22 (cit. on p. 36).

[HF06] Nathalie Henry and Jean-Daniel Fekete. *MatrixExplorer: A dual-representation system to explore social networks*. In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006), pp. 677–684. DOI: 10.1109/TVCG.2006.160 (cit. on p. 23).

[HF07] Nathalie Henry and Jean-Daniel Fekete. *MatLink: Enhanced matrix visualization for analyzing social networks*. In: *INTERACT '07: Proc. 11th IFIP TC 13 International Conference on Human-computer interaction*. 2007, pp. 288–302. DOI: 10.1007/978-3-540-74800-7_24 (cit. on pp. 21, 24, 46, 126–128).

[HFM07] Nathalie Henry, Jean-Daniel Fekete, and Michael J. McGuffin. *NodeTrix: A hybrid visualization of social networks*. In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* 13.6 (2007), pp. 1302–1309. DOI: 10.1109/TVCG.2007.70582 (cit. on pp. 24–26, 279, 280).

[HHE05] Weidong Huang, Seok-Hee Hong, and Peter Eades. *Layout effects: Comparison of sociogram drawing conventions*. Tech. rep. 575. University of Sydney, 2005 (cit. on p. 251).

[HHE06a] Weidong Huang, Seok-Hee Hong, and Peter Eades. *How people read sociograms: A questionnaire study*. In: *APVis '06: Proc. 2006 Asia-Pacific Symposium on Information Visualisation*. 2006, pp. 199–206 (cit. on p. 250).

[HHE06b] Weidong Huang, Seok-Hee Hong, and Peter Eades. *Layout effects on sociogram perception*. In: *GD '05: Proc. 13th International Symposium on Graph Drawing*. Vol. 3843/2006. Lecture Notes in Computer Science. 2006, pp. 262–273. DOI: 10.1007/11618058_24 (cit. on pp. 251, 259).

[HHE06c] Weidong Huang, Seok-Hee Hong, and Peter Eades. *Predicting graph reading performance: A cognitive approach*. In: *APVis '06: Proc. 2006 Asia-Pacific Symposium on Information Visualisation*. 2006, pp. 207–216. DOI: 10.1145/1151903.1151933 (cit. on pp. 247, 251).

[HHE07] Weidong Huang, Seok-Hee Hong, and Peter Eades. *Effects of sociogram drawing conventions and edge crossings in social network visualizations*. In: *JGAA: Journal of Graph Algorithms and Applications* 11.2 (2007), pp. 397–429 (cit. on p. 251).

[HHE08] Weidong Huang, Seok-Hee Hong, and Peter Eades. *Effects of crossing angles*. In: *PacificVIS '08: Proc. 2008 IEEE Pacific Visualization Symposium*. 2008, pp. 41–46. DOI: 10.1109/PACIFICVIS.2008.4475457 (cit. on p. 257).

[HJ05] Stefan Hachul and Michael Jünger. *Drawing large graphs with a potential-field-based multilevel algorithm*. In: *GD '04: Proc. 12th International Symposium on Graph Drawing*. Vol. 3383/2005. Lecture Notes in Computer Science. 2005, pp. 285–295. DOI: 10.1007/978-3-540-31843-9_29 (cit. on pp. 5, 7, 41, 172, 285).

[HJ06] Stefan Hachul and Michael Jünger. *An experimental comparison of fast algorithms for drawing general large graphs*. In: *GD '05: Proc. 13th International Symposium on Graph Drawing*. Vol. 3843/2006. Lecture Notes in Computer Science. 2006, pp. 235–250. DOI: 10.1007/11618058_22 (cit. on pp. 6, 41–43, 172).

[HJ07] Stefan Hachul and Michael Jünger. *Large-graph layout algorithms at work: an experimental study*. In: *JGAA: Journal of Graph Algortihms and Applications* 11.2 (2007), pp. 345–369. DOI: 10.7155/jgaa.00150 (cit. on p. 172).

[HK01] David Harel and Yehuda Koren. *A fast multi-scale method for drawing large graphs*. In: *GD '00: Proc. 8th International Symposium on Graph Drawing*. 2001, pp. 235–287. DOI: 10.1007/3-540-44541-2_18 (cit. on p. 220).

[HK02a] David Harel and Yehuda Koren. *A fast multi-scale method for drawing large graphs*. In: *JGAA: Journal of Graph Algorithms and Applications* 6.3 (2002), pp. 179–202. DOI: 10.7155/jgaa.00051 (cit. on pp. 5, 11, 12, 41, 107, 109, 114, 116, 128, 171–174, 178, 179, 189, 190, 195, 196, 199, 228, 284).

[HK02b] David Harel and Yehuda Koren. *Drawing graphs with non-uniform vertices*. In: *AVI '02: Proc. Working Conference on Advanced Visual Interfaces*. 2002, pp. 157–166. DOI: 10.1145/1556262.1556288 (cit. on p. 246).

[HK02c] David Harel and Yehuda Koren. *Graph drawing by high-dimensional embedding*. In: *GD '02: Proc. 10th International Symposium on Graph Drawing*. Vol. 2528. Lecture Notes in Computer Science. 2002, pp. 207–219. DOI: 10.1007/3-540-36151-0_20 (cit. on pp. 5, 7, 172, 285).

[HL03] Xiaodi Huang and Wei Lai. *Force-Transfer: A new approach to removing overlapping nodes in graph layout*. In: *ACSC '03: Proc. 26th Australasian Computer Science Conference*. 2003, pp. 349–358 (cit. on p. 246).

[HMM00] Ivan Herman, Guy Melançon, and M. Scott Marshall. *Graph visualization and navigation in information visualization: a survey*. In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* 6.1 (2000), pp. 24–43. DOI: 10.1109/2945.841119 (cit. on p. 232).

[Hol06] Danny Holten. *Hierarchical edge bundles: visualization of adjacency relations in hierarchical data*. In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006), pp. 741–748. DOI: 10.1109/TVCG.2006.147 (cit. on pp. 43, 44).

[HR08] Frank van Ham and Bernice E. Rogowitz. *Perceptual organization in user-generated graph layouts*. In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* 14.6 (2008), pp. 1333–1339. DOI: 10.1109/TVCG.2008.155 (cit. on pp. 250, 268).

[HSS11] Derek Hansen, Ben Shneiderman, and Mark Smith. *Analyzing social media networks with NodeXL: Insights from a connected world*. Ed. by Mary James and David Bevans. Morgan Kaufmann, 2011 (cit. on pp. 49, 50, 105, 106, 108, 109).

[Hua+05] Weidong Huang, Colin Murray, Xiaobin Shen, Le Song, Ying Xin Wu, and Lanbo Zheng. *Visualisation and analysis of network motifs*. In: *INFOVIS '05: Proc. 9th International Conference on Information Visualisation*. 2005, pp. 697–702. DOI: 10.1109/IV.2005.138 (cit. on p. 38).

[Hua+07] Xiaodi Huang, Wei Lai, A. S. M. Sajeev, and Junbin Gao. *A new algorithm for removing node overlapping in graph visualization*. In: *Information Sciences* 177.14 (2007), pp. 2821–2844. DOI: 10.1016/j.ins.2007.02.016 (cit. on p. 246).

[Hua06] Weidong Huang. *An eye tracking study into the effects of graph layout*. Tech. rep. University of Sydney, 2006 (cit. on pp. 251, 257).

[Hua07a] Weidong Huang. *Beyond time and error: A cognitive approach to the evaluation of graph visualizations*. PhD thesis. University of Sydney, 2007 (cit. on pp. 251, 257).

[Hua07b] Weidong Huang. *Using eye tracking to investigate graph layout effects*. In: *APVis '07: Proc. 2007 Asia-Pacific Symposium on Information Visualisation*. 2007, pp. 97–100. DOI: 10.1109/APVIS.2007.329282 (cit. on pp. 46, 251, 257, 258, 267).

[HW04] Frank van Ham and Jarke J. van Wijk. *Interactive visualization of small world graphs*. In: *INFOVIS '04: Proc. IEEE Symposium on Information Visualization*. INFOVIS '04. 2004, pp. 199–206. DOI: 10.1109/INFVIS.2004.43 (cit. on p. 35).

[Ima+09] Takashi Imamichi, Yohei Arahori, Jaeseong Gim, Seok-Hee Hong, and Hiroshi Nagamochi. *Removing node overlaps using multi-sphere scheme*. In: *GD '08: Proc. 16th International Symposium on Graph Drawing*. 2009, pp. 296–301. DOI: 10.1007/978-3-642-00219-9_28 (cit. on p. 247).

[Joh+91] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon. *Optimization by simulated annealing: An experimental evaluation; Part II, graph coloring and number partitioning*. In: *Operations Research* 39 (3 1991), pp. 378–406. DOI: 10.1287/opre.39.3.378 (cit. on p. 288).

[JP87] David S. Johnson and Henry O. Pollak. *Hypergraph planarity and the complexity of drawing Venn diagrams*. In: *Journal of Graph Theory* 11.3 (1987), pp. 309–325. DOI: 10.1002/jgt.3190110306 (cit. on p. 288).

[JS91] Brian Johnson and Ben Shneiderman. *Tree-Maps: A space-filling approach to the visualization of hierarchical information structures*. In: *VIS '91: Proc. 1991 IEEE Conference on Visualization*. 1991, pp. 284–291. DOI: 10.1109/VISUAL.1991.175815 (cit. on p. 160).

[KÖ4] Christof Körner. *Sequential processing in comprehension of hierarchical graphs*. In: *Applied Cognitive Psychology* 18.4 (2004), pp. 467–480. DOI: 10.1002/acp.997 (cit. on p. 251).

[KA02] Christof Körner and Dietrich Albert. *Speed of comprehension of visualized ordered sets.* In: *Journal of Experimental Psychology: Applied* 8.1 (2002), pp. 57–71. DOI: 10.1037/1076-898X.8.1.57 (cit. on pp. 250, 267).

[Kan+06] Hyunmo Kang, Catherine Plaisant, Bongshin Lee, and Benjamin B. Bederson. *NetLens: Iterative exploration of content-actor network data*. In: *VAST '06: Proc. IEEE Symposium on Visual Analytics Science And Technology*. 2006, pp. 91–98. DOI: 10.1109/VAST.2006.261426 (cit. on pp. 30, 34, 63).

[Kar72] Richard M. Karp. *Reducibility among combinatorial problems*. In: *Complexity of Computer Computations*. 1972, pp. 85–103 (cit. on pp. 94, 282).

[KCH03] Yehuda Koren, Liran Carmel, and David Harel. *Drawing huge graphs by algebraic multigrid optimization*. In: *Multiscale Modeling & Simulation* 1.4 (2003), pp. 645–673. DOI: 10.1137/S154034590241370X (cit. on pp. 5, 172, 285).

[Kel+03] Brian P Kelley, Roded Sharan, Richard M Karp, Taylor Sittler, David E Root, Brent R Stockwell, and Trey Ideker. *Conserved pathways within bacteria and yeast as revealed by global protein network alignment.* In: *PNAS: Proc. National Academy of Sciences of the United States of America* 100.20 (2003), pp. 11394–11399. DOI: 10.1073/pnas.1534710100 (cit. on p. 3).

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. *Optimization by simulated annealing*. In: *Science* 220.4598 (1983), pp. 671–680. DOI: 10.1126/science.220.4598.671 (cit. on p. 288).

[Knu93] Donald Ervin Knuth. *The Stanford GraphBase: A platform for combinatorial computing*. Addison-Wesley, 1993 (cit. on p. 141).

[KSS06] Christian Klukas, Falk Schreiber, and Henning Schwöbbermeyer. *Coordinated perspectives and enhanced force-directed layout for the analysis of network motifs*. In: *APVis '06: Proc. 2006 Asia-Pacific Symposium on Information Visualisation*. 2006, pp. 39–48 (cit. on p. 38).

[LA87] Peter J. M. Laarhoven and Emile H. L. Aarts. *Simulated annealing: theory and applications*. D. Reidel Publishing Company, 1987 (cit. on p. 288).

[Lam+11] Heidi Lam, Enrico Bertini, Petra Isenberg, Catherine Plaisant, and Sheelagh Carpendale. *Empirical studies in information visualization: Seven scenarios*. In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* PP.99 (2011), p. 1. DOI: 10.1109/TVCG.2011.279 (cit. on p. 45).

[LE02] Wei Lai and Peter Eades. *Removing edge-node intersections in drawings of graphs*. In: *Information Processing Letters* 81.2 (2002), pp. 105–110. DOI: 10.1016/S0020-0190(01)00194-6 (cit. on pp. 245, 246).

[Lee+05] Bongshin Lee, Mary Czerwinski, George Robertson, and Benjamin B. Bederson. *Understanding research trends in conferences using PaperLens*. In: *CHI EA '05:: Proc. CHI '05 Extended Abstracts on Human Factors in Computing Systems*. 2005, pp. 1969–1972. DOI: 10.1145/1056808.1057069 (cit. on p. 63).

[Lee+06] Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. *Task taxonomy for graph visualization*. In: *BELIV '06: Proc. 2006 AVI workshop on BEyond time and errors: novel evaLuation methods for Information Visualization*. 2006, pp. 1–5. DOI: 10.1145/1168149.1168168 (cit. on pp. 46, 126).

[Lee+09] Bongshin Lee, Greg Smith, George G. Robertson, Mary Czerwinski, and Desney S. Tan. *FacetLens: Exposing trends and relationships to support sensemaking within faceted datasets*. In: *CHI '09: Proc. SIGCHI Conference on Human Factors in Computing Systems*. 2009, pp. 1293–1302. DOI: 10.1145/1518701.1518896 (cit. on pp. 29, 63).

[LEN05] Wanchun Li, Peter Eades, and Nikola S. Nikolov. *Using spring algorithms to remove node overlapping*. In: *APVis '05: Proc. 2005 Asia-Pacific Symposium on Information Visualisation*. 2005, pp. 131–140 (cit. on pp. 245, 246, 256).

[Lim13] Manuel Lima. *Visual Complexity: Mapping Patterns of Information*. Princeton Architectural Press, 2013 (cit. on p. 1).

[Llo82] Stuart P. Lloyd. *Least squares quantization in PCM*. In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489 (cit. on pp. 140, 150).

[LMR98] Kelly A. Lyons, Henk Meijer, and David Rappaport. *Algorithms for cluster busting in anchored graph drawing*. In: *JGAA: Journal of Graph Algorithms and Applications* 2.1 (1998), pp. 1–24 (cit. on p. 246).

[LNS85] R. J. Lipton, S. C. North, and J. S. Sandberg. *A method for drawing graphs*. In: *SCG '85: Proc. 1st Annual Symposium on Computational Geometry*. 1985, pp. 153–160. DOI: 10.1145/323233.323254 (cit. on p. 267).

[LSS12] Qi Liao, Lei Shi, and Xiaohua Sun. *Anomaly analysis and visualization through compressed graphs*. In: *LDAV '12: Proc. IEEE Symposium on Large-Scale Data Analysis and VIsualization Poster Session*. 2012 (cit. on p. 35).

[Lus+04] Nicholas M Luscombe, M. Madan Babu, Haiyuan Yu, Michael Snyder, Sarah A Teichmann, and Mark Gerstein. *Genomic analysis of regulatory network dynamics reveals large topological changes*. In: *Nature* 431.7006 (2004), pp. 308–312. DOI: 10.1038/nature02782 (cit. on p. 37).

[Mac86] Jock Mackinlay. *Automating the design of graphical presentations of relational information*. In: *TOG: ACM Transactions on Graphics* 5.2 (1986), pp. 110–141. DOI: 10.1145/22949.22950 (cit. on pp. 26, 80, 82).

[Mar+03] Kim Marriott, Peter Stuckey, Vincent Tam, and Weiqing He. *Removing node overlapping in graph layout using constrained optimization*. In: *Constraints* 8.2 (2003), pp. 143–171. DOI: 10.1023/A:1022371615202 (cit. on p. 246).

[MB04] Cathleen McGrath and Jim Blythe. *Do you see what I want you to see? The effects of motion and spatial layout on viewers' perceptions of graph structure*. In: *JOSS: Journal of Social Structure* 5.2 (2004) (cit. on p. 157).

[MBK97] Cathleen McGrath, Jim Blythe, and David Krackhardt. *The effect of spatial arrangement on judgments and errors in interpreting graphs*. In: *SN: Social Networks* 19.3 (1997), pp. 223–242. DOI: 10.1016/S0378-8733(96)00299-7 (cit. on pp. 4, 267).

[MDD09] Saif Mohammad, Cody Dunne, and Bonnie Dorr. *Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus*. In: *EMNLP '09: Proc. 2009 conference on Empirical Methods in Natural Language Processing*. 2009, pp. 599–608 (cit. on pp. 67, 68).

[Med+02] Michael C. Medlock, Dennis Wixon, Mark Terrano, Ramon L. Romero, and Bill Fulton. *Using the RITE method to improve products: A definition and a case study*. In: *Proc. Usability Professionalï£¡s Association 2002*. 2002 (cit. on p. 45).

[Med+05] Michael C. Medlock, Dennis Wixon, Mike McGee, and Dan Welsh. *Cost-justifying usability: An update for an Internet age*. In: 2005. Chap. 17, pp. 489–517 (cit. on p. 45).

[Met+53] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. *Equation of state calculations by fast computing machines*. In: *Journal of Chemical Physics* 21.6 (1953), pp. 1087–1092. DOI: 10.1063/1.1699114 (cit. on p. 288).

[Mil+02] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. *Network motifs: Simple building blocks of complex networks.* In: *Science* 298.5594 (2002), pp. 824–827. DOI: 10.1126/science.298.5594.824 (cit. on p. 36).

[Mis+95] Kazuo Misue, Peter Eades, Wei Lai, and Kozo Sugiyama. *Layout adjustment and the mental map*. In: *Journal of Visual Languages & Computing* 6.2 (1995), pp. 183–210. DOI: 10.1006/jvlc.1995.1010 (cit. on pp. 157, 245, 246).

[Mor53] Jacob L. Moreno. *Who shall survive? Foundations of sociometry, group psychotherapy and sociodrama.* Beacon House, 1953, p. 141 (cit. on pp. 3, 249).

[Mou04] David M. Mount. *Geometric intersection*. In: *The Handbook of Discrete and Computational Geometry*. 2nd ed. 2004, pp. 857–876 (cit. on pp. 249, 252, 263).

[Mul91] Kentan Mulmuley. *A fast planar partition algorithm, II*. In: *Journal of the ACM* 38.1 (1991), pp. 74–103. DOI: 10.1145/102782.102785 (cit. on p. 252).

[Mur08] Scott Murray. *Visualizing network relationship*. Tech. rep. Massachusetts College of Art and Design, 2008 (cit. on p. 83).

[Mut97] Petra Mutzel. *An alternative method to crossing minimization on hierarchical graphs*. In: *SIAM Journal on Optimization*. Vol. 1190/1997. Lecture Notes in Computer Science. 1997, pp. 318–333. DOI: 10.1007/3-540-62495-3_57 (cit. on pp. 249, 250).

[Nav+09] Saket Navlakha, James White, Niranjan Nagarajan, Mihai Pop, and Carl Kingsford. *Finding biologically accurate clusterings in hierarchical tree decompositions using the variation of information*. In: *RECOMB '09: Proc. 14th Annual international conference on Research in Computational Molecular Biology*. 2009 (cit. on pp. 140, 150).

[New04] Mark E. J. Newman. *Fast algorithm for detecting community structure in networks*. In: *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics* 69.6 (2004), p. 066133. DOI: 10.1103/PhysRevE.69.066133 (cit. on pp. 154, 240, 241).

[NG04] Mark E. J. Newman and Michelle Girvan. *Finding and evaluating community structure in networks*. In: *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics* 69.2 Pt 2 (2004), p. 026113. DOI: 10.1103/PhysRevE.69.026113 (cit. on pp. 141, 147).

[Noa04] Andreas Noack. *An energy model for visual graph clustering*. In: *GD '03: Proc. 11th International Symposium on Graph Drawing*. Vol. 2912/2004. Lecture Notes in Computer Science. 2004, pp. 425–436. DOI: 10.1007/978-3-540-24595-7_40 (cit. on pp. 41, 142).

[Nor06] Chris North. *Toward measuring visualization insight*. In: *CGA: IEEE Computer Graphics and Applications* 26.3 (2006), pp. 6–9. DOI: 10.1109/MCG.2006.70 (cit. on p. 45).

[NRS08] Saket Navlakha, Rajeev Rastogi, and Nisheeth Shrivastava. *Graph summarization with bounded error*. In: *SIGMOD '08: Proc. 2008 ACM SIGMOD international conference on Management of data*. SIGMOD '08. 2008, pp. 419–432. DOI: 10.1145/1376616.1376661 (cit. on pp. 35, 154, 185, 283).

[NS00] Chris North and Ben Shneiderman. *Snap-together visualization: can users construct and operate coordinated visualizations?* In: *International Journal of Human-Computer Studies* 53.5 (2000), pp. 715–739. DOI: DOI:10.1006/ijhc.2000.0418 (cit. on p. 31).

[NWB06] NWB Team. *Network Workbench [Software]*. 2006 (cit. on p. 31).

[OM+03] Joshua O' Madadhain, Danyel Fisher, Scott White, and Yan-Biao Boey. *The JUNG (Java Universal Network/Graph) framework*. Tech. rep. UCI-ICS 03-17. University of California, Irvine, 2003 (cit. on p. 22).

[PAC02] Helen C. Purchase, Jo-Anne Allder, and David Carrington. *Graph layout aesthetics in UML diagrams: User preferences*. In: *JGAA: Journal of Graph Algorithms and Applications* 6.3 (2002), pp. 255–279 (cit. on pp. 250, 267).

[PCA02] Helen C. Purchase, David Carrington, and Jo-Anne Allder. *Empirical evaluation of aesthetics-based graph layout*. In: *Empirical Software Engineering* 7.3 (2002), pp. 233–255. DOI: 10.1023/A:1016344215610 (cit. on p. 250).

[PCJ96] Helen C. Purchase, Robert F. Cohen, and Murray James. *Validating graph drawing aesthetics*. In: *GD '95: Proc. 3rd International Symposium on Graph Drawing*. Vol. 1027/1996. Lecture Notes in Computer Science. 1996, pp. 435–446. DOI: 10.1007/BFb0021827 (cit. on p. 250).

[PFG08] Catherine Plaisant, Jean-Daniel Fekete, and Georges Grinstein. *Promoting insight-based evaluation of visualizations: From contest to benchmark repository*. In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* 14.1 (2008), pp. 120–134. DOI: 10.1109/TVCG.2007.70412 (cit. on p. 45).

[PHG07] Helen C. Purchase, Eve Hoggan, and Carsten Görg. *How important is the "mental map"? – An empirical investigation of a dynamic graph layout algorithm*. In: *GD '06: Proc. 14th International Symposium on Graph Drawing*. Vol. 4372. Lecture notes in Computer Science. 2007, pp. 184–195. DOI: 10.1007/978-3-540-70904-6_19 (cit. on p. 157).

[PL96] Helen C. Purchase and David Leonard. *Graph drawing aesthetic metrics*. Tech. rep. 361. Key Centre for Software Technology, Dept. of Computer Science, University of Queensland, 1996 (cit. on pp. 14, 33, 226).

[Pre+93] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in FORTRAN; the art of scientific computing*. 2nd ed. Cambridge University Press, 1993 (cit. on p. 154).

[PS06] Adam Perer and Ben Shneiderman. *Balancing systematic and flexible exploration of social networks*. In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006), pp. 693–700. DOI: 10.1109/TVCG.2006.122 (cit. on pp. 23, 157, 230, 232, 241, 255, 263, 269, 273).

[PS08a] Adam Perer and Ben Shneiderman. *Integrating statistics and visualization: Case studies of gaining clarity during exploratory data analysis*. In: *CHI '08: Proc. SIGCHI Conference on Human Factors in Computing Systems*. 2008, pp. 265–274. DOI: 10.1145/1357054.1357101 (cit. on pp. 46, 49, 157, 230, 233, 269).

[PS08b] Adam Perer and Ben Shneiderman. *Systematic yet flexible discovery: Guiding domain experts through exploratory data analysis*. In: *IUI '08: Proc. 13th International Conference on Intelligent User Interfaces*. 2008, pp. 109–118. DOI: 10.1145/1378773.1378788 (cit. on pp. 157, 230, 233, 269).

[PS09] Adam Perer and Ben Shneiderman. *Integrating statistics and visualization for exploratory power: From long-term case studies to design guidelines*. In: *CGA: IEEE Computer Graphics and Applica-*

tions 29.3 (2009), pp. 39–51. DOI: 10.1109/MCG.2009.44 (cit. on pp. 46, 49).

[Pup+11] Sergey Pupyrev, Lev Nachmanson, Sergey Bereg, and Alexander E. Holroyd. *Edge routing with ordered bundles*. In: *GD '11: Proc. 19th International Symposium on Graph Drawing*. 2011, pp. 136–147. DOI: 10.1007/978-3-642-25878-7_14 (cit. on p. 43).

[Pur02] Helen C. Purchase. *Metrics for graph drawing aesthetics*. In: *JVLC: Journal of Visual Languages & Computing* 13 (2002), pp. 501–516. DOI: 10.1006/jvlc.2002.0232 (cit. on pp. 14, 33, 46, 226, 244, 245, 251, 258, 266, 267).

[Pur97] Helen C. Purchase. *Which aesthetic has the greatest effect on human understanding?* In: *GD '97: Proc. 5th International Symposium on Graph Drawing*. Vol. 1353/1997. Lecture Notes in Computer Science. 1997, pp. 248–261. DOI: 10.1007/3-540-63938-1_67 (cit. on pp. 250, 258, 266, 267).

[Pur98] Helen C. Purchase. *The effects of graph layout*. In: *OZCHI '08: Proc. 2008 Australasian Computer Human Interaction Conference*. 1998, pp. 80–86. DOI: 10.1109/OZCHI.1998.732199 (cit. on p. 250).

[RLD] Nathalie Riche, Bongshin Lee, and Cody Dunne. *Interactive visualization for exploring multi-modal, multi-relational, and multivariate graph data*. English. U.S. Patent Application 13/041474 (cit. on p. 60).

[Rod+11] Eduarda Mendes Rodrigues, Natasa Milic-Frayling, Marc Smith, Ben Shneiderman, and Derek Hansen. *Group-in-a-Box layout for multi-faceted analysis of communities*. In: *SocialCom '11: Proc. 2011 IEEE 3rd International Conference on Social Computing*. 2011, pp. 354–361. DOI: 10.1109/PASSAT/SocialCom.2011.139 (cit. on pp. 13, 142, 143, 160, 161, 223).

[SA06] Ben Shneiderman and Aleks Aris. *Network visualization by Semantic Substrates*. In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006), pp. 733–740. DOI: 10.1109/TVCG.2006.166 (cit. on pp. 1, 5, 21, 26, 27, 46, 126, 148).

[Sar+06] Purvi Saraiya, Chris North, Vy Lam, and Karen A. Duca. *An insight-based longitudinal study of visual analytics*. In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* 12.6 (6 2006), pp. 1511–1522. DOI: 10.1109/TVCG.2006.85 (cit. on p. 45).

[SD12] Ben Shneiderman and Cody Dunne. *Interactive network exploration to derive insights: Filtering, clustering, grouping, and simplification*. In: *GD '12: Proc. 20th International Symposium on Graph Drawing*. Vol. 7704. Keynote. 2012, pp. 2–18. DOI: 10.1007/978-3-642-36763-2_2 (cit. on pp. 7, 13, 20, 54, 76, 145).

[Sha+03] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S. Baliga, Jonathan T. Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. *Cytoscape: A software environment for integrated models of biomolecular interaction networks.* In: *Genome Research* 13.11 (2003), pp. 2498–2504. DOI: 10.1101/gr.1239303 (cit. on pp. 21, 23, 24, 63, 184, 185, 281).

[Shn+12] Ben Shneiderman, Cody Dunne, Puneet Sharma, and Ping Wang. *Innovation trajectories for information visualizations: Comparing treemaps, cone trees, and hyperbolic trees*. In: *IVS: Information Visualization* 11.2 (2012), pp. 87–105. DOI: 10.1177/1473871611424815 (cit. on pp. 64–66).

[Shn92] Ben Shneiderman. *Tree visualization with Tree-Maps: 2-D space-filling approach*. In: *ACM Trans. Graph.* 11.1 (1992), pp. 92–99. DOI: 10.1145/102377.115768 (cit. on p. 160).

[Smi+09] Marc Smith, Ben Shneiderman, Natasa Milic-Frayling, Eduarda Mendes Rodrigues, Vladimir Barash, Cody Dunne, Tony Capone, Adam Perer, and Eric Gleave. *Analyzing (social media) networks with NodeXL*. In: *C&T '09: Proc. 4th International Conference on Communities and Technologies*. 2009, pp. 255–264. DOI: 10.1145/1556460.1556497 (cit. on pp. 21, 23, 49, 95).

[Smi+10] Marc Smith, Ben Shneiderman, Natasa Milic-Frayling, Eduarda M. Rodrigues, Vladimir Barash, Cody Dunne, Tony Capone, Adam Perer, and Eric Gleave. *NodeXL: A free and open network overview, discovery and exploration add-in for Excel 2007/2010*. Social Media Research Foundation. 2010. URL: http://nodexl.codeplex.com (cit. on pp. 7, 10, 13, 16, 20, 26, 41, 45, 48, 49, 53, 68, 71, 72, 75, 95, 105, 138, 144, 160, 181, 225, 228, 230, 242, 247, 255, 269, 271–273, 289).

[Smi+13] Marc Smith, Ben Shneiderman, Natasa Milic-Frayling, Eduarda M. Rodrigues, Vladimir Barash, Cody Dunne, Tony Capone, Adam Perer, and Eric Gleave. *NodeXLGraph Gallery*. Social Media Research Foundation. 2013. URL: http://nodexlgraphgallery.org/ (cit. on pp. 213, 218, 225).

[SNA08] Pekka Salmela, Olli S. Nevalainen, and Tero Aittokallio. *A multi-level graph layout algorithm for Cytoscape bioinformatics software platform*. Tech. rep. 861. Turku Centre for Computer Science, 2008 (cit. on p. 41).

[SND05] Purvi Saraiya, Chris North, and Karen A. Duca. *An insight-based methodology for evaluating bioinformatics visualizations*. In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* 11.4 (2005), pp. 443–456. DOI: 10.1109/TVCG.2005.53 (cit. on p. 45).

[SP06] Ben Shneiderman and Catherine Plaisant. *Strategies for evaluating information visualization tools: Multi-dimensional in-depth long-term case studies*. In: *BELIV '06: Proc. 2006 AVI workshop on BEyond time and errors: novel evaLuation methods for Information Visualization*. 2006, pp. 1–7. DOI: 10.1145/1168149.1168158 (cit. on p. 45).

[Spo] *Spotfire*. spotfire.tibco.com (cit. on pp. 56, 286).

[SS06] Jinwook Seo and Ben Shneiderman. *Knowledge discovery in high-dimensional data: case studies and a user survey for the rank-by-feature framework*. In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* 12.3 (2006), pp. 311–322. DOI: 10.1109/TVCG.2006.50 (cit. on p. 46).

[STT81] Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. *Methods for visual understanding of hierarchical system structures*. In: *IEEE Transactions on Systems, Man and Cybernetics* 11.2 (1981), pp. 109–125. DOI: 10.1109/TSMC.1981.4308636 (cit. on pp. 249, 258).

[Sug02] Kozo Sugiyama. *Graph drawing and applications for software and knowledge engineers*. Vol. 11. Series on Software Engineering and Knowledge Engineering. World Scientific Publishing Company, 2002 (cit. on pp. 32, 33, 245, 249, 255).

[Tab] *Tableau*. www.tableausoftware.com (cit. on p. 56).

[TTT06] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. *The worst-case time complexity for generating all maximal cliques and computational experiments*. In: *Theoretical Computer Science* 363.1 (2006), pp. 28–42. DOI: 10.1016/j.tcs.2006.06.015 (cit. on p. 93).

[TTT12] Orestis Tsigkas, Olivier Thonnard, and Dimitrios Tzovaras. *Visual spam campaigns analysis using abstract graphs representation*. In: *VizSEC '12:: Proc. 9th International Symposium on Visualization for Cyber Security*. VizSec '12. 2012, pp. 64–71. DOI: 10.1145/2379690.2379699 (cit. on p. 36).

[Wal01] C. Walshaw. *A multilevel algorithm for force-directed graph drawing*. In: *GD '00: Proc. 8th International Symposium on Graph Drawing*. 2001, pp. 31–55. DOI: 10.1007/3-540-44541-2_17 (cit. on p. 41).

[War+02] Colin Ware, Helen C. Purchase, Linda Colpoys, and Matthew McGill. *Cognitive measurements of graph aesthetics*. In: *IVS: Information Visualization* 1.2 (2002), pp. 103–110. DOI: 10.1057/palgrave.ivs.9500013 (cit. on pp. 14, 46, 250, 256–258, 266, 268).

[War04] Colin Ware. *Information visualization: perception for design*. Morgan Kaufmann Publishers Inc., 2004 (cit. on pp. 33, 249).

[Wat06] Martin Wattenberg. *Visual exploration of multivariate graphs*. In: *CHI '06: Proc. SIGCHI Conference on Human Factors in Computing Systems*. CHI '06. 2006, pp. 811–819. DOI: 10.1145/1124772.1124891 (cit. on pp. 7, 29, 34, 148).

[WD08] Jo Wood and Jason Dykes. *Spatially ordered treemaps*. In: *TVCG: IEEE Transactions on Visualization and Computer Graphics* 14.6 (2008), pp. 1348–1355. DOI: 10.1109/TVCG.2008.165 (cit. on pp. 41, 44).

[Wel+07] Howard T. Welser, Eric Gleave, Danyel Fisher, and Marc Smith. *Visualizing the signatures of social roles in online discussion groups*. In: *JOSS: Journal of Social Structure* 8.2 (2007) (cit. on pp. 3, 38).

[Won+08] Pak Chung Wong, Harlan Foote, Patrick Mackey, George Chin, Heidi Sofia, and Jim Thomas. *A dynamic multiscale magnifying tool for exploring large sparse graphs*. In: *IVS: Information Visualization* 7.2 (2008), pp. 105–117. DOI: 10.1057/palgrave.ivs.9500177 (cit. on pp. 41, 283).

[WS79] Charles Wetherell and Alfred Shannon. *Tidy drawings of trees*. In: *IEEE Transactions on Software Engineering* SE-5.5 (1979), pp. 514–520. DOI: 10.1109/TSE.1979.234212 (cit. on pp. 33, 245).

[WS92] Christopher Williamson and Ben Shneiderman. *The dynamic HomeFinder: evaluating dynamic queries in a real-estate information exploration system*. In: *SIGIR '92: Proc. 15th annual international ACM SIGIR conference on research and development in information retrieval*. 1992, pp. 338–346. DOI: 10.1145/133160.133216 (cit. on p. 56).

[WT07] Ken Wakita and Toshiyuki Tsurumi. *Finding community structure in mega-scale social networks: [extended abstract]*. In: *WWW '07: Proc. 16th international conference on World Wide Web*. 2007, pp. 1275–1276. DOI: 10.1145/1242572.1242805 (cit. on pp. 140, 143, 147).

[Ye+05] Ping Ye, Brian D Peyser, Forrest A Spencer, and Joel S Bader. *Commensurate distances and similar motifs in genetic congruence and protein interaction networks in yeast.* In: *BMC Bioinformatics* 6 (2005), p. 270. DOI: 10.1186/1471-2105-6-270 (cit. on p. 37).

[YEL10] Ji Soo Yi, Niklas Elmqvist, and Seungyoon Lee. *TimeMatrix: Analyzing temporal social networks using interactive matrix-based visualizations*. In: *IJHCI: International Journal of Human-Computer Interaction* 26.11–12 (2010), pp. 1031–1051. DOI: 10.1080/10447318.2010.516722 (cit. on p. 23).

[ZCM05] Shengdong Zhao, Mark H. Chignell, and Michael J. McGuffin. *Elastic Hierarchies: Combining treemaps and node-link diagrams*. In: *INFOVIS '05: Proc. IEEE symposium on Information Visualization*. INFOVIS '05. 2005, pp. 57–64. DOI: 10.1109/INFVIS.2005.1532129 (cit. on p. 26).

[ZGS07] Xiaowei Zhu, Mark Gerstein, and Michael Snyder. *Getting connected: Analysis and principles of biological networks.* In: *Genes Development* 21.9 (2007), pp. 1010–1024. DOI: 10.1101/gad.1528707 (cit. on p. 37).