VISUALIZING MULTIVARIATE NETWORKS

by

Carolina Nobre

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computing

School of Computing

The University of Utah

May 2020

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Carolina Nobre**
has been approved by the following supervisory committee members:

Alexander Lex ,	Chair(s)	**Feb 19, 2020** Date Approved
Miriah Meyer ,	Member	**Feb 19, 2020** Date Approved
Jason Wiese ,	Member	**Feb 19, 2020** Date Approved
Nils Gehlenborg ,	Member	**Feb 19, 2020** Date Approved
Marc Streit ,	Member	**Feb 19, 2020** Date Approved

by **Ross Whitaker** , Chair/Dean of
the Department/College/School of **Computer Science**
and by **David B. Kieda** , Dean of The Graduate School.

ABSTRACT

This dissertation contributes an in-depth treatment of multivariate network (MVN) visualization techniques and their evaluation. A multivariate network is one in which the nodes and/or edges are associated with attributes. This type of network is widespread, with examples including social networks, physical networks such as power grids, networks modeling cellular processes in biology, and trees describing evolutionary relationships between species. The need for visualizing MVNs arises when the structure (the topology) of the network needs to be analyzed together with the node or edge attributes. Such analysis presents a challenge when visualizing topology and attributes in the same view, since choosing efficient encodings for one aspect often interferes with the ability to effectively visualize the other.

Our work contributes to the space of multivariate network visualization by first organizing the design space of MVN visualization techniques into a typology and identifying the limits of existing approaches. Given this landscape of techniques, we make two technique contributions: 1) an applied design study with domain experts that explores visualizing attributed genealogies; and 2) a general MVN visualization approach that addresses the challenge of simultaneously visualizing topology and attributes. We also contribute an empirical study that provides experimental evidence concerning the performance of the two most commonly used MVN visualization techniques: node-link diagrams and adjacency matrices. This body of work will provide guidance for practitioners, visualization researchers, and domain experts using MVNs for real world exploration.

For Mom

CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to extend my deepest gratitude to my advisor, Alex Lex. Alex has been my mentor since my very first steps into the visualization world several years ago, and it would be no exaggeration to say that I could not have done this without him. I would also like to thank my supervisory committee: Alexander Lex, Miriah Meyer, Jason Wiese, Nils Gehlenborg, and Marc Streit. Working with them has been a great honor.

I would like to thank the people who contributed to this dissertation as coauthors: Miriah Meyer, Nils Gehlenborg, Marc Streit, Hilary Coon, Lane Harisson, and Dylan Wootton. A special thanks to Dylan, who gave 110% to our user study paper, and showed me how incredibly rewarding it was have a partner in crime during those late nights at the office.

I am also grateful for the feedback and support of the members of the Vis Design Lab at the University of Utah: Ethan Kerzner, Dylan Wootton, Alex Bigelow, Sean McKenna, Nina McCurdy, Jimmy Moore, Jennifer Rogers, Sam Quinan, Kiran Gadhave, and Haihan Lin. Special thanks to Christine Pickett because her feedback helped me to write and to think clearly about this dissertation.

This dissertation extends published work for which we have permission to re-use. Reprinted [1], with permission, Nobre, Carolina and Gehlenborg, Nils and Coon, Hilary and Lex, Alexander; "Lineage: Visualizing Multivariate Clinical Data in Genealogy Graphs;" *Transaction on Visualization and Computer Graphics*; © 2019 IEEE. Reprinted [2], with permission, Nobre, Carolina and Streit, Marc and Lex, Alexander; "Juniper: A Tree+Table Approach to Multivariate Graph Visualization;" *Transaction on Visualization and Computer Graphics (InfoVis 18)*; © 2019 IEEE. Reprinted [3], with permission, from Nobre, Carolina and Meyer, Miriah and Streit, Marc and Lex, Alexander; "The State of the Art in Visualizing Multivariate Networks;"*Computer Graphics Forum*; © 2019 John Wiley and Sons. Reprinted [4], with permission, from Nobre, Carolina and Wootton, Dylan and Harrison, Lane and Lex, Alexander; "Evaluating Multivariate Network Visualization Tech-

niques Using a Validated Design and Crowdsourcing Approach;" *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, to be published; © 2020 SIG.

This dissertation was supported in part by the Utah Genome Project and the DoD Office of Economic Adjustment (OEA), ST1605-16-01. We also acknowledge support from the Multinet project.

CHAPTER 1

INTRODUCTION

This dissertation provides an in-depth treatment of multivariate network (MVN) visualization techniques and their evaluation. This includes an overview of existing approaches to MVN visualization, two novel MVN visualization techniques, as well as a user study comparing the two most common MVN visualization approaches: node-link diagrams and adjacency matrices. Multivariate networks are those that contain information about both the structure of the network — i.e, the topology — and attributes associated with the nodes and edges. These attributes can be of several different data types, such as numeric or descriptive.

Most real world networks are multivariate, with several examples in fields such as biology, transportation, and software engineering. Biologists, for example, need to explore canonical pathways in the context of experimental data, to judge whether a pathway is valid for a given tissue or organisms. To this end, they analyze the structure of the network, mapping how entities are connected to form pathways, and inspect the attribute information along these paths, such as metabolite concentration.

Another common real world application of MVNs is social network analysis. In this field, nodes represent social entities, such as people, and edges capture the relationships between them. Additionally, some social networks can contain heterogeneous nodes and edges, representing distinct entities with unique sets of attributes. Social scientists analyze these networks to better understand the behavior of social entities and how they interact with each other. For example, an analyst may be interested in whether a tight group of friends are all in the same age group, and whether they often interact with people who went to a neighboring school. Social networks are often large and topologically complex, making them challenging to visualize. Additionally, both the nodes and links can be associated with several attributes of different types.

In software engineering, nodes represent software components such as classes and functions, and edges represent the association between nodes. Attributes often include computed metrics such as the number of lines in a function or how often a class is called. Similar to MVNs in the fields of biology and social sciences, these networks can be very large, with more node and edge attributes than can be easily visualized at once. Analysis tasks can include finding bottlenecks in the code, or better understanding how network components influence the output.

As the examples above show, analyzing multivariate networks often relies on understanding both the relationships in the network and the attributes of the nodes and edges. This process of network analysis can be effectively supported by well-designed visual representations. In this respect, network visualization research is aimed at generating effective network layouts and visual mappings [5]. A closely related field is that of graph drawing, which draws on methods from geometric graph theory and information visualization to produce optimized placement of the nodes and edges in a graph layout. MVN visualization builds on the field of graph drawing by also accounting for attributes and how they can be visualized to best support MVN analysis tasks.

A common challenge with visualizing networks is the limited number of nodes and edges that can be effectively visualized at once. Network visualizations can quickly become overly cluttered and unreadable when the size of the network exceeds a few hundred nodes. This lack of scalability is exacerbated for multivariate networks, where rich attributes for the nodes and/or the links need to be considered as well.

The main challenge with visualizing multivariate networks, however, arises from two conflicting goals that need to be reconciled: visualizing topology and visualizing node and edge attributes. The visualization community has a good understanding of how to visualize either the topology of a network or the multidimensional data that are associated with the nodes and edges, yet addressing both topology and attributes at the same time is still an open research problem. Given the limits of human perceptual and cognitive abilities, MVN visualizations cannot simply display all the information at once. They must carefully choose which aspects of the network to prioritize. Effective techniques often take into account the specific analysis tasks, and provide interaction to dynamically prioritize different aspects of the network. The challenges with visualizing MVNS can

be further exacerbated when the network is particularly large or when the number of attributes increases.

1.1 Contributions

In order to address the challenges above, this dissertation makes the following contributions:

- **Typology of MVN visualization techniques and taxonomy of MVN analysis tasks**
  In order to organize the design space of existing approaches to this challenge, we conducted a survey of MVN visualization techniques, which resulted in a typology of MVN visualization layouts as well as visual and data operations that are commonly applied when visualizing MVNs [3]. From the corpus of papers in this survey, we also gained a better understanding of MVN analysis tasks and created a taxonomy of MVN tasks that builds on existing general network exploration tasks to account for the presence and importance of attributes [3].
- **Technique for visualizing attributed genealogies**
  Lineage [1] is a design study that resulted from a collaboration with psychiatrists at the University of Utah. Lineage provides a solution that addresses their analysis needs in a layout which juxtaposes a linear layout of the genealogy with a juxtaposed table of node attributes. This configuration provides tight integration between the topology of the network and the attributes of the nodes and edges, therefore supporting both topology and attribute based tasks.
- **Tree-based technique for visualizing general MVNs**
  In an effort to generalize the Lineage approach, which was designed for tree-like networks, we also contribute a general network visualization approach with the *Juniper* technique [2]. The core idea behind Juniper is to extract a spanning tree from a network and visualize a linearized version of this tree alongside a juxtaposed table of node and/or edge attributes. The Juniper technique expands the current design space of visualizing MVNs, particularly for integrated layouts.
- **User study comparing node-link diagrams and adjacency matrices**
  We carried out a formal user study comparing the two most prevalent forms of MVN visualization: node-link diagrams and adjacency matrices [4]. This study

generated the first empirical evidence on the performance of these techniques for MVNs as well as a novel method for evaluating complex interactive visualization with crowdsourcing.

1.2 Organization

This dissertation summarizes related work and then presents our work in the following order: theory, techniques, and evaluation. In Chapter 2, we provide definitions for different types of networks, outline the major network visualization approaches and their suitability to different network types, summarize how MVNs are used in different application areas, and outline the most common forms of evaluating MVNs. In Chapter 3, we discuss multivariate network exploration tasks, including both existing approaches to MVN exploration and a novel taxonomy of tasks that explicitly accounts for both the topology and the attribute components. Chapter 4 presents the typology of multivariate network visualization techniques, derived from the analysis of publications for our state of the art report on visualizing MVNs. In Chapter 5, we describe our design study with psychiatrists and the resulting tool *Lineage*. Chapter 6 presents *Juniper*, a more general technique we developed for visualizing MVNs. In Chapter 7, we report on our crowdsourced study comparing node-link diagrams and adjacency matrices, including a detailed description of our validated design approach to conducting crowdsourced studies for complex interactive visualizations. In Chapter 8, we conclude with a broader discussion on the main contributions of this dissertation and identifying areas of future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, we summarize the most prevalent approaches to network visualization and their suitability to different network types. We discuss the main ways in which attributes are encoded but leave a more detailed and formal analysis of the design space of attribute encoding to our typology of MVN visualizations in Chapter 4. Then, we provide a summary of the major application areas for multivariate networks in the real world. Lastly, we provide an overview of the evaluation methods currently used in MVN visualization. We preface this chapter with terminology and definitions for networks.

2.1 Definitions and Terminology

The term *network* is widely used, and can have different meanings depending on the context. For the purpose of this dissertation, we define a *network*, or a *graph*, as $G = (N, E)$ with nodes $n \in N$ and edges $e \in E$, which can be of different types $t \in T$. A *multivariate* network is one in which the nodes or edges, or both, have attributes associated with them. We refer to the nodes and edges as the network's *topology*. The topology of the network captures its structure, or how the nodes are connected to each other. Attributes on the nodes and edges can be of varying data types, such as textual, quantitative, ordinal, nominal, or set data. Attributes can also be computed based on topological properties. For example, the degree of a node, or its centrality, is a common derived attribute. Analogously, topological structures can be derived from attributes. For example, two nodes can be connected if they share an attribute.

Both nodes and edges can be of different types. We consider nodes and edges of a type to be of a semantic unit that has a different set of attributes associated with it. For example, a node type *actor* could have a name, current residence, age, etc. associated with it. Another node type in the same network could be *movies*, with attributes such as movie title, budget, revenue, genres, etc.

We distinguish between the following network types:

- **Complex networks**, i.e., networks with nontrivial topological features, such as a diverse degree distribution (the distribution of the number of links per node), clusters, hubs, etc. [6]. Among the complex networks are scale-free networks, with a degree distribution that follows the power law distribution. These networks are characterized by a small number of nodes with a very high degree that exhibit preferential attachment, i.e., that nodes added to the network are likely to attach to these central hubs. An example is the world wide web, where major websites are linked to much more frequently than others. Another type of complex network is a **small world network**, with a degree distribution that peaks around the average value. These networks exhibit significant clustering and are characterized by a short average path distance, hence small world. Social networks are a typical example.
- **Layered and K-partite networks** are networks in which the nodes can be partitioned into discrete sets and in which links are mostly (in the case of layered networks) or exclusively (in the case of k-partite networks) between the different sets. Layered networks are ordered (e.g., as in genealogies, where there is an order of generations that is represented as layers) and have links only between successive layers, whereas k-partite networks can have links between multiple partitions and are not necessarily ordered.
- **Trees** are acyclic networks, commonly with a defined root, for which each node is either a leaf or the root of a subtree. Trees have exactly one path between two nodes and have a low average degree.

For complex networks, we also distinguish between *sparse* networks, with a low average degree, and *dense* networks, with a high average node degree. Layered networks, k-partite networks, and trees are sparse by their nature.

2.2 Network Visualization

In the following sections, we outline the two main approaches to network visualization: node-link diagrams and adjacency matrices, including how attributes are most commonly encoded with each technique. We also describe tree-specific approaches, since they provide the background for both technical contributions in this dissertation (Chapters 5

and 6). We provide a more detailed treatment of all approaches to multivariate network visualization in Chapter 4, including reference implementations and guidelines for when each approach is most appropriate.

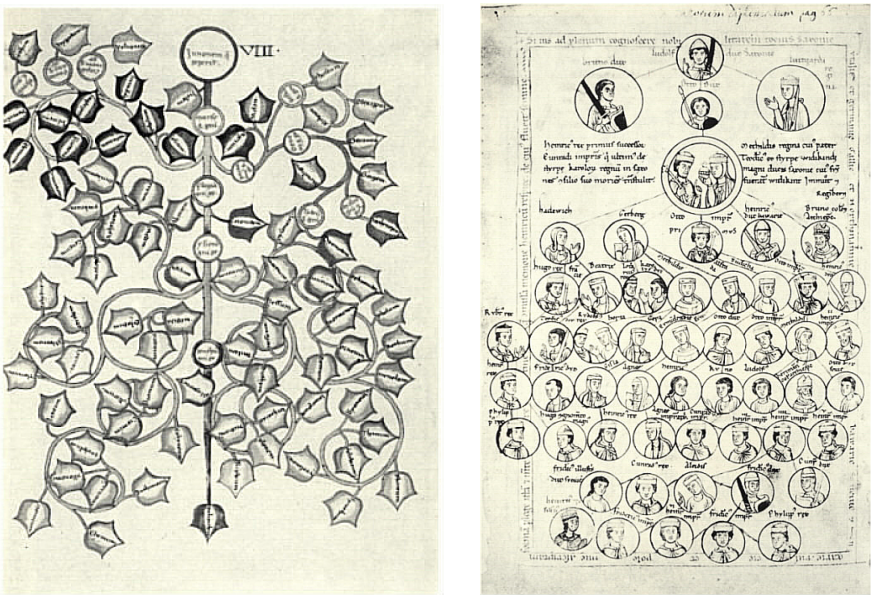
2.2.1 Node-Link Diagrams

Unlike the origin of charts for tabular data — such as line charts and bar charts — whose early use in the works of William Playfair are well documented, the exact origins of network drawings are not well known. The earliest forms of network drawings are gameboards in stone carvings that date back to the 13th century [7]. These drawings predate the establishment of graph theory in 1735 by the Swiss mathematician Leonhard Euler. Euler is credited with solving the first theorem in graph theory when he solved the Knigsberg bridge problem, a puzzle concerting the possibility of finding a path over seven bridges, without crossing any bridge twice. Even though Euler's solution did not directly reference the components of a graph, he did prove that no such path exists, and is thus credited with originating graph theory.

The early drawings of gameboards from the 13th century were in the form of node-link diagrams, where the nodes are drawn as point marks and the links as line/curve marks connecting the nodes (Figure 2.1(a)). Aside from the game boards, the other common form of ancient network drawings was in the form of family trees [8]. Figure 2.1(b) shows early recorded genealogies that appeared in manuscripts from the Middle Ages [7]. A selection of examples of these early family tree drawings can be found in Klapisch-Zuber [9]. A more thorough history of how trees have been visualized with node-link diagrams over the centuries can be found in *The Book of Trees* by Manuel Lima [10]. The advent of modern network drawing happened in the late 18th century and early 19th century, when node-link representations began to appear more frequently in varied contexts, such as mathematics, engineering, and chemistry [7]. For example, mathematicians started using graph drawings for problems involving path tracing, and in chemistry graph drawings were used to represent the structures of crystals [7].

Node-link diagrams are by far the most common graphical representation of networks. They are the subject of their own field of study — graph drawing — and countless algorithms for node-link layouts have been developed [11]. In order to quantify the aesthetics

(a) Morris gameboards



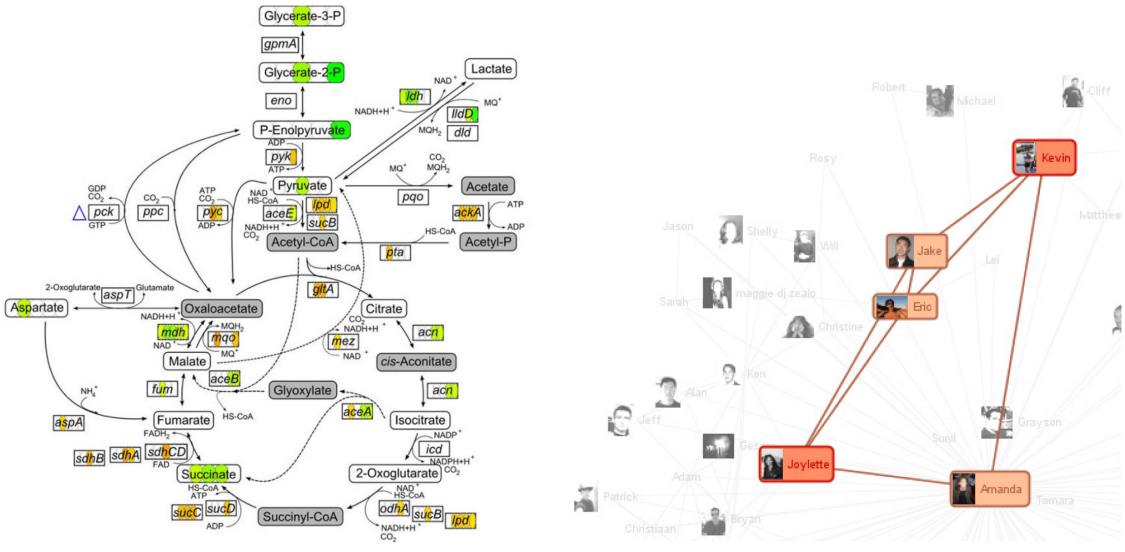
(b) Ancient family trees

Figure 2.1. Historic examples of node-link diagrams: (a) Morris gameboards from the 13th century. The nodes of the graph drawings are the positions that game counters can occupy and edges indicate how game counters can move between nodes. (b) Family trees that appear in manuscripts in the Middle Ages [7].

and usability of a graph drawing, different quality measures can be used, such as number of edge crossing, symmetry, and area of the drawing, among others. For example, as a general rule, an optimized graph drawing will minimize the number of edge crossings, although avoiding them altogether is often impossible depending on the structure of the graph. Also in regard to edges, shorter edge lengths are commonly viewed as easier to follow and interpret than longer ones. Additionally, it may be preferable for the lengths of edges to be uniform rather than highly varied. Ensuring uniform edge lengths becomes increasingly challenging as the size of the network increases, a problem that is compounded when attributes of the nodes and edges must also be visualized. Different graph layout

strategies, such as force-directed or orthogonal, prioritize different metrics depending on the goal of the visualization. Force-directed layouts, for example, favor a layout in which edges are short, while keeping nodes distinct and well separated. Orthogonal layouts allow the edges to run only horizontally or vertically to the coordinate axes of the layout. HOLA is a type of orthogonal network layout method that leverages a human-centered approach to produce layouts of comparable quality to that of a human. Layered layouts place nodes into layers and thus clearly separate edges that connect adjacent layers.

Encoding attributes with node-link diagrams is most often done through **on-node encoding** (Figure 2.2), which refers to modifying the visual appearance of a node or embedding marks in it. Color coding is a common choice to encode a single data value or a node type; the latter is also often encoded using node shapes or icons. Gehlenborg et al. [14] review techniques used in systems biology for visualizing multivariate networks, many of which make use of on-node encoding using embedded charts, such as line charts, box plots, etc. On-node encoding is also widely supported by common network visualization tools such as Cytoscape [15] and Gephi [16]. Van den Elzen and van Wijk [17] use embedded visualizations to show distributions of values aggregated in a super-node. On-node encoding supports the integration of topology and attribute-based tasks well; however,



(a) (b)

Figure 2.2. On-node encoding. (a) Multiple attributes (metabolite concentrations) are encoded directly on the nodes using color [12]. Figure licensed under CC BY 2.0. (b) Photos and labels are encoded on the nodes of a social network [13].

it comes with scalability trade-offs. Even for a modest number of nodes in a node-link layout, node size has to be limited; hence little space is available to encode attributes. When details about nodes are shown, as, for example, in MoireGraphs [18], the number of nodes that can be displayed simultaneously is limited.

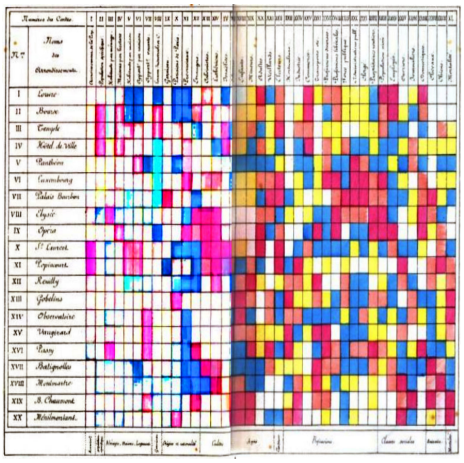
Complex networks, as defined in Section 2.1, are often visualized with node-link diagrams, particularly if they are sparse and have at most a few hundred nodes. Complex networks that are larger or more dense quickly exceed the limitations of the node-link representation and are better represented by adjacency matrices, which are described in the following section. Sparse layered and k-partite networks can be visualized with node-link diagrams where nodes are grouped by their layer and edges are drawn between adjacent layers. Trees are also inherently well suited for node-link diagrams since they are sparse by definition and can guarantee a layout with no edge crossings.

Although node-link diagrams are the oldest and most prevalent form of visualizing networks, other approaches such as matrix based or tree-specific layouts have emerged as alternatives, each providing optimized encodings for certain network types.

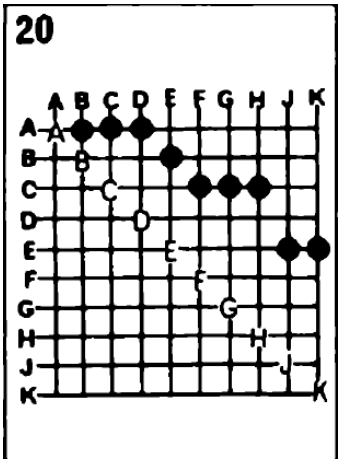
2.2.2 Adjacency Matrices

The concept of an adjacency matrix has its origin in graph theory, and is defined as a matrix whose elements indicate whether pairs of vertices are connected in a network. Adjacency matrices have several similarities to tabular structures since they are both composed of rows and columns, which can then support nested visualizations within the cells. The idea of encoding cell values visually dates back to the late 19th century, with *shaded matrices* or *heat maps* used to encode values in tabular data (Figure 2.3(a)) [19].

A key component of heat maps, and by extension adjacency matrices, is the idea of reordering rows and columns to reveal patterns. This practice also dates back to the late 19th century, when the English Egyptologist Flinders Petrie reordered strips of paper to reconstruct the chronology of excavated graves [21]. The use of tabular visualizations was popularized by Jacques Bertin [20]. Bertin also leveraged the use of matrix visualizations to show the structure of an underlying network [20], as seen in Figure 2.3(b) from the seminal book *Semiology of Graphics* [22]. Although adjacency matrices are used to represent network structures, the technique leverages the same visualization techniques used for tabular data,



(a) Shaded matrix display from Loua (1873).

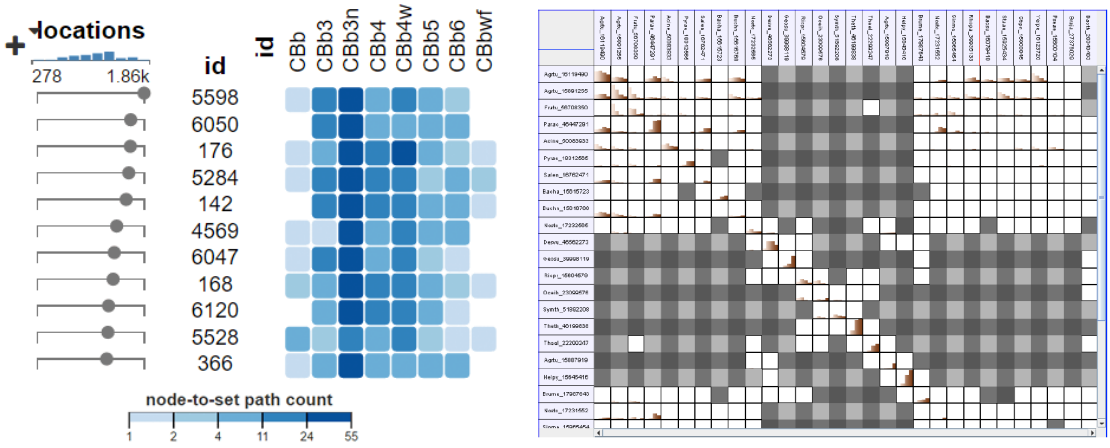


(b) Adjacency matrix for network visualization from Bertin [20]

Figure 2.3. Early tabular and adjacency matrix visualizations. (a) Shaded matrix display from Loua (1873). This display represents a summary of 40 separate maps of Paris, showing the attributes of 20 districts, using a color scale that ranged from white (low) through yellow and blue to red (high). [19] (b) Early example of a network visualized as an adjacency matrix.

with the only difference being the semantic interpretation of the cells, as edges between adjacent vertices.

Adjacency matrices have both favorable and unfavorable properties compared to node-link layouts when judging topology [23]. Various attempts have been made to combine node-link layouts with matrices to find a compromise between these trade-offs. Examples are NodeTrix [24], which embeds adjacency matrices for subgraphs of a node-link layout, and MatLink [25], which enhances matrices with links. For attribute visualization, however, adjacency matrices are superior to node-link diagrams. For example, adjacency matrices can naturally encode edge attributes in matrix cells, as seen in Figure 2.4. Although this is mostly done with a single color value (Figure 2.4(a)), multiple edge attributes can be visualized as nested networks [26] (Figure 2.4(b)). Similar to the on-node encoding in node-link diagrams, however, the small space available for a matrix cell limits how much can be encoded. For node attributes, in contrast, it is easy to juxtapose multiple attribute visualizations with the rows or columns of the matrix, which has been done, for example, in Graffinity [27] (Figure 2.4(a)) and in MapTrix [28]. Adjacency matrices are well suited for dense complex networks, since their encoding ensures space for all edges with no overlap. However, they also suffer from a few disadvantages, including the fact they are not as



(a) (b)

Figure 2.4. Adjacency matrix approaches. (a) Connectivity matrix approach where cells represent total path counts between nodes [27]. Image courtesy of Ethan Kerzner. (b) Aggregated matrix represented by an adjacency matrix where aggregated edge attributes are displayed in the cells [26].

well known as node-link diagrams. As a result, users must become acquainted with the encodings to take full advantage of the visualization technique. Additionally, adjacency matrices are particularly ill-suited for path tracing tasks, as shown in several user studies comparing node-link diagrams and adjacency matrices [4], [23], [29]. Trees and layered networks can technically be visualized with an adjacency matrix, but the sparsity of these networks suggests that they are not a good fit.

2.2.3 Tree Layouts

Trees are a unique type of network, since they guarantee certain topological characteristics, such as containing only one parent per node. Trees can be visualized with both explicit and implicit layout techniques. Explicit layouts are those that encode both nodes and edges, such as the node-link diagram. Implicit layouts, on the other hand, encode the relationships between nodes only by their relative positions, and hence edges are only implicitly represented.

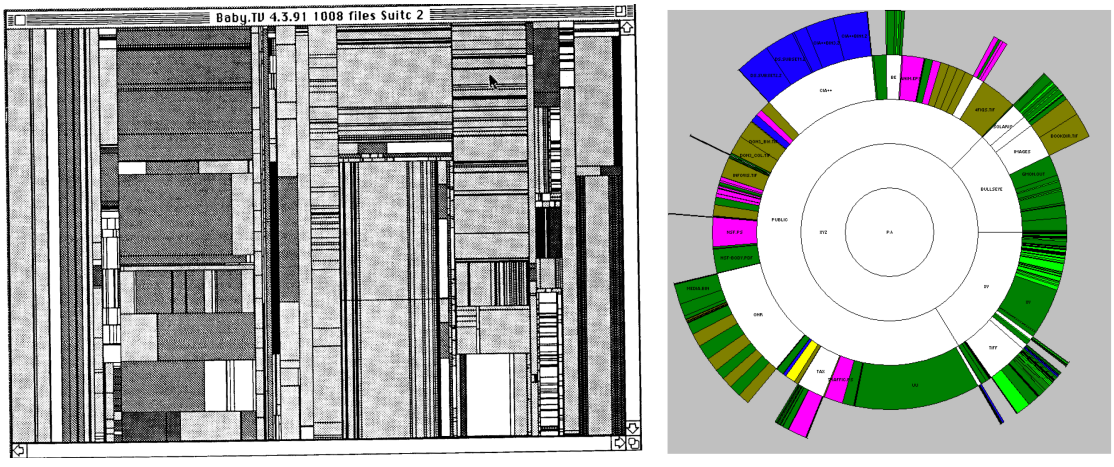
For explicit tree layouts, the on-node mapping strategies discussed in the previous section can be used to visualize attributes (e.g., [30]). Another common example for tree visualizations associated with many attributes is the use of a dendrogram tree derived from a hierarchical clustering algorithm that is aligned to a heat map [31]–[33]. Similar

approaches have been used for visualizing phylogenies and attributes about the species they contain [34], [35] or transactions associated with a hierarchy [36]. Surprisingly few techniques also visualize attributes for inner nodes in a tree. One example is a tree-table as it is used, e.g., in file browsers, showing properties such as file types and file/directory sizes.

Implicit layouts, such as tree maps [37], sunburst plots [38], or icicle plots [39], are more recent than the node-link diagram, and are designed specifically to represent hierarchical data. They are well suited to visualize one or two attributes of nodes in trees (using size and color), but they do not scale to more attributes.

With treemaps, each level in the tree is assigned a rectangle, which is then tiled with nested rectangles for the lower levels (Figure 2.5(b)). For certain variants of treemaps, the leaf node's rectangle can be sized and/or colored proportionally to a given attribute. The treemap technique was introduced by Ben Shneiderman in 1991 [37], but was only popularized in the late 90s and early 2000s when the technique was improved to allow for improved aspect ratios of the rectangles, and more semantically meaningful ordering of the nodes [40].

A closely related technique is the sunburst diagram, which uses a radial rather than a rectangular layout to represent hierarchical data (Figure 2.5(a)). With this approach, items



(a) Treemap (b) Sunburst

Figure 2.5. Implicit tree visualizations (a) Early example of the treemap technique, showing a collection of 1000 files [37]. (b) Sunburst visualization showing the file structure in a directory [38].

in a hierarchy are laid out radially, with the top of the hierarchy at the center and deeper levels farther away from the center. Attributes can be encoded in the angle of the item or its fill color. Stasko and Zhang [38] coined the term "sunburst diagram." The technique builds on the work of Andrew and Heidigger [41] who used cascading, semicircular discs as an approach to visualizaing hierarchical data.

A broad overview of tree visualization techniques can be found in the tree visualization reference by Schulz [42].

2.3 Application Areas for Multivariate Networks

MVNs are a prevalent data type in several domains, including digital humanities [43], oceanography [44], system analysis [45], and engineering [46]. The most common application areas are those of social network analysis, biological applications, and software engineering. In this section, we discuss these application areas and give a brief overview of others. We do not, however, claim an exhaustive review of application areas since the range of journals and conferences where application papers can be published is too broad for the scope of this dissertation. A survey of application areas for MVNs, also focusing on social networks, biology, and software engineering, can be found in the book edited by Kerren et al. [47].

2.3.1 Social Network Analysis

Social networks are a popular and widely available source of data. Social networks are defined as a network whose vertices are social entities such as people or organizations, with edges capturing relationships between them. The field of social network analysis (SNA) is concerned with finding structural properties in the network and analyzing the associated attributes of the nodes and edges [48], [49]. Social networks often contain a multitude of attributes on both nodes and edges. This large number of node and edge attributes represents one of the major challenges with visualizing social networks. Additional attributes are frequently derived using SNA algorithms, which compute additional topology-related attributes such as degree, centrality, and clustering coefficients.

Visual analytics systems in this area can leverage multiple coordinated views, one for each aspect of the network. Since social networks are commonly categorized as a small

world network [49], several systems in this field use adjacency matrices for their ability to support dense and highly connected portions of the network [25], [50]–[52]. Examples of such approaches include the work by Henry et al. in both MatLink [25] and MatrixExplorer [50], which use adjacency matrices in either hybrid or juxtaposed MCV layouts. MatLink leverages an overloaded node-link and matrix layout to address the trade-offs between using these two opposing layouts for locally dense structures such as those found in social networks. Their overloaded layout consists of an adjacency matrix with edges connecting the nodes overlaid on top to allow for easier path following. This modification is particularly useful in addressing path-related topology tasks, a shortcoming of tabular approaches.

Similar to MatLink, NodeTrix [24] leverages two topology encodings, but in a hybrid layout. Their design displays the network with a node-link layout while replacing dense and highly connected areas with embedded adjacency matrices. NodeTrix also allows the user to encode network attributes for both nodes and edges using the following visual channels: color, transparency, shape size, filled area of the shape, border color, width, and labels.

Despite the prevalence of adjacency matrices for SNA visualization, several MVN approaches use node-link diagrams to represent the topology of the network [13], [53], [54]. GraphDice [54] displays a grid of small multiples where each multiple contains an attribute-driven node-link with two attributes, thereby giving the user an overview of how several attributes vary throughout the network. Vizster [13] displays a node-link view for the topology of the network, linked with a juxtaposed attribute panel that shows details on demand for nodes of interest.

Combining multiple ways of encoding attributes in a single view can also be effective, an example of which can be found in the paper by Ghani et al. [55]. The system uses coordinated views, one of which is an instance of *parallel node-link bands*. This view leverages attributes in three ways: a faceted node-link diagram partitions the nodes into bands according to their attribute values for categories of interest; within each band, the user can order the nodes according to either node or edge attributes; and on-node encoding is used to represent an additional attribute, such as the previously computed degree centrality of each node.

2.3.2 Biological Applications

Biological networks are another common area of applications for MVN visualizations. Networks in this space are most often characterized by nodes and edges with complex attributes, with several examples in fields such as genetics, cancer research, and systems biology.

Unlike the prevalence of adjacency matrices in social network visualizations, MVNs in biological applications rely heavily on node-link layouts [1], [14], [56]–[58]. These layouts are often optimized for displaying paths, since several tasks in this area relate to understanding how attributes vary along biologically meaningful cascades. As a result, work such as that done by Schreiber et al. [59] and Karp and Paley [60] has focused exclusively on providing layout algorithms for node-link diagrams of biological networks that focus on displaying paths in an intuitive manner. These optimized layouts are particularly useful for MVN tasks that target either topology-driven or attribute exploration of biological pathways.

Techniques that focus on path-related tasks include Entourage [57], Pathline [61], Pathfinder [62], and Enroute [58]. Entourage and Enroute allow the user to highlight a path of interest in a network, which is then juxtaposed with an attribute view to create a single integrated view. In a similar vein, Pathline linearizes a pathway network, which is then juxtaposed with an attribute table.

In a survey of existing techniques for visualizing omics data in systems biology, Gehlenborg et al. [14] highlight the prevalent use of node-link diagrams, often with on-node encoding or attribute-driven layouts. Among the examples given in their survey is Cerebral [63], which uses both small multiples and multiple coordinated views to display the topology and attributes of the network. The node-link view facets the nodes according to their locations in the cell and is linked to a small multiples view and a parallel coordinate view to display the associated attributes. In Lineage, Nobre et al. [1] linearize a node-link representation of genealogies and juxtapose an attribute table to create a single integrated view of the topology and attributes of the network.

New et al. [64] describe an example of linked views that includes an adjacency matrix instead of the more commonly used node-link diagram. The authors introduce a tilted adjacency matrix that permutes rows and columns of a matrix in such a way as to cluster

nonzero elements in blocks along the diagonal.

2.3.3 Software Engineering

Visualization of MVNs in software engineering is part of the broader area of software visualization [65]. Software visualization refers to visual depictions of any component of the software lifecycle, from the source code itself to the associated documentation, mental models, and output data [66]. The survey by Diehl and Telea [66] gives an overview of multivariate network visualization as applied to software engineering.

Within the context of multivariate networks, software visualization often models networks with nodes representing entities such as files, classes, functions, or other components of software. Edges encode the relationships between these entities and can either represent a hierarchical relationship, such as files in a folder or functions in a class, or simply reflect the association between nodes, such as function calls or data flow between nodes [66]. Attributes can be diverse and include computed software metrics such as lines of codes, numbers of classes, number of calls, or runtime in a specific module or function, which make up the multivariate node and edge attributes of these networks [67].

Similar to MVNs in the field of biology, networks in software engineering are often very large and contain several attributes, and therefore require particular attention to scalability [66]. Another defining characteristic of networks in this field is the large variety of attribute types, since nodes and edges can represent diverse entities software components, data, people, files, etc.

Visualizing software engineering data has been done in a myriad of ways, including with UML diagrams [68], treemaps [69], tabular visualizations [70], icicle plots [71], and multiple coordinated views [69], [71]. Adjacency matrices, although less common, have been used to compare the hierarchies of two different systems [72].

UML diagrams are node-link layouts with relevant attributes encoded directly on the node marks in the form of text, glyphs, or small embedded visualizations. The often hierarchical nature of structure data is well suited to the combined use of node-link layout and overloading techniques, an approach optimized to encode group membership or hierarchical relations between elements [73].

Treemaps are often used to convey the hierarchical nature of the data, e.g., from pack-

ages to classes, to functions, to lower level control structures. The size of the node represents attributes such as the lines of code underlying the structure. Other attributes in treemaps are usually encoded using color on the treemap nodes, but 3D objects have also been used [74].

Another network datatype in the context of software engineering is versioning data. Visualization approaches are used to track the evolution of software components (the nodes) between different versions [75].

2.3.4 Other Application Areas

Other areas where MVN visualization techniques are used include transportation networks, communication networks, and security.

2.3.4.1 Transportation Networks

In transportation networks, nodes often represent locations, such as countries, cities, or intersections, and edges represent either connectivity between these nodes (roads of flight routes), or actual movement of people or goods between the nodes. Node attributes are commonly properties of the locations, such as the number of inhabitants of a city; edge attributes are frequently distance, travel time, or number of people traveling between locations. Edge attributes are typically encoded using color [76]. Due to their geospatial nature, transportation networks are often visualized on maps in the form of node-link diagrams, resulting in a fixed layout of the nodes. This fixed layout exacerbates the problem of edge clutter. Edge bundling techniques can improve readability by aggregating edges with common sources and targets, thus making higher level flow patterns visible [77]. By aggregating areas, it is possible to reduce the scale of the network and display sophisticated visualizations embedded on the nodes [17]. Certain transportation networks, such as subway maps, often do not use precise geolocations and instead preserve only approximate positions and the relative ordering of nodes [78]. An example of attribute-driven positioning used in transportation networks is "isotime flow maps," where distances between nodes encode the travel times between them [79].

2.3.4.2 Communication Networks

Communication networks are concerned with the flow of information between devices or people. Similar to transportation networks, the most prevalent form of MVN visualization within communication networks is node-link diagrams [80], [81]. Geolocated node-link representations are common in cases where the precise location of the nodes is of interest [82], [83], but many communication networks are laid out differently, for example using attribute-driven positioning to group similar nodes [80].

2.3.4.3 Security Networks

The field of security visualization is frequently concerned with the analysis and visualization of security networks, i.e., networks aimed at detecting anomalous patterns that may indicate vulnerability and attacks [84]. These networks are often communication networks. The difference between these networks and generic communication networks lies mostly in the types of attributes and analysis tasks, which are geared toward capturing unexpected behavior and traffic patterns. The types of visualizations used are varied and include force-directed node-link layouts [85], [86], node-link layouts as parallel axis [87], and matrices [88].

In general, we have noticed that social network analysis, biological and life science applications, and software engineering use a variety of different encodings, but other domains rely heavily on node-link diagrams. We speculate that this difference is because visualization research papers are often case studies for the former domains and therefore see a wider variety of techniques.

2.4 Evaluation of Multivariate Networks

In this section we report on the types of evaluation methods used for MVN visualization techniques, as encountered in the corpus of papers surveyed for our state-of-the-art report on MVNs [3]. Although an in-depth analysis of how MVN visualization techniques are evaluated is beyond the scope of this dissertation, this survey of evaluation methods allowed us to understand how multivariate network techniques are currently being evaluated, and to, we hope, open up a conversation on whether these methods of evaluation are sufficient and appropriate. This overview of evaluation methods also serves to ground

the discussion on the evaluation approaches used in the contributions of this dissertation. Chapter 5 elaborates on the use of a *case study* as the evaluation for our design study. We describe a selection of *use cases* in Chapter 6. Our crowdsourced study in Chapter 7 elaborates on the challenges of *controlled experiments* as an evaluation approach, including a proposed methodology to address them.

The types of evaluation we consider and our interpretation of each method are as follows:

- **Use cases** are informal demonstrations of usage, without the involvement of a domain expert or any quantitative measure of the tool's validity.
- **Controlled experiments** are those that control some of the analysis process and include crowdsourced studies.
- **Case studies** are a form of evaluation that reports in detail on domain experts' use of the proposed tool in the wild.
- **Theoretical evaluation** argues for the efficacy of a proposed technique based on existing visualization principles and a comparison with similar work.
- **Usability evaluation** is aimed at observing how users interact with a tool, either through a set of predefined usability tasks or through observations of free-form usage.
- **Algorithmic evaluation** measures computational performance of a technique.
- **Heuristics evaluations** entail inspecting a system according to a chosen set of relevant heuristics or guidelines.

Our analysis found that the majority of technique papers in the surveyed corpus used use cases (63%), followed by controlled experiments (28%), case studies (15%), and usability studies (15%). Less frequently, we found examples of theoretical evaluation (8%) or algorithmic evaluation (4%). We did not encounter heuristic evaluation in our corpus. Also, likely due to our inclusive definition of evaluation, we found that all papers included at least one of these methods.

Use cases were by far the most common approach to evaluation, likely due to the simplicity of conducting them and their usefulness for explaining a new technique. When done well, use cases leveraged real world data to walk the reader through examples of how a tool can be helpful for data analysis and exploration. A common problem with

this approach is the choice of data. Sample datasets such as movie actors and movies, or interactions between the characters in *Les Miserables*, are readily available and explainable without extensive domain knowledge, but they do not provide evidence that the technique would scale well to real world applications and datasets. With more complex, real world datasets, use cases provide stronger evidence that the proposed technique could be effective in the wild.

Controlled experiments were also common and were most often conducted in one of two scenarios: 1) comparing a novel MVN approach with existing techniques [28], [89]–[91]; or 2) comparing variations of a given design for visualizing MVNs [92], [93].

Other common types of evaluation encountered were case studies and usability evaluations. The systems evaluated with case studies were often developed in close collaboration with domain experts. These systems are frequently complex and unique so that they cannot be easily compared to other tools.

Usability studies were often used in conjunction with another evaluation approach, such as a usage scenario or a case study. Theoretical evaluations were uncommon, likely due to the challenge of convincing readers of the superiority of the proposed tool through theoretical arguments. Only 4% of the papers were coded as algorithmic evaluations, which is unsurprisingly low given the types of contributions—novel encodings as opposed to novel algorithms—of papers in this survey.

2.5 Conclusion

This dissertation is grounded in the rich history of multivariate network visualization and its applications in real world scenarios. It is also based on the meta-analysis of real world applications of MVN visualizations, and the evaluation methods used for them in the last two decades.

CHAPTER 3

MULTIVARIATE NETWORK ANALYSIS TASKS

An MVN analysis task can be defined as an activity that users wish to perform by interacting with a visual representation of a multivariate network. Defining a set of tasks that properly capture these user actions can be useful for evaluating and comparing MVN visualization systems. We developed a MVN task taxonomy that is designed to evaluate MVN visualization techniques for their suitability to address a task and is complementary to the general network task taxonomies outlined in Section 3.1. This task analysis, combined with considerations of network type and network scale described in Chapter 2, form the basis of our assessment of individual visualization techniques and of our guidelines for MVN visualization as presented in Chapter 4.

Aside from our proposed taxonomy, another relevant task taxonomy in the context of this dissertation is that by Lee et al. [94], who distinguish between *overview tasks* and *focus tasks*, a classification we use to describe the tasks that Juniper supports (Chapter 6). *Overview tasks* are those for which analysts need to see a large set of nodes and edges, such as when estimating the size of a network, identifying clusters, or finding articulation points. An example of an overview task with multivariate networks is to explore how migration patterns within the US differ by age. When visualizing overviews of all but trivial networks, the large number of nodes and edges makes it impossible to show labels and attributes for individual nodes. For **focus tasks**, the details of a small, well-defined subset of nodes are relevant and necessary for the task. These details include topological information such as neighborhoods of or paths between nodes, as well as attributes, including node labels and other associated data.

Of the nine tasks Lee et al. identify, five are focus tasks (adjacency, accessibility, common connection, follow path, revisit). The attribute-related tasks — node attributes, link attributes — are described mostly in a focus context by Lee et al., but they can also be

useful in a global context (for example, estimating the average age of members of a social network). One task — connectivity — can be broken up into overview and focus tasks. For example, finding the shortest path between nodes is a focus connectivity task, but identifying clusters, connected components, bridges, or articulation points is an overview variant of the connectivity task.

With regard to topology-attribute interaction, focus tasks can be classified into two groups: 1) those that can be achieved in the context of neighborhoods (adjacency, accessibility, common connection), e.g., to see whether friends have similar educational attainment or whether health issues, such as obesity, spread in a neighborhood of friends [95]; and 2) those related to exploring attributes in the context of paths (follow path, connectivity), for example, to judge delays over time in a computer network or whether a path in a biological pathway is active in a set of samples [58].

3.1 Related Work — Network Task Taxonomies

There are various data-type independent visualization task taxonomies. The works by Brehmer and Munzner [96] and Schulz et al. [97] are examples of recent, generic visualization task taxonomies. An example of a data-type-specific task taxonomy is the work by Valiati et al. [98] that covers multidimensional visualization techniques.

Specialized task taxonomies for networks also exist, such as the well-known taxonomy of network tasks by Lee et al. [94]. In addition to topology-focused tasks, their taxonomy includes selected attribute-based tasks. For node attributes, these tasks are *finding nodes having a specific attribute value* and *reviewing the set of nodes*. For edges, the two tasks are about identifying neighbors of a node connected by an edge with a specific property (edge type, specific edge value). Shneiderman and Aris [99] also discuss examples of MVN tasks but do not take a systematic approach to the topic.

The most comprehensive analysis of tasks for multivariate networks was done by Pretorius et al. [100]. The authors generalize the tasks proposed by Lee et al. to cater to cases for which the network's attributes are of interest to the user. This approach is useful in that it provides a systematic and comprehensive approach to MVN tasks. They list 25 tasks in the categories of structure-based (topology-based), attribute-based, browsing, and estimation tasks. They also discuss how these tasks are composed of lower level analytical

tasks proposed by Valiati et al. (identify, determine, relocate, compare) [98].

Pretorius et al. discuss that in the abstract, a task is an analytical activity performed on an entity and a property of an entity, which are called "actions" applied to "targets" by Brehmer and Munzner [96]. For the purpose of this dissertation, we believe that a slightly simplified approach is useful for both characterizing existing techniques and recommending techniques based on tasks specified by analysts.

3.2 Multivariate Network Visualization Task Taxonomy

In this section we describe the task taxonomy derived from analyzing the corpus of MVN visualization techniques surveyed for our state-of-the-art report on MVN visualizations [3]. We describe the taxonomy and then give a summary of how these tasks were observed in real world MVN visualization applications.

In our taxonomy, any task can be expressed as a combination of two fundamental tasks, as applied to different topological structures of a network. These two basic tasks are:

- *analyzing topology for given attributes (TgA)*, which are aimed at identifying, characterizing, or comparing topological structures that have certain attributes, and
- *analyzing attributes for a given topological structure (AgT)*, which are aimed at identifying, characterizing, or comparing the attributes of a given topological structure.

We do not further distinguish between more detailed tasks, such as identifying, summarizing, or searching, because we believe that in most practical scenarios these go hand in hand, and because most visualization techniques that support one of these also support the others. A notable exception is comparison, which can often be achieved without special considerations, but can benefit from dedicated support by a visualization technique in certain situations.

The targets in the context of network tasks are the *topological structures*. We consider major types of topological structures, shown in Figure 3.1, but recognize that others, such as cliques or spanning trees, could be relevant for specific analysis questions. The structures include individual *nodes and edges*, *paths* of sequential nodes and edges, *neighborhoods* of a particular node or edge, *clusters* of nodes and edges, and larger *networks or subnetworks* that can be considered as complex networks by themselves. We use the term "cluster" to refer to a set of nodes that are well connected to each other, but less so with other nodes

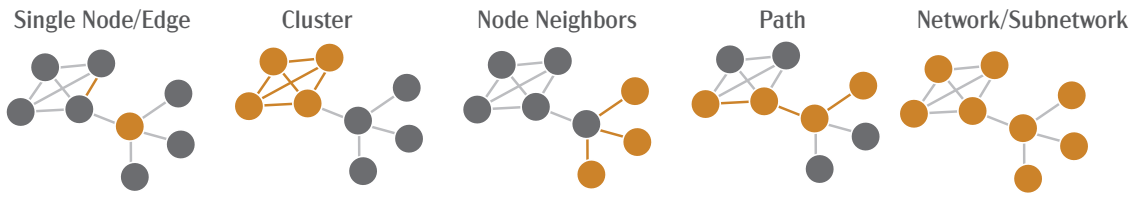


Figure 3.1. Topological structures that are the targets of MVN analysis tasks.

with regard to the topology of the network. A cluster is also often called a community, especially in the context of social networks. Conversely, a "subnetwork" is a subset of a network that is not limited to a specific structure.

We selected these specific structures based on our literature analysis of MVN visualization techniques, where we identified many published techniques that are optimized for performing MVN tasks on specific topological structures. For example, approaches such as Pathfinder [62] and Entourage [57] are explicitly developed for path-based MVN tasks. GraphCharter [101], on the other hand, is optimized for analysis on node neighbors.

A comprehensive analysis of multivariate network tasks can thus be described as a combination of general network tasks (e.g., as described by Lee et al. [94]) and our MVN tasks. By reducing MVN tasks to a combination of two basic tasks and a topological structure, we are able to decompose complex tasks into a simple combination of tasks and targets. The task "Estimate the average value of a specific attribute of all nodes connected to a node" (a concrete example: what's the average income of University of Utah graduates in a social network) can be deconstructed into three basic tasks: 1) find the node with label "University of Utah" (analyzing topology for given attributes, on target "network"); followed by 2) the adjacency task of Lee et al: find the set of nodes adjacent to a node; followed by 3) estimate the average value of an attribute of a node neighborhood (analyzing attributes for a given topolgy on target "neighbors").

Table 3.1 gives examples of the two basic MVN analysis tasks as applied to the set of topological structures listed in Figure 3.1. For each task, we also provide an example for a domain-specific task from the literature.

Table 3.1. List of generic and domain-specific examples of MVN analysis tasks composed of our two basic tasks and the described topological structures. Each domain-specific example references a paper that describes that task.

Toplogical Structure	Task Type	Example	Domain-Specific Example
Node/Edge	TgA	Which node/edge has the given attribute value?	Who is the oldest person in a social network? [13]
	AgT	What is the attribute value of a given node/edge?	What house is Jon Snow in? [2]
Neighbors	TgA	Which neighbors have a given attribute value?	Who is Mary's tallest friend? [89]
	AgT	What are the attribute values of a node's neighbors?	What are the zodiacs of Joe's friends? [101]
Paths	TgA	Which path between two nodes has the lowest cost?	Which path between two genes is most frequently cited in pathway databases? [62]
	AgT	What are the attribute values along a path?	How does a metabolite increase along a pathway? [61]
Clusters	TgA	Which cluster has certain attribute characteristics?	Which cluster has the most educated people? [50]
	AgT	What are the attribute characteristics within a cluster?	What is the average education level in Mary's cluster? [50]
Network	TgA	Find the subnetwork of nodes with a specific attribute.	Which subnetwork contains only people with a PhD? [50]
	AgT	What are the attribute values in this network?	What is the average age in this network? [51]

3.3 MVN Tasks in Real World Applications

When surveying the corpus of papers, we observed both TgA and AgT tasks, but noticed a preference for one of the task types in certain application areas. Techniques developed for social network analysis, for example, tend to first focus on topology-only tasks, followed by attribute tasks on the selected topological structures (AgT). For example, a commonly described task is to identify particular structures, such as communities, followed by an analysis of the attributes in these structures [25].

We also observed that identifying a topological structure for given attributes (TgA) is frequently supported by query or filter operations, instead of being a purely visual task. For example, an analyst may want to identify a fraudulent bank account by querying for patterns of transactions, and then expand a network of interactions between that account and others [102]. Similarly, Juniper [2], Pathfinder [62], and Enroute [58] use queries or

filters to isolate a subnetwork or path of interest, followed by inspection of the attributes in that structure.

3.4 Conclusion

In this chapter we presented a taxonomy of MVN analysis tasks, which we designed to provide a framework with which to evaluate MVN visualization techniques. We leverage this taxonomy when describing MVN visualization techniques and their strengths and weaknesses (Chapter 4), as well as when designing tasks for the user study comparing node-link diagrams and adjacency matrices (Chapter 7). We also provide an overview of related network analysis task taxonomies, with a focus on the taxonomy of Lee et al. [94] since we use this distinction between *overview tasks* and *focus tasks*, to describe the tasks that Juniper supports (Chapter 6).

CHAPTER 4

MULTIVARIATE NETWORK VISUALIZATION TECHNIQUES

Networks are ubiquitous in today's society. Visualizing MVNs is particularly challenging, as both the *topology* of the network (also called *structure* of the network), and *attributes* associated with the nodes and links need to be shown. When visualizing topology and attributes in the same view, choosing efficient encodings for one aspect often interferes with the ability to effectively visualize the other. For example, when attempting to encode multiple node attributes directly on the node in a node-link diagram, a visualization designer has to make compromises with regard to the node size, the network size, and the number of attributes shown. Conversely, when visualizing attributes and topology in separate views, an analyst's ability to judge interactions between nodes/links and attributes is limited. Most of the published techniques on multivariate networks attempt to mitigate the trade-off between visualizing topology and visualizing attributes well by introducing novel encodings for certain data types or analysis tasks.

In this chapter, we introduce a new typology for multivariate network techniques. This visualization technique typology consists of four dimensions: layouts, view operations, layout operations, and data operations. The **layouts** category includes the eight unique multivariate network layouts. **View operations** describe how multiple views can be used for multivariate network visualization. **Layout operations** either combine multiple layouts or multiply layouts based on attributes. **Data operations** describe possible network wrangling operations such as aggregation and computing new attributes. We also provide explicit guidelines on recommended usage of each technique, as well as on how to select an appropriate technique given a network type, attribute characteristics, and a set of tasks the user aims to support. A companion website, available at `https://vdl.sci.utah.edu/mvnv/`, provides an overview of the taxonomy and a wizard to select

a technique for a specific data and task combination. This work is the result of a survey of MVN visualization techniques, covering over 200 papers published between 1991 and 2018.

4.1 Related Work — Network Visualization Surveys

The body of literature in the space of visualizing networks is consistently growing. There are several STARs on network-related topics such as visualizing large networks [5], trees [42], implicit hierarchies [103], group structures in networks [104], dynamic networks [105], and edge bundling techniques [106], among others.

The survey on multifaceted network visualization by Hadlak et al. [107] covers the visualization of four facets of network visualization, two of which are relevant to our work — partitions and attributes. Their work distinguishes three overarching categories of MVN visualizations — visualizations with a topology-driven layout, an attribute-driven layout, and coordinated views — but is not exhaustive in describing MVN techniques, including tabular techniques and hybrid approaches. Most related to the work reported on in this chapter is a survey book edited by Kerren et al. based on a Dagstuhl Seminar [47] that reports on selected topics in the field but does not introduce a taxonomy of existing techniques. This chapter fills that gap by describing the design space of techniques used to encode multivariate networks and classifying these techniques into a typology.

4.2 Methodology

This chapter is focused on techniques for visualizing multivariate networks. We analyzed these techniques through the study of a corpus of literature composed of two types of papers: those that propose a specific encoding to enable visualization of multivariate networks, which we refer to as *technique* papers; and those that focus on evaluation and application of these techniques.

We developed a typology of MVN visualization techniques through our analysis of the corpus, following the analysis method used by Vehlow et al. [104]. After outlining the scope of techniques we consider, our core analysis activities were to compile a corpus of relevant papers and code each paper according to its described technique, application area, and evaluation method. This section describes our method in detail and provides a list of

the categories used to derive the proposed taxonomy.

4.2.1 Scope

We limit ourselves to techniques that specifically aim to visualize multivariate networks. Techniques that focus on visualizing either only the network topology or attributes are not within the scope for this survey. We consider techniques for all network types, including general (complex) networks, layered networks, k/bipartite networks, and trees.

4.2.2 Corpus

We identified candidate papers for the corpus using three sources: 1) all papers published in the VIS and EuroVis conference proceedings since their respective inceptions; 2) papers in the IEEE Digital Library identified with a search for the keywords *network, graph, visualization, multivariate, attributes, faceted, and multidimensional*; and 3) papers that cite, or are cited by, the papers identified in steps 1 and 2, according to Google Scholar. For each of these candidate papers, we manually surveyed its title, keywords, and abstract, and included the paper in the corpus if it proposed a technique, evaluation, or application for visualizing multivariate networks. This process resulted in a corpus of 210 papers published between 1991 and 2018.

4.2.3 Coding

We completed two rounds of coding on the corpus using the MAXQDA Qualitative Data Analysis Software [108]. In the first round, we coded papers for their application area and evaluation method, and in the second round we focused on coding the visualization techniques. The application areas were coded using an open-coding approach. The evaluation methods, however, were coded with a closed set derived from the seven guiding scenarios identified by Lam et al. [109]. This first coding pass allowed us to review the entire body of work and prepared us for the second round of exclusively coding the visualization technique used in each paper.

The aim of the second round of coding was to identify the specific visual encodings employed by each multivariate network technique. We used the open-coding method described by Beck et al. [105], and coded papers for the specific visual encodings used to visualize network structure and associated attributes.

In the second round, we did not code the entire corpus, instead stopping when we reached saturation [110], i.e., we no longer encountered new concepts in any of the categories we were coding for. More specifically, the categories we coded for were visualization techniques and encoding methods. We refer to *techniques* as a visualization idiom such as node-link diagram, adjacency matrix, tree-like layout, and others. Nuances of the technique used were also captured, including encoding methods such as the use of color, a particular network layout algorithm, and ordering for matrices.

In addition to textual codes, we also developed a set of *visual codes* that summarized the encountered approaches to encoding both the topology and the attributes of a network (Figure 4.1). These representations did not aim to faithfully reproduce every nuance of the techniques, but instead supplemented the assigned textual codes with a visual summary of each technique. The need for visual codes arose from the limited ability of textual codes to efficiently capture the ways in which different techniques encoded the topology and the attributes of a network. As a result, the visual codes emerged as a way to capture these aspects and were critical in our analysis and in the development of the multivariate network typology.

We categorized the textual and visual codes into 33 distinct techniques for visualizing multivariate networks, grouped by their underlying layout approach: node-link, tabular, implicit, or coordinated views. We then performed an analysis of these techniques, aided by feedback from a second senior author. This analysis revealed core common encodings among a set of techniques. For example, although PivotGraph [111], GraphDice [54], and FlowVizMenu [112] appear to have unique approaches, they build on a common backbone of positioning the nodes according to attribute values. Similarly, Circos plots [113] and force-directed layouts share the basics of node-link layouts with edges connecting items.

This analysis process provided us with two insights regarding the similarities and differences of the techniques. First, we found a useful distinction between the layout technique used by an approach, and the data transformations applied to the network prior to visualization. The second insight is that complex layouts are often the result of combining two or more core layouts into a single design. Guided by these insights, we used the technique analysis to develop a typology with four dimensions: layouts, view operations, layout operations, and data operations. The layouts category includes the eight

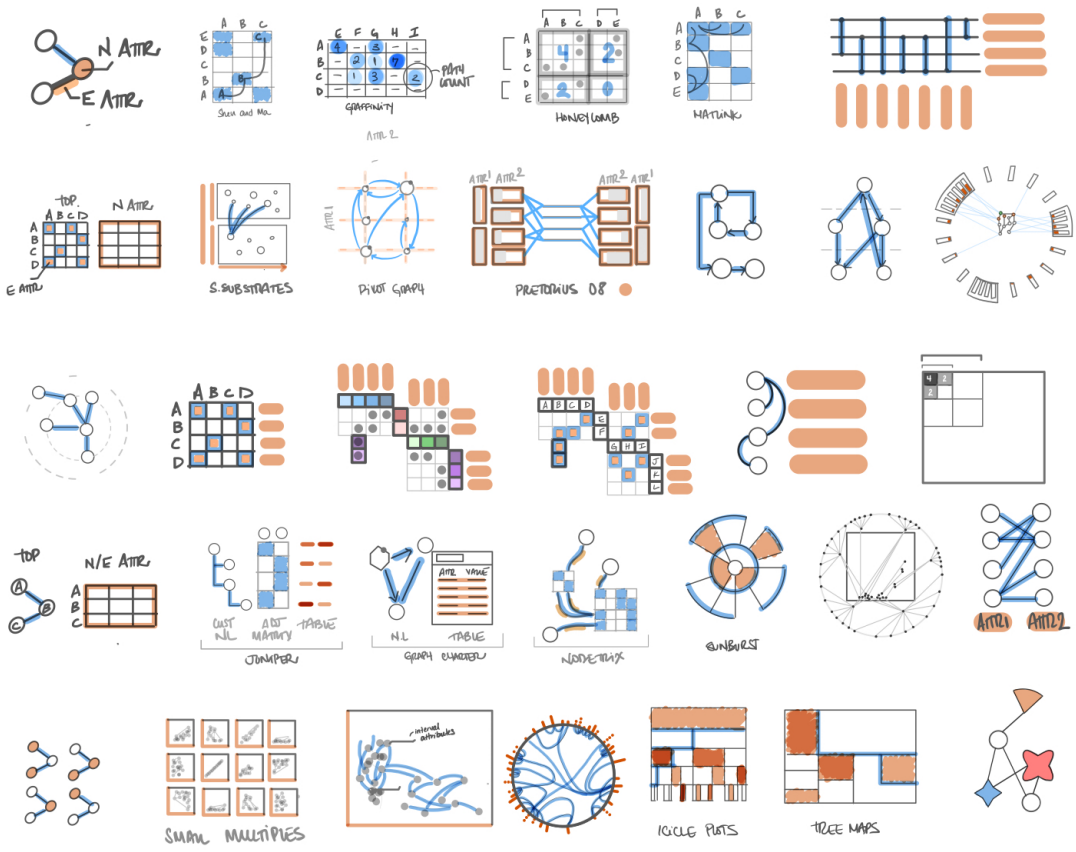


Figure 4.1. Visual technique codes, used to capture different visual approaches to encoding topology (blue) and attributes (orange).

core layout approaches observed in the literature (Figure 4.2). View operations are those that combine the basic layouts into multiple coordinated views. Layout operations refer to the process of either multiplying a given layout or combining components of two or more layouts into a hybrid layout, such as the NodeTrix design [24]. The last dimension, data operations, encompasses any data processing that is done to the data prior to visualization, such as aggregation and computing new attributes. All four dimensions are discussed in more detail in Section 4.3.

4.2.4 Developing the Numerical Guidelines

The process of scoring the techniques (described in Section 4.4) for how well they support certain data types and tasks was done with a 'peer review'-like system among all authors of the paper. The first phase involved independent scoring by each author for

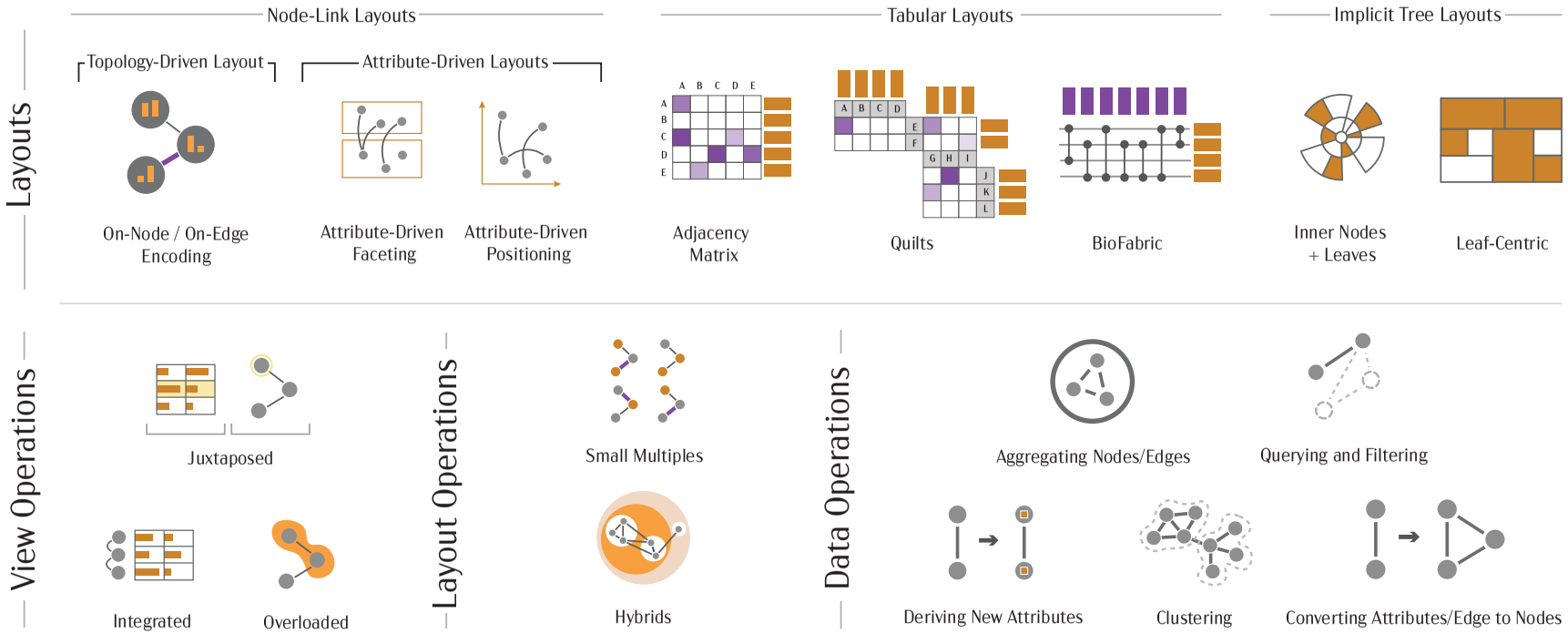


Figure 4.2. A typology of operations and layouts used in multivariate network visualization. **Layouts** describe the fundamental choices for encoding multivariate networks. **View Operations** capture how topology and attribute focused visualizations can be combined. **Layout Operations** are applied to basic layouts to create specific visualization techniques. **Data Operations** are used to transform a network or derive attributes before visualizations. The colors reflect node attributes (orange), edge attributes (purple), and topology (grey).

each of the techniques along the 20 dimensions. Once all authors submitted their scores, each author identified his or her outlier scores and wrote a short justification for that score or modified the score when a mistake was made. A discussion phase allowed us to reach a consensus score for every combination in the table.

The rating of techniques that fall within the view operations, such as overlays and juxtaposed views, required some assumptions in order to grade them consistently across authors. For example, when rating juxtaposed views, we considered an optimized encoding for the topology and attributes in order to focus the score on the juxtaposed aspect of this technique, and not on the choice of layout for the topology or the attributes. The same applied to the overlay category, where the score was given to account for the overlaid aspect of the views, regardless of the specific layout technique used for each component.

4.3 MVN Visualization Typology

The MVN visualization typology we contribute in this chapter is made up of four components: a taxonomy of MVN layouts, and three types of operations that can be applied to these layouts (Figure 4.2). Every technique and tool discussed in this chapter can be described by selecting (at least) one layout and optionally applying operations to it.

The *layout taxonomy* encompasses core approaches to encoding the topology and attributes of multivariate networks and trees. The taxonomy of layouts contains three classes: node-link, tabular, and implicit layouts, with various layouts within these classes.

The *view operations* capture different ways of combining multiple views with each other. For example, view operations include juxtaposition for placing an attribute and a topology visualization side-by-side.

The *layout operations* allow for different combinations of the layouts to produce hybrids and small multiples.

The *data operations* category contains methods that can be applied to the data either as a preprocessing step or during the analysis. Example data operations are aggregating nodes into supernodes or filtering parts of a network.

In describing MVN visualization techniques along these four orthogonal dimensions, we not only cover the techniques observed in the set of papers surveyed, but also provide a design space in which to decompose and analyze any MVN visualization technique.

When describing and discussing these approaches, e.g., with respect to scalability, we assume a static visualization technique. However, we call out opportunities for *interaction* where appropriate. Certain types of interactions, such as zooming and panning, or visualizing attributes such as labels in tool tips, can be applied to most techniques. Other, more sophisticated interactions, such as linking and brushing, sorting, changing the layout algorithms, or manually adjusting a layout, depend on the visualization technique used. The *data operations* we describe can frequently be invoked interactively, and especially operations such as filtering or aggregating can drastically change how much and what types of raw data can be visualized with a basic layout technique.

4.3.1 Layouts

We distinguish between three approaches to network visualization: node-link layouts, tabular layouts, and implicit tree layouts. The class of tabular layouts contains the widely used matrix layout and two variants: quilts and biofabric. Implicit tree layouts encode the relationships between nodes only by their relative positions; hence edges are only implicitly represented. In practice, the lack of explicit edges means that implicit layouts are useful only for trees.

4.3.1.1 Node-Link Layouts

In a node-link diagram, nodes are drawn as vertices, and edges are lines that link connected entities in the network. Schulz and Schumann [114] distinguish between *free layouts*, where the node layout is not restricted, an example of which is the force-directed placement; *styled layouts*, where the node positions are determined by a predefined scheme, such as a grid, or a radial layout; and *fixed layouts*, where the position of the node is determined by an attribute, such as latitude and longitude on a map. In fixed layouts, the only degree of freedom is conferred to the drawing of the edges connecting the fixed nodes.

We use this distinction, but for the purposes of multivariate network visualization, we group free and styled layouts into one larger category of *topology-driven layouts* and take a slightly relaxed view on fixed layouts and describe it as *attribute-driven layouts*.

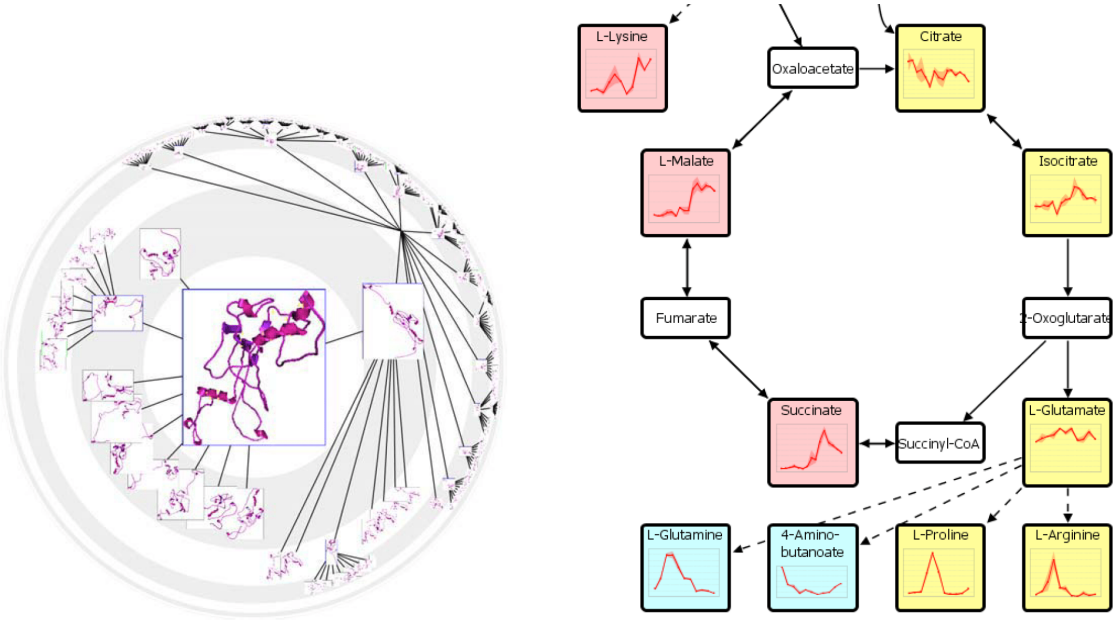
Topology-driven layouts prioritize the topology of the network over the attributes of the nodes and edges. The most common node-link layouts, such as force-directed layouts,

spectral layouts, or orthogonal layouts, fall into this class, or more specifically into the free layouts as described by Schulz and Schumann. Trees are also commonly drawn as topology-driven node-link layouts. Styled layouts position nodes based on a pattern, such as along a circle or in a grid, and then connect them with edges. Styled layouts quickly lead to clutter, which can be partially mitigated by optimizing the ordering of nodes, e.g., by placing connected nodes close to each other. As topology-driven layouts are optimized to visualize the network structure, position, the most powerful visual channel, is not available to encode attributes of the network. Hence, the only layout strategy to visualize attributes is to alter the appearance of the nodes and links through on-node or on-edge encoding.

Attribute-driven layouts use node position to encode attributes. We distinguish two cases: attribute-driven faceting, which places the nodes in regions corresponding to a categorical attribute, but the exact node position within that region is determined in another way, or attribute-driven positioning, which places the nodes exactly by an (often numerical) attribute value.

4.3.1.1.1 On-node/on-edge encoding. On-node and on-edge encoding refers to modifying the visual appearance (size, color) of a node or an edge or embedding marks (bar charts, line charts, etc.) in a node or an edge in a node-link diagram (Figure 4.3).

For *nodes*, labels are often shown as text, whereas color coding is a common choice to encode numerical or categorical data values. Other common encodings for node types are shapes or icons. Gehlenborg et al. [14] review techniques used in systems biology for visualizing multivariate networks, many of which make use of on-node encoding by means of embedded charts, such as line charts, box plots, etc. An example of a systems biology use of on-node encoding is shown in Figure 2.2(a), where metabolic pathways are overlaid with node attributes on metabolite concentration [12]. Auber et al. [115] use on-node encoding on social networks, displaying small network representations for previously computed topological features of interest. Also in the realm of social media, Vizster [13] uses photo and labels on the nodes. Dunne and Shneiderman [116] use glyphs to encode aggregate nodes representing topological patterns in the network such as cliques and fans. On-node encoding is also widely supported by common network visualization tools such as Cytoscape [15] and Gephi [16].



(a) (b)

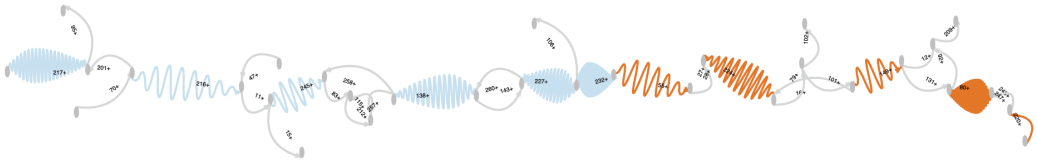
Figure 4.3. Complex on-node encoding. (a) A protein interaction network where nodes contain visualizations of protein configurations [18]. (b) Line charts show metabolite concentrations over time in a pathway network [119].

Ghani and Elmqvist [117] study the efficiency of different channels for on-node encoding for revisitation tasks. The authors compared the use of color, size, and color combined with size, but did not find significant difference in the performance of their tasks.

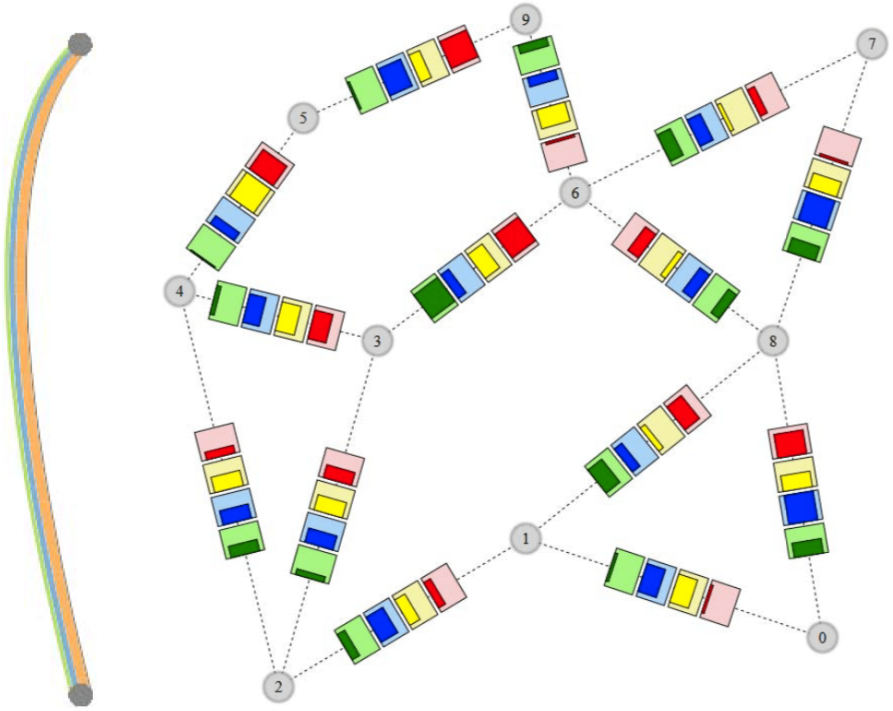
On-node encoding can also be realized using more complex visualizations within the nodes. Examples of these encodings include line-charts, or a "business card" layout for a person, with name, picture, age, etc.

Examples for techniques using complex on-node encodings are MoireGraphs [18] (see Figure 4.3(a)) and Network Lens [67]. Network Lens allows the users to enlarge nodes and encodes their attributes with approaches such as bar charts and parallel coordinates plots. Other examples from the biology domain include embedded bar charts with error bars [118] and line charts [119].

On-edge encoding shows edge attributes by modifying the channel of the line mark used for the edge. Common approaches are modifying line width [120],[121], line color [121], stippling and dashes [119], curvature [122], or blurring [123]. Multiple attributes are occasionally encoded with bar charts overlaid on the edges [92] (Figure 4.4(c)) or with multi-



(a)



(b) (c)

Figure 4.4. On-edge encoding. (a) A numerical attribute is encoded by the overall length of the edge curve. To make the distance between connected nodes of approximately uniform length, wiggles are used [124]. (b) Several numerical attributes are encoded by the thickness of the colored segments [125]. (c) Colored bars encoding four quantitative edge attributes [92].

colored segments with different line width [125] (Figure 4.4(b)).

Abyss Explorer [124], shown in Figure 4.4(a), encodes edge attributes by edge length. However, as modifying edge length has a negative impact on the freedom to place nodes, the authors "wiggle" the edges, so that the attribute is also encoded by the frequency of the wiggles.

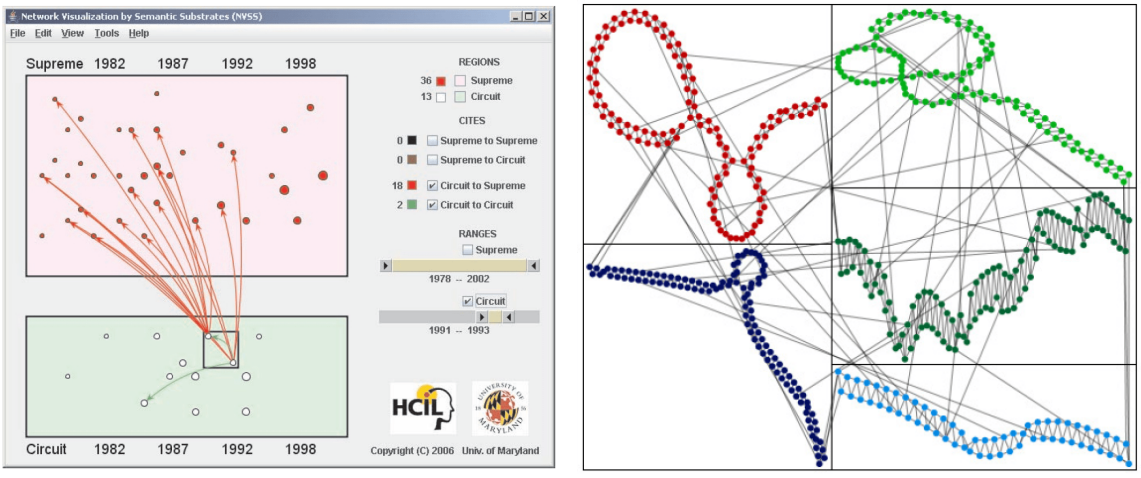
Edge directionality is commonly visualized with arrows at the ends but tapering or gradients are other approaches [126].

We recommend the following usage for this technique. On-node/on-edge encoding supports the integration of topology- and attribute-based tasks well and supports all kinds of MVN tasks on all structures (see Section 4.4). On-node/on-edge encoding is easily understood by most users, and works well for sparse complex networks, layered networks, and trees. However, it comes with scalability trade-offs. Even for a modest number of nodes in a node-link layout, node size has to be limited; hence little space is available to encode attributes. When details about nodes are shown, as for example in MoireGraphs [18], the number of nodes that can be displayed simultaneously is limited. We recommend on-node layouts when only a few (usually under five) attributes on the nodes are shown, or in combination with a zooming/filtering strategy. Complex on-node encodings can also be embedded on top of aggregated nodes, to summarize the content of the aggregates, in which case their larger size is often justified [17]. On-node encoding generally works well for networks with different node types. On-edge encoding is even more limited than on-node encoding. First, most node-link layouts guarantee that nodes do not overlap; however, edges commonly cross even in sparse networks, interfering with on-edge encoding. Second, the nature of the link mark as a slim line limits the discriminability of any modulation of the visual channel. We recommend on-edge encoding for a single numerical or categorical attribute.

4.3.1.1.2 Attribute-driven faceting. Attribute-driven faceting groups nodes according to one or more attributes and places the elements of a group in a shared region. A prominent example is Semantic Substrates [99], Figure 4.5(a), which facet a network based on a categorical attribute and let analysts choose whether to show links between or within facets, or both. This approach is also commonly used in biological networks to visualize nodes within spatially segregated cell compartments [63], or to lay out k-partite or mul-

tityped networks [45], [55], [128]–[130]. There are alternatives to faceting by sets or node types: Figure 4.5(b) shows faceting based on a hierarchical clustering algorithm in which the clusters are laid out in a treemap [127] and nodes are shown within the treemap cells. A key choice after faceting is how to place the nodes within the assigned region, which can be done using any network layout technique. Semantic Substrates [99], [131], for example, place nodes according to other attribute values, whereas Cerebral [63] uses a layout optimized for topology. A series of other techniques uses a linear layout [45], [128], [129], which is amenable to attribute visualization.

We recommend the following usage for this technique. Attribute-driven faceting is well suited for networks with different node types or with an important categorical or set-like attribute. Such faceting is especially useful when the separation into groups and the study of the interaction within and between the groups are the subject of the analysis, which is commonly the case in k-partite and layered networks (see Section 4.4). Due to restrictions on the layout, it is slightly less scalable with respect to the number of nodes and network density than node-link layouts. Other attributes can be visualized independently of the basic principle of faceting, so that the scalability with respect to other attributes depends on these choices. Edge attributes are not supported by faceting and have to rely



(a) (b)

Figure 4.5. Attribute faceting. (a) Nodes are faceted by node type or a set attribute and placed in corresponding regions. The horizontal position within the facets is driven by an attribute [99]. (b) Nodes are faceted based on a clustering algorithm. The clusters are arranged using a treemap layout [127].

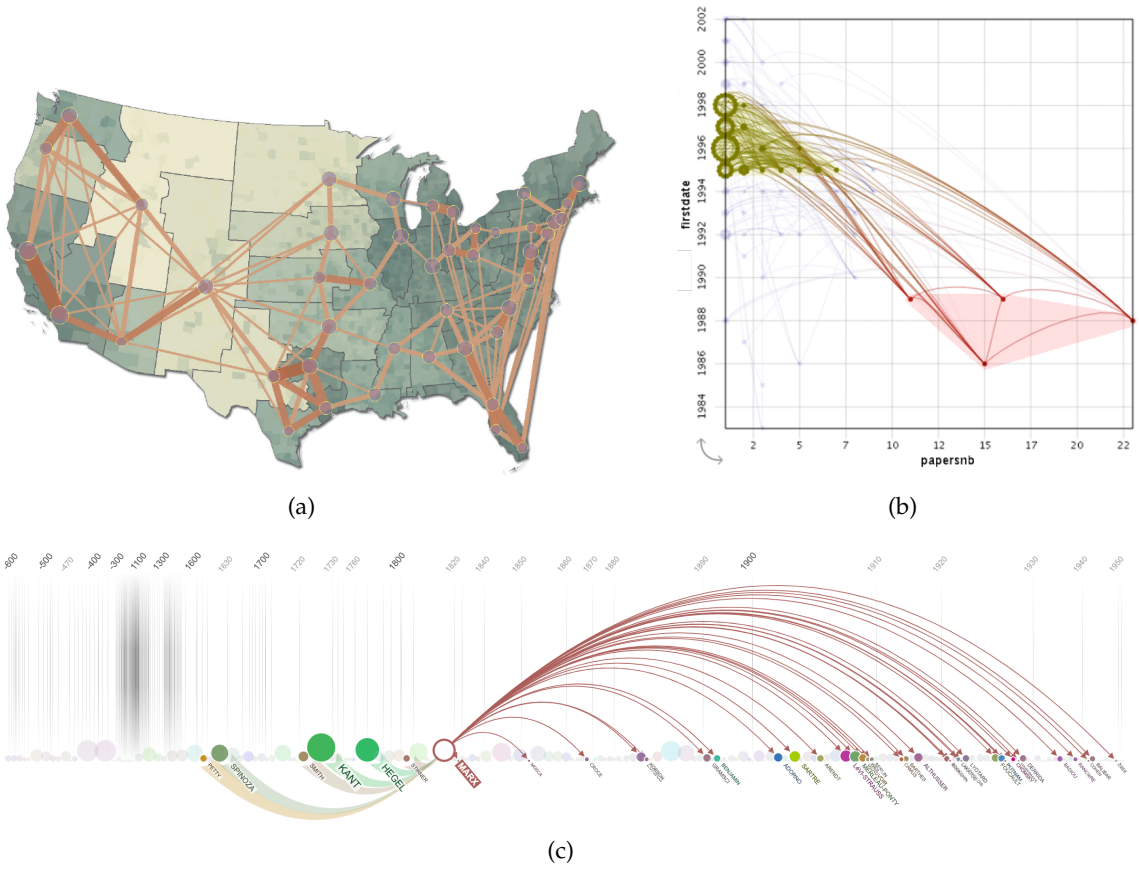
on a secondary encoding. Neighborhoods, paths, and clusters are not easily visible if they span different facets. We recommend attribute faceting for cases in which nodes can be separated into groups easily and in which these groups are central to the analysis.

4.3.1.1.3 Attribute-driven positioning. Attribute-driven positioning (fixed layouts) assigns node or edge positions according to one or more attribute values. Spatial networks, such as networks overlaid on maps [17], [121] (Figure 4.6(a)), are the most common example of attribute-based node positioning. However, attributes other than geographic coordinates have also been used. In 1 dimension (1D), genomic coordinates are a common example [113], [132], as are networks with a time component [133], shown in Figure 4.6(c). Multiple nonspatial attributes can be encoded in two dimensions [54], [99], [112] (Figure 4.6(b)). This principle has also been extended to visualize multiple attribute pairs [54], [112] in a small multiple/scatterplot matrix display. The attributes used for positioning can also be derived from a larger attribute vector. For example, Bonabeau [134] uses self-organizing maps, and Doerk et al. [133] use dimensionality reduction based on attributes to compute placement of the nodes in space.

Spatial positioning is also sometimes combined with an aggregation operation (see Section 4.3.4). For example, Wattenberg [111] leverages attribute-based positioning to generate an overview of an aggregated network along two user-defined attributes.

Using spatial positioning for edges is less straightforward, as the edges need to connect nodes at different positions. One approach is edge bundling, which routes edges that are related according to some metric of similarity in proximity [135], [136] (see the recent survey by Llhuillier et al. [106] for details on edge bundling). The similarity metric is frequently based on the topology of the network [137], so that edges that have a similar region of origin and destination are bundled together. Edge bundling reduces clutter and makes it easier to detect connectivity patterns in the network [77]. A similarity metric can also consider attributes of the edges [77]. Similar edges are then bundled only if they share directionality or specific attributes. Edge bundling methods do not alter the endpoints of an edge; they only alter the routing of the edge between source and target nodes [106].

Figure 4.7 illustrates the usage of edge bundling in a dataset of flight paths (edges) between hubs (nodes) in France [77]. The routing of similar edges based on their source and target, as well as their direction attribute, simplifies the interpretation of flight paths



(a) (b) (c)

Figure 4.6. Attribute-driven node positioning. (a) Nodes are positioned according to their latitude and longitude on a map [121]. (b) Nodes are positioned according to nonspatial attributes [54]. (c) Nodes are positioned in 1D, time in this case [133].

across the country.

We recommend the following usage for this technique. Attribute-driven positioning is well suited for cases in which the value of a single node attribute or the relationships between two node attributes are the most important feature in a network dataset, but it does not lend itself well to visualizing the topology of the network. Even simple structures such as neighborhoods can be difficult to spot. Complex structures such as paths or clusters can be hidden completely (see Section 4.4). Unlike attribute-driven faceting, the technique is well suited for quantitative attributes. The technique works mostly for homogeneous networks since it relies on common node attributes for positioning. Due to the placement driven by attributes, nodes can occlude each other (although jitter was suggested to address that [54]), and edge crossings are much more likely than, e.g., in

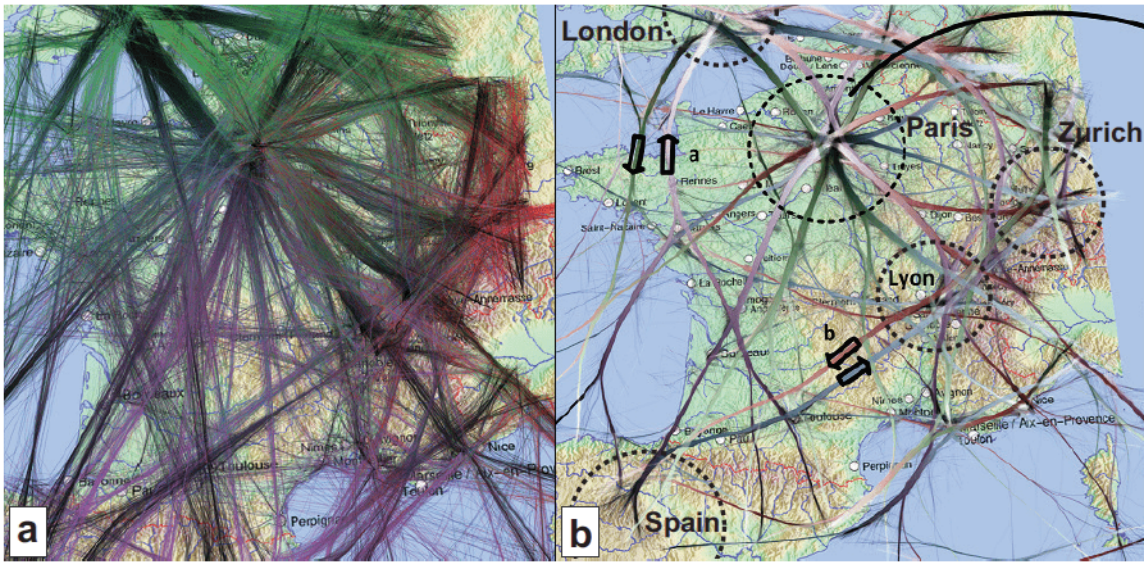


Figure 4.7. Attribute-driven edge positioning via edge bundling [77]. (a) Raw edges, colored according to directionality. (b) Edges bundled while taking direction into account.

a force-directed layout. Hence, attribute-driven positioning is not well suited for dense networks or for visualizing edge attributes. We recommend attribute-driven positioning for smaller, sparse networks in which relationships between node attributes are paramount to the analysis task, and topological features provide only context.

4.3.1.2 Tabular Layouts

Tabular layouts encompass approaches in which nodes and/or links are represented as columns and/or rows of a table. The most well known of these approaches is the adjacency matrix, but this category also includes Quilts [138] and BioFabric [139]. A beneficial property of all tabular layouts with respect to multivariate networks analysis is that nodes, and in some cases edges, are in dedicated rows or columns, which can also be used to visualize attributes.

4.3.1.2.4 Adjacency matrices. Adjacency matrices encode nodes as rows and columns, whereas the presence/absence of an edge between two nodes is encoded in the cell where the nodes rows and columns intersect. Matrices work for both directed networks, in which case the rows are the source of the edge and the columns the targets (or vice versa), and undirected networks, in which case the encoding is redundant above and below the

diagonal of the matrix. Matrices have both favorable and unfavorable properties compared to node-link layouts when analyzing topological features [23], [29], [140]: matrices are widely acknowledged to be well suited to analyze neighborhoods and clusters (given a suitable matrix seriation) but perform poorly when analyzing paths. For attribute visualization, adjacency matrices have favorable properties for both node and edge attribute visualization.

For *node attributes*, it is easy to juxtapose attribute visualizations with the rows or columns of the matrix [27], [28], as shown in Figure 2.4(a). Attribute visualizations can be aligned in a tabular layout so that comparisons between nodes on the same scale are possible. Operations, such as sorting or filtering based on attributes, can also be integrated conveniently.

Edge attributes can be encoded in the matrix cells, with a wide range of choices, similar in complexity and expressiveness to on-node encoding in node-link diagrams. The simplest form, beyond a presence/absence mark, is to use color saturation/luminance to show a numerical attribute, such as an edge weight, or to use color hue for a categorical value, such as an edge type [141]. Alternatives encodings include the size of a glyph such as a bar or a circle [142]. Multiple attributes can be encoded with more complicated glyphs [26], [143], as shown in Figure 2.4(b) and Figure 4.8. However, the small space available for matrix cells limits how much can be encoded.

One of the key factors influencing the efficacy of understanding the topology of a network is its seriation, i.e., the ordering of the rows and columns. Numerous studies have examined reordering algorithms [50], [144], [145], [145]–[148]. Different algorithms vary in terms of runtime performance and the types of patterns emphasized [149]. Edge and node attributes could conceivably be considered when calculating the distance matrix underlying the seriation algorithms.

We recommend the following usage for this technique. Adjacency matrices are one of the most versatile approaches with regard to visualizing multiple attributes for nodes and edges (see Section 4.4). Alper et al. [143] studied the efficacy of edge-attribute encodings by comparing edge-weight encodings on node-link diagrams to different edge-weight encodings in the cells of adjacency matrices. They conclude that in-cell encoding in adjacency matrices outperformed on-edge encoding on node-link diagrams for effectively displaying

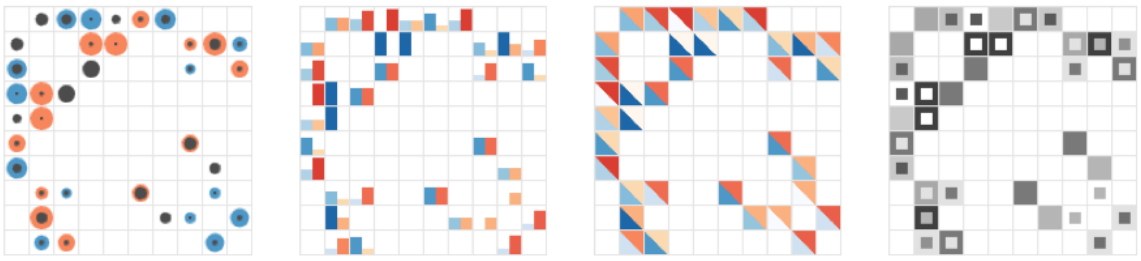


Figure 4.8. Comparison of on-edge encoding for adjacency matrix edge attribute encodings [143].

edge weights for their study subjects.

Adjacency matrices require quadratic screen space with respect to the number of nodes; hence, the size of the network that can be visualized without aggregation is limited. Matrices reserve space for every possible edge, and, thus, dense and even completely connected networks are an ideal fit for matrices. Another challenge of matrices is the complexity of choosing the right reordering algorithm, as different algorithms are best suited for revealing different types of patterns. Matrices are well suited for tasks involving all the topological structures we discuss, except for paths, assuming an appropriate seriation method was applied. Overloaded approaches such as visually superimposing the paths directly on the adjacency matrix can aid in path-related tasks [150]. Trees and layered networks can technically be visualized with an adjacency matrix, but the sparsity of these networks suggests that they are not a good fit. Overall, we recommend adjacency matrices for smaller, complex and dense networks with rich node and/or edge attributes, for all tasks except for those involving paths.

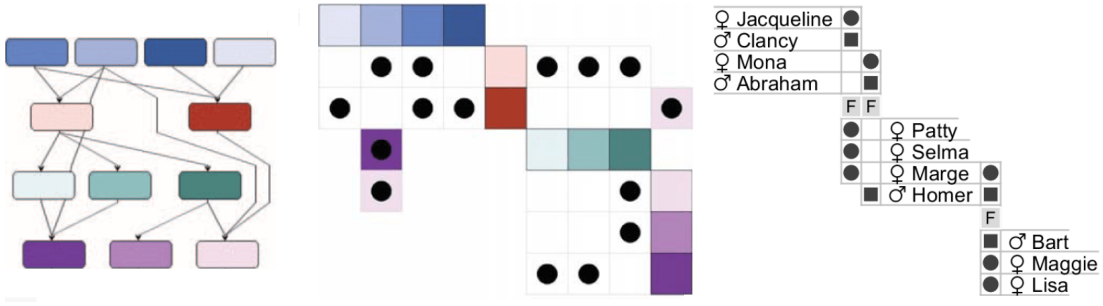
4.3.1.2.5 Quilts. A quilt is a tabular layout optimized for layered networks [93]. Quilts are similar to an adjacency matrix in that nodes are represented as either rows or columns, and edges are shown in the cells at the intersection of the source and target nodes. The main difference between quilts and adjacency matrices is that nodes are assumed to be partitioned into layers, and no links exist within a given layer. Hence, nodes are not duplicated between rows and columns. Figure 4.9(a) shows an illustration of quilts. Quilts alternate rows and column-wise arrangement for nodes for successive layers to optimize the use of space. This design works for strictly layered networks, yet in practice, layered datasets often contain some links to nonadjacent layers, known as skip-links, which are

difficult to encode for layers separated by an odd number of intermediate layers, since these layers have the same orientation. Watson et al. [151] address skip links by encoding the targets of skip links using color and labels. An area of application where quilts have been used is genealogies. GeneaQuilts [138] treats each generation as one layer, and the families founded by that generation as the successive layer.

Quilts have properties similar to those of matrices with respect to node and edge attribute visualization. Node attribute visualizations can be easily juxtaposed with nodes. For example, GeneaQuilts juxtaposes labels and attributes for the sex of the nodes. Unlike in matrices, nodes are in either rows or columns, and hence comparison between node attributes cannot rely on an aligned scale. Edge attributes in quilts can be visualized using the same methods as for adjacency matrices.

We recommend the following usage for this technique. Quilts are well suited for layered networks or k-partite networks in which all partitions have connections to at most two layers. For these kinds of networks, quilts require less screen-space than matrices and have similar favorable properties in terms of attributes (see Section 4.4). Links between nonconsecutive layers, however, can be problematic to integrate. Albeit the class of networks suitable for quilts is small, they support all relevant tasks on these well.

4.3.1.2.6 BioFabric. Biofabric is a tabular layout that places each node in a row of the table and draws edges between the nodes in columns [139] (Figure 4.10). BioFabric has, to our knowledge, not been used to encode attributes, yet the layout of both nodes and edges



(a) (b)

Figure 4.9. The quilts technique. (a) Showing how a small layered network (left) is represented using quilts (right). Links that connect nodes in nonconsecutive layers are represented by colored cells with black dots in them. (b) Simpson family as represented by a quilt [138].

in unique rows or columns makes it suitable to visualize both node and edge attributes by juxtapositoning attribute visualizations, as for adjacency matrices, but in this case for both edges and nodes.

We recommend the following usage for this technique. Biofabric is unique in that it can be used to visualize rich edge attributes and node attributes at the same time, while also making it possible to align these attribute visualizations on the same scale. It therefore has the potential to visualize large attribute datasets and also heterogeneous node types. Biofabric is about as scalable as an adjacency matrix in terms of the number of nodes, but less scalable with respect to the density of the network. Biofabric is not well studied with respect to users' ability to detect topological features, but it is likely slightly more difficult to discover neighbors and clusters in Biofabric compared to in matrices. Overall, we recommend BioFabric for small, sparse networks with many nodes and rich edge attributes.

4.3.1.3 Implicit Tree Layouts

Implicit hierarchical layouts are techniques for visualizing trees that rely on node positioning to encode edges. Well known examples are TreeMaps [37], [152], SunBursts [38], [41], or Icicle Plots [39], but many variations are possible [103]. These layouts excel at visualizing a numerical node attribute as node size, and an additional node attribute as color, but are usually not amenable to encode multiple attributes simultaneously. Since edges are only implicitly encoded, edge attributes cannot be shown.

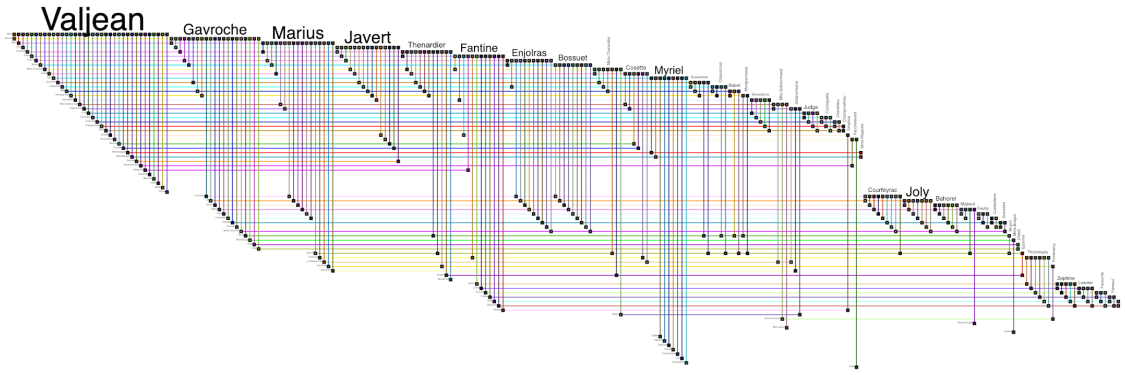


Figure 4.10. BioFabric places nodes in rows and edges in columns [139].

We distinguish between two types of implicit hierarchical layouts: those that show the whole tree, i.e., the inner nodes and the leaves, and those that show only the leaves of the tree.

4.3.1.3.7 Implicit tree layouts for inner nodes and leaves. Implicit layouts such as Sunburst [38], [41] (Figure 4.11(a)) and Icicle Plots [39], [153] (Figure 4.11(b)) show both the backbone of the tree and the leaves. The hierarchy is encoded by adjacency [103]: a child node is adjacent to its root. The root-child relationship is encoded by the order (inner-to-outer for Sunburst, for example). One numerical attribute is used to encode a size parameter, such as the angle in Sunburst and the width in Icicle Plots. A typical example used to demonstrate implicit layouts are file systems, where a root folder contains (sub)folders and (nested) files. Files, the leaves, are assigned a width or angle corresponding to their size. Folders are sized according to the sum of the sizes of their subfolders and files. These techniques assume that the size of the primary attribute in the intermediate nodes corresponds to the sum of the size of their leaves, as this is necessary to ensure proper nesting in the layout. Both intermediate nodes and leaves can use color-coding for a secondary attribute.

We recommend the following usage for this technique. Implicit tree layouts for inner nodes and leaves are well suited for tree datasets with numerical attributes at the leaves and potentially a secondary attribute at inner nodes and/or leaves. In contrast to leaves-only layouts, these layouts support finding neighbors and paths between nodes well (see Section 4.4). The layouts scale well with regard to the number of nodes; however, deep hierarchies tend to use plenty of space. Various intuitive interactions are available, such as successively revealing leaves on demand, to deal with deeper trees. Overall, we recommend implicit tree layouts for inner nodes and leaves for tree datasets where one numerical leaf attribute is dominant, and the tree topology plays an important role.

4.3.1.3.8 Leaf-centric implicit tree layouts. Leaf-centric layouts are those that only or predominantly allocate screen space for the leaves of a multivariate tree and encode the hierarchy by inclusion/nesting [103]. The classical example of this type of layout is the Treemap, originally proposed by Johnson and Shneiderman [37]. Treemaps use the size of a shape to represent a numerical attribute of a leaf node, as shown in Figure 4.12(a). The leaves belonging to one parent are arranged spatially so that their parent is implicitly

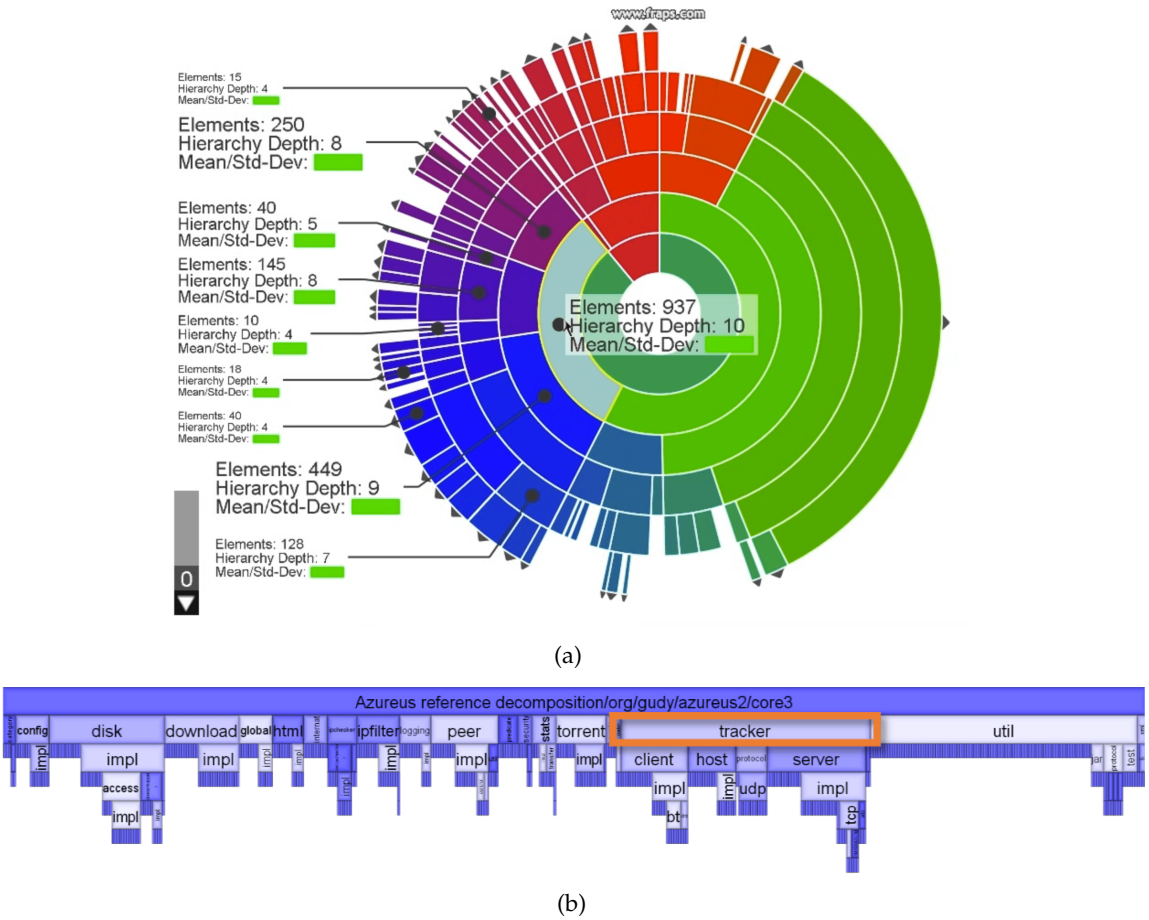
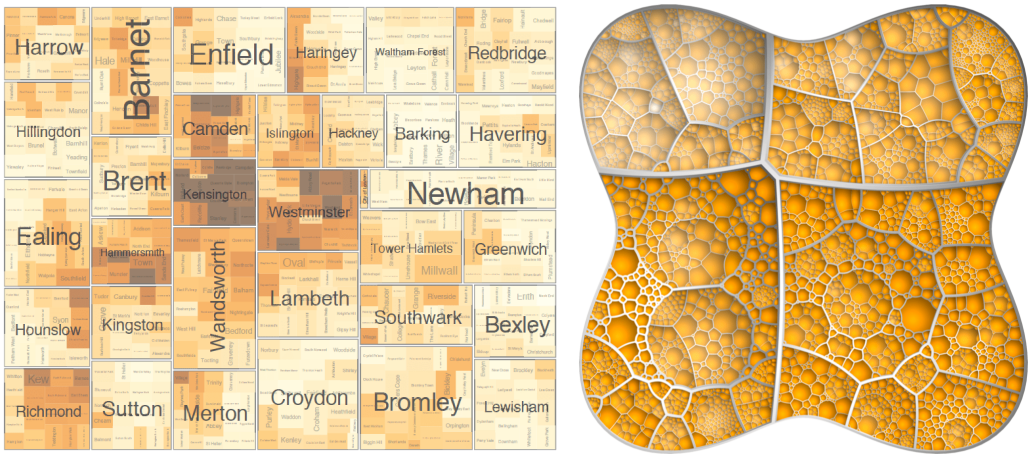


Figure 4.11. Implicit tree layouts for inner nodes and leaves. (a) The Sunburst layout encodes tree topology by adjacency and a primary (numerical) attribute by angle, a secondary attribute by color [38]. (b) Icicle plots also encode topology by adjacency, a primary numerical attribute by node size, and a secondary attribute by color [153].

represented by a discernible shape, usually a rectangle. Variations of Treemap algorithms exist to either represent only inner nodes implicitly by the arrangement of the nodes, or to draw a border outline for inner nodes. A series of different layout algorithms has been developed to improve the readability of the size of nodes and of the topology [155],[156]. A secondary attribute can be encoded as color hue or value/saturation, but other encodings also exist, including glyphs [157] or approximate position [154]. Depending on whether inner nodes are shown or not, color or labels can be used to encode an attribute for them. However, in practice, inner nodes are assigned little space, often only a border, making salient encoding of attributes difficult.

We recommend the following usage for this technique. Leaf-centric implicit layouts



(a) (b)

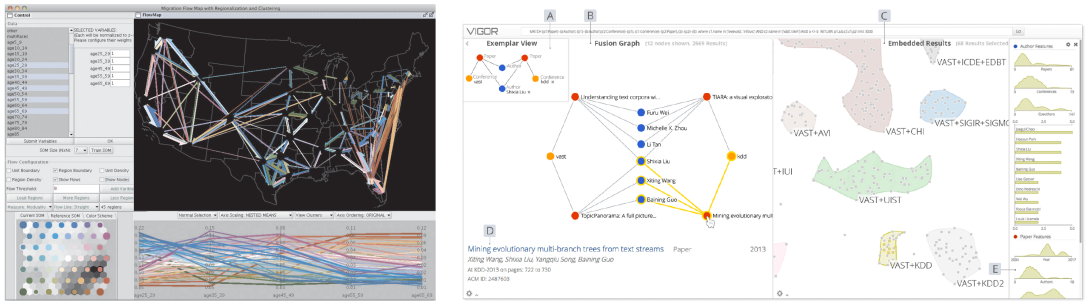
Figure 4.12. Treemap examples. (a) A Treemap layout showing housing data for different boroughs in London. Attributes are encoded by size, color, labels, and approximate position [154]. (b) A Voronoi treemap with thousands of leaves and explicit borders to indicate inner nodes [155].

are well suited for analyzing trees with an important numerical attribute on the leaves. Because this layout assigns most, if not all, display space to the leaves of a tree, and the hierarchical structure groups small elements, it is more scalable than implicit layouts that also visualize inner nodes (see Section 4.4). Treemaps have been used to visualize up to a million items [158]. Path-related tasks can be difficult because the tree structure is often only implied.

4.3.2 View Operations

View operations combine existing techniques using several views, a technique commonly referred to as multiple coordinated views (MCVs) [159], [160]. These approaches use separate, dedicated views for the attributes and the topology. Common examples are combinations of node-link diagrams with multidimensional data visualization techniques [161], [162] or providing a detail view for individual nodes [13], [101].

We distinguish three types of MCVs: juxtaposed, integrated, and overloaded. The juxtaposed and overloaded categories are adopted from the design space of composite visualizations described by Javed and Elmqvist [163]. We consider their category of nested views as a variant of our on-node/on-edge encoding category within layouts. We introduce the separate category, “integrated”, which could be treated as a subtype of juxtaposed, yet is



(a) (b)

Figure 4.13. Examples of juxtaposed views. (a) A geospatial node-link diagram juxtaposed with two attribute views: a parallel coordinates plot and a self-organizing map [121]. (b) VIGOR [167] displays two views for the topology of the network along with a juxtaposed attribute view that shows attribute-specific distributions across the network.

more tightly coupled than typical juxtaposed views, since matches between nodes or edges and their attributes are encoded by their alignment.

4.3.2.1 Juxtaposed Views

In the context of MVN visualization, juxtaposed views separate the topology visualization from the attribute visualization into two or more views. Relationships between the topology and the attributes are not explicitly encoded and typically are revealed through interaction by linking and brushing. The key benefit of juxtaposed views is that each view can do what it does best: visualize either topology or multivariate data. Another benefit is that standard techniques can be employed. Juxtaposed views are widely used in the MVN visualization literature [13], [54], [112], [121], [125], [164], [165]. Juxtaposed views are also one of the few MVN techniques, other than on-node encoding, supported by the major network visualization tools Cytoscape [15] and Gephi [16]. Plaisant et al. [166] showed that whereas a well-designed juxtaposed layout increases user performance, designing effective juxtaposed views can be a challenging task. Figure 4.13(a) shows a juxtaposition of a geospatial node-link view to encode the topology of the network, with a parallel coordinates plot to encode attributes [121]. VIGOR [167], shown in Figure 4.13(b), uses juxtaposition to show two topology-related views and two attribute views.

We recommend the following usage for this technique. Juxtaposed MCVs are recommended for large networks and/or very large numbers or heterogeneous types of node and link attributes (see Section 4.4). Since each view can optimize for either topology or

attributes without concern for the other, the independent analysis of attributes or topology is generally well supported. Linking and brushing can reintroduce the connection, but only for selected items, and even then matches between specific items in a large brushed set are difficult to identify. Consequently, juxtaposed views do not support the tasks on our topological structures well.

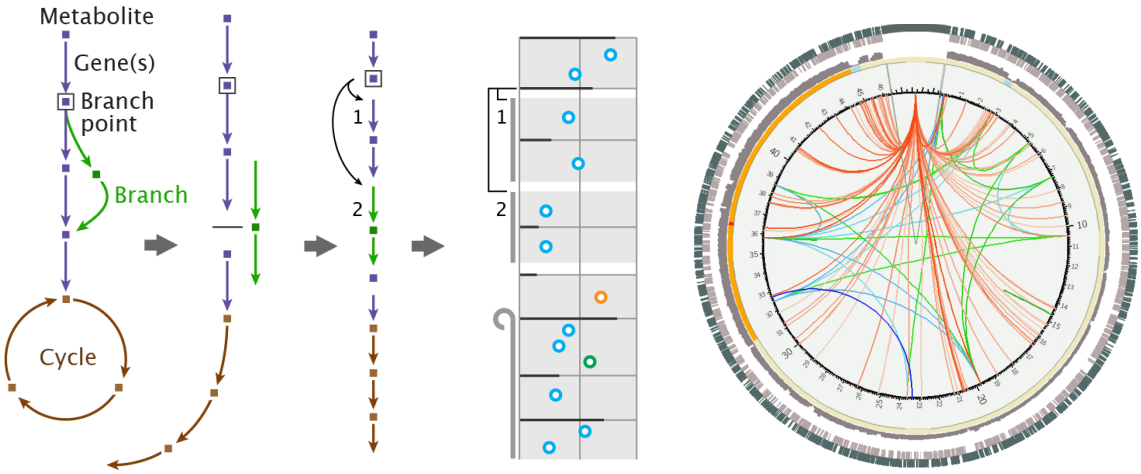
4.3.2.2 Integrated Views

Unlike juxtaposed views, in integrated views the topology and the attribute visualizations are laid out with the other view in mind. Typically, integrated MCVs have an unambiguous spatial relationship between the topological features and their attributes. This spatial relationship is easily achieved in tabular views, such as adjacency matrices [27]: since nodes are in rows and columns, tabular attribute visualizations can be aligned to the rows or columns (see Figure 2.4(a)). Other examples are linear or circular genome views, where edges between genomic regions indicate a variety of relationships [113],[132] (Figure 4.14(b)).

However, integrated views can also use a node-link layout for visualizing the network topology [2], [58], [61], [62], [142], [168]. These approaches use a variety of strategies to make sure attributes and nodes can be juxtaposed. For example, simple networks can be linearized with special encodings [61] (shown in Figure 4.14(a)). For more complex networks, one strategy is to ask users to query for paths and integrate attribute views only with these paths [58], [62], [168], or use interactively extracted spanning trees [2].

Integrated views also work well for trees represented as node link diagrams. These views are commonly used for leaf-only attributes, for example in heat maps juxtaposed with a dendrogram [31], [32], [169], but integrated layouts that also visualize intermediate nodes also exist [1].

We recommend the following usage for this technique. Integrated view approaches are exceptionally good at integrating complex attribute vectors of various types with topology, if the topology can be represented sensibly in a linear layout. Integration is easily achieved for tabular approaches such as adjacency matrices, and for specific types of datasets, such as trees and datasets with a natural linear ordering, such as when using genome coordinates. For general networks and node link approaches, integrated views can usually not



(a) Pathline (b)

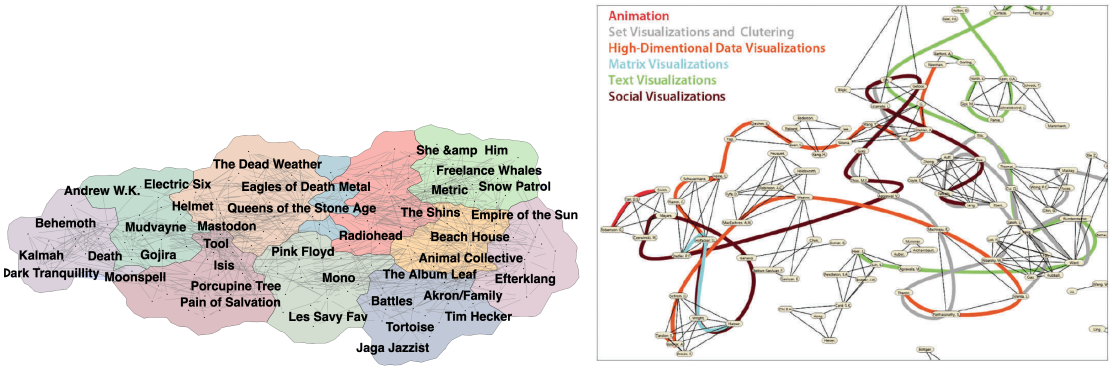
Figure 4.14. Integrated views. (a) Pathline introduces dedicated encodings for cycles and branches to linearize a network and juxtapose it with attribute visualizations [61]. (b) Circos plot of structural genome variations from sequencing data [170].

visualize complex topology, but they can be very useful if the network can be linearized, e.g., using spanning trees or user-selected paths. Compared to juxtaposed views, integrated views excel at tasks related to paths, neighborhoods, and when used with matrices and clusters (see Section 4.4). One drawback of integrated views is scalability with respect to the number of nodes and density.

4.3.2.3 Overloaded Views

Overloaded views are those that display two encodings on top of each other by encoding shared properties of nodes by overlaying visual features on the whole view. A classical example, shown in Figure 4.15(a), is to show a hull around a cluster of nodes (as opposed to coloring the nodes by their cluster membership, which would be on-node encoding). Set memberships are also commonly encoded in a similar way [13], [54], [91], [171]–[173]. Alternatives to the predominant "hull" approach are curves [91], as seen in Figure 4.15(b).

We recommend the following usage for this technique. Overloaded views are well suited for displaying sets, groupings, or clusters on top of an existing representation of the topology of the network. Overloading works best if the grouped elements are also in spatial proximity in the underlying representation, which is commonly the case when visualizing cluster membership. The major limitation of this approach is the limited number of concurrent attributes it supports. Encoding one or two attributes simultaneously is



(a) (b)

Figure 4.15. Overloaded views. (a) Group membership encoded on top of the network topology [172]. (b) Group membership encoded as curves [91].

possible, but encoding more than two attributes with overloading can lead to clutter (see Section 4.4). We hence recommend overloading for the particular use case of visualizing set-memberships or clusters on top of node-link diagrams.

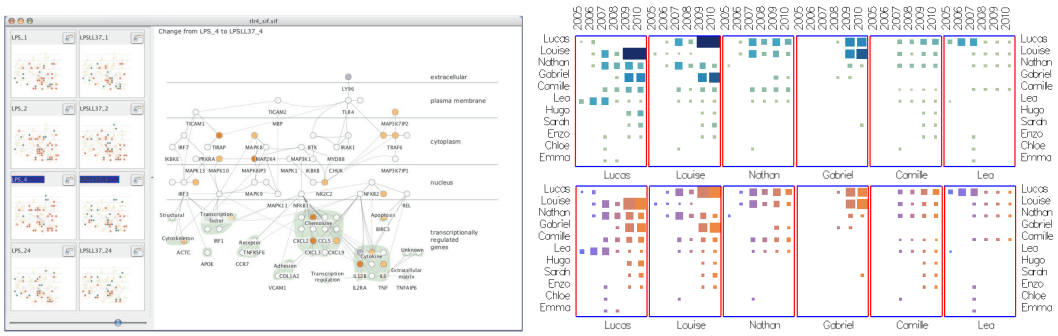
4.3.3 Layout Operations

Layout operations combine layouts to create new multivariate network representations. They are distinct from view operations because they are either integrated in a single view or show different facets of the same view using the same layout.

4.3.3.1 Small Multiples

Small multiples show multiple instances of the same layout under different conditions. Small multiples are commonly used to encode a different attribute for each of the multiple views using on-node/on-edge encoding in node link diagrams [63], [121], [174], [175] (see Figure 4.16(a)) or adjacency matrices [176] (Figure 4.16(b)). Individual views in small multiples can be difficult to see in detail due to their size. A common strategy to avoid this problem is to combine the small multiples display with a large focus view [63], [174].

We recommend the following usage for this technique. Small multiples are well suited for comparing several attributes of a small network. The use of a common layout makes it easy to look for attribute variations in specific topological features, such as clusters or paths. A disadvantage of small multiples is scalability: by rendering a network layout multiple times, plenty of screen real estate is used even for small networks. Also, attribute



(a) (b)

Figure 4.16. Small multiples. (a) Small multiples showing eight experimental conditions using on-node encoding (left) with a focus view showing the network enlarged (right) [63]. (b) Small multiples of a time-varying multivariate network in an adjacency matrix [176].

comparison between different views can be difficult and require memorization.

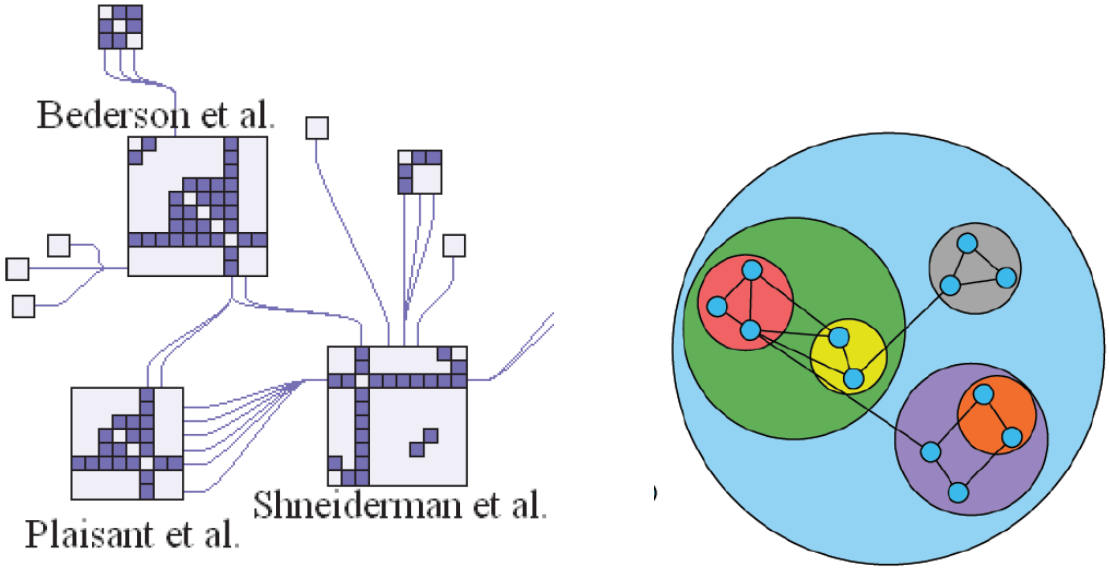
4.3.3.2 Hybrid Layouts

Hybrid layouts combine multiple approaches to laying out network topology. Unlike the techniques using *view operations*, these layouts are characterized by combining different encodings for different portions of the topology of a network. Hybrid approaches are used to either represent networks that have different topological characteristics in different portions of the network [24], or to highlight a certain topological aspect of the network, such as a hierarchy embedded in a network [177], [178].

NodeTrix [24] is a hybrid node-link/matrix approach (Figure 4.17(a)). The idea is to use matrices for dense subsets of the network but node-link layouts to connect these subsets or connect to outliers. Various techniques combine a Treemap encoding for the hierarchical structure of the network with other network encodings [178], [179] (see Figure 4.17(b)).

Hybrid techniques can also be used to encode multivariate attributes. For example, NodeTrix [180] supports on-node encoding for the node-link/matrix hybrid.

We recommend the following usage for this technique. Hybrid layouts can be useful for networks with an irregular degree distribution, such as small-world networks. However, the benefits of using an optimal layout for each part of the network has to be weighed against the cost associated with understanding and reading a visualization that combines multiple visualization techniques. Hybrid techniques are not well suited for visualizing multivariate networks, since also encoding attributes increases the complexity of the visu-



(a) (b)

Figure 4.17. Hybrid approaches. (a) Hybrid node-link layout and adjacency matrices [24]. (b) Hybrid of a circular Treemap and a node-link layout [178].

alization technique further, and a different encoding schema likely has to be used for each of the techniques employed.

4.3.4 Data Operations

Multivariate network visualization techniques often rely on data wrangling operations invoked either as a preprocessing step or interactively during the analysis. Here we introduce five types of data wrangling operations that we frequently encountered in the surveyed papers. However, many other operations are conceivable. For a more comprehensive discussion of network wrangling operations, refer to the work by Bigelow et al. [181].

4.3.4.1 Aggregating Nodes/Edges

Aggregating nodes and edges is a common strategy to increase the scalability of a visualization technique and to summarize nodes and edges with shared characteristics. Aggregation also allows the visualization to more easily support overview tasks on the network. Techniques such as PivotGraph [111], Zame [26], and Motif Simplications [116] employ aggregation strategies to succinctly visualize important properties of large net-

works.

Aggregation can be based on topological features, such as clusters or motifs [116], but also be driven by network attributes. PivotGraph, for example, aggregates the nodes by a categorical attribute (node type) and then lays out the aggregate nodes according to their attribute value on a grid-like node-link layout (Figure 4.18(a)). Van den Elzen and van Wijk [17] aggregate nodes based on an attribute, such as a geographic region, and then display the aggregated nodes, their attribute properties, and the relationships within and between the aggregates (Figure 4.18(b)). Zame [26] aggregates the nodes as well as the attributes, which are then displayed in an adjacency matrix. An interactive approach to aggregation is used by GraphCharter [101], which allows a user to select an attribute of interest and generates an aggregate node with all unique values for that attribute. Graffinity [27] introduces a novel approach to connectivity exploration by aggregating certain paths that connect pairs of nodes in the network.

We recommend the following usage for this technique. Aggregation is a key operation to ensure the scalability of MVN visualization and is especially useful when high-level overviews are desired. Aggregation by attributes can reveal interesting relationships between attribute properties and topological features. Aggregation can be achieved in a preprocessing step, yet it is most powerful when invoked interactively.

4.3.4.2 Querying and Filtering

Filtering a network or querying for a subnetwork allows users to focus on nodes and edges of interest, reducing visual clutter from other nodes and edges. Querying or filtering enables analysts to work with much larger networks than can be reasonably displayed otherwise. Filters and queries can be defined based on topological structures, attributes, or combinations of both [167], [182]. Querying first and expanding a network from a seed is a hallmark approach of many large network MVN visualization approaches [2], [62], [89], [102]. These approaches are often combined with degree-of-interest functions [183], [184] to further manage network size (see Figure 4.19).

Filtering edges is a valuable operation, especially in dense networks. EdgeMaps [133], for example, reveals edges only for selected nodes, whereas the technique by van den Elzen and van Wijk [17] displays only edges that connect selected nodes, as shown in

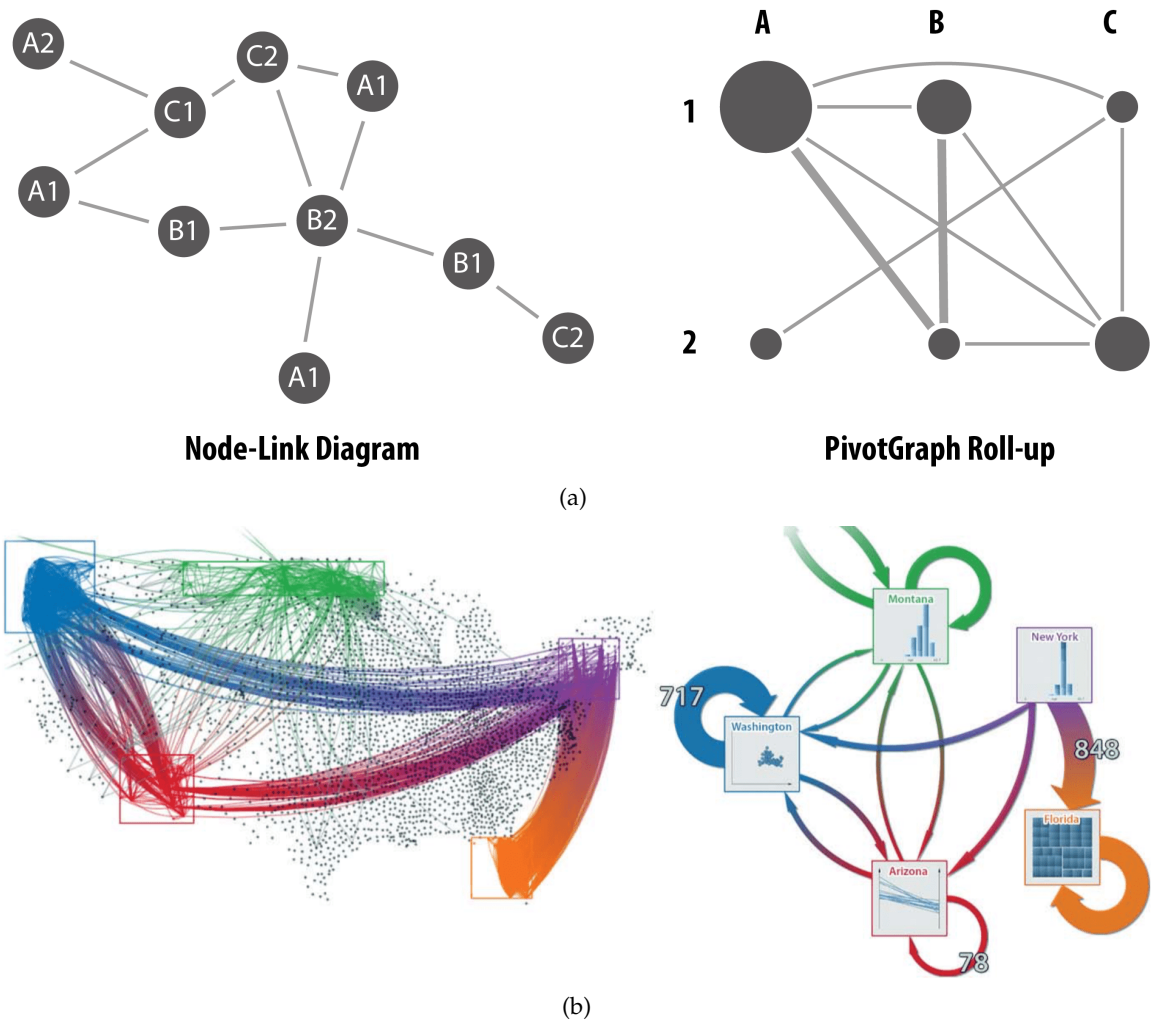


Figure 4.18. Aggregation operations. (a) Original node-link network on the left, a network aggregated based on attributes on the right [111]. (b) Geospatial network on the left, an aggregated version with on-node encoding on the right [17].

Figure 4.18(b).

We recommend the following usage for this technique. Although interactive filtering based on topological and attribute features can and should be available in every MVN visualization approach, query-first strategies follow a different analysis paradigm. The benefit of the latter is that very large networks that cannot be reasonably visualized at once can be investigated with this approach. We recommend a query-first strategy for cases where analysis questions target specific nodes and relationships, rather than overall network patterns.

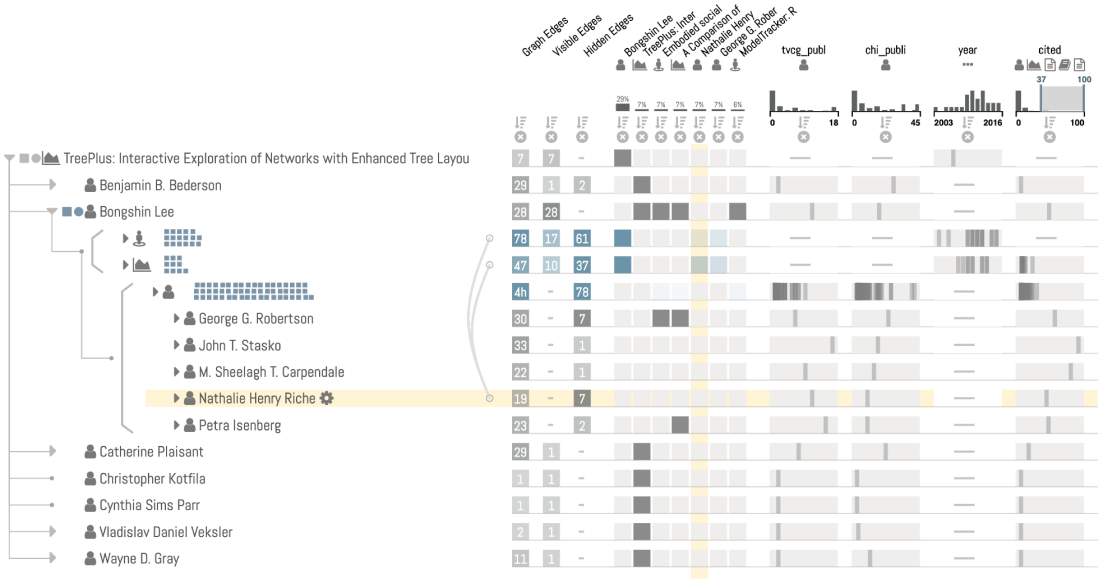


Figure 4.19. A query-first approach to network visualization: visualizing a spanning tree originating from a node of interest, combined with a degree-of-interest function to manage large node sets [2].

4.3.4.3 Deriving New Attributes

Node and edge attributes can be derived either from topological features (e.g., node degree, clusters) or from other attributes. Derived measures can be helpful in, e.g., finding the most connected node in a network or quickly identifying clusters of interest. Once these derived attributes have been computed, they can be visualized just like other attributes. Examples of techniques that rely on derived attributes are Juniper [2], which visualizes node degrees for various subsets (Figure 4.19), and EdgeMaps [133], which uses a precomputed similarity metric to position the nodes in an attribute-driven node link view.

We recommend the following usage for this technique. Attribute derivation can be useful to answer specific analysis questions. Derivations that are local to a node, such as determining degree or deriving an attribute based on existing attributes, are easy to realize. However, derivations that rely on topological features, such as clustering, or deriving attributes based on connectivity patterns, require more sophisticated methods.

4.3.4.4 Clustering

Network clustering is used to group nodes based on similar features. In most cases, clustering of networks is based on network topology. Social network analysis, for example, often employs hierarchical clustering in order to find community structures in the network. Clustering can be used to highlight groups on existing networks, i.e., to create a derived attribute, or as a precursory step to aggregation. Techniques such as Zame [26], Honeycomb [51], and TreeMatrix [141] all rely on hierarchical clustering on the network before visualization. Zame and Honeycomb visualize the clustered network with zoomable adjacency matrices, whereas TreeMatrix displays the results in a hybrid approach using both Treemaps and adjacency matrices.

Hierarchical algorithms applied to tabular datasets also generate trees encoding the cluster similarity, which is a frequent data type in multivariate tree visualization [31], [32].

We recommend the following usage for this technique. Clustering can be used to create derived attributes, highlighting topological features, to improve layouts, e.g., in matrix seriation or to create groups for faceted layouts, or as a precursor step to aggregation. It is a widely used and versatile operation in multivariate network visualization.

4.3.4.5 Converting Attributes/Edge to Nodes

Converting attributes or edges to nodes is a useful operation in various scenarios. Many visualization techniques are better suited to visualize node attributes than edge attributes, and hence converting an edge to a node with new edges connecting the original nodes is a convenient way to visualize edge attributes [45]. Making an attribute into a node can aid in the analysis of which nodes share that specific attribute [185].

We recommend the following usage for this technique. Converting attributes or edges to nodes is a powerful operation in the right circumstances but is also a significant reshaping operation, which risks confusing analysts. Hence, it should be applied with care. We believe that it is most useful as a preprocessing operation in dedicated network wrangling tools [181].

4.4 Guidelines for Visualizing MVNs

In this section, we provide general guidelines on visualizing multivariate networks, complementing our technique-specific recommendations given in Section 4.3.1 and our recommender system on the accompanying website (`https://vdl.sci.utah.edu/mvnv/`).

In order to support these guidelines, we scored each of the layouts and view operations along 20 dimensions, which include network characteristics such as size and number of attributes, as well as different types of MVN analysis tasks. Table 4.1 shows the scores for each type. Our methodology for developing these scores is described in Section 4.2.4. We used the following scores:

- 0 This technique **cannot** support this data type or task.
- 1 This technique supports this data type or task very poorly.
- 2 This technique can support this data type or task, but is not ideal and may require interaction to achieve it.
- 3 This technique supports this data type or task very well.

One of the major challenges that MVN visualization techniques face is scaling to large networks (Table 4.1) and large attribute datasets. For large *trees*, treemaps are very well suited, with the caveat that they are tailored to displaying only a few attributes and only for leaves. For other types of networks, MVN techniques typically do not scale to networks with over 1,000 nodes, especially when attempting to show both attributes and network topology. *Data operations* can be very useful to either reduce the subset of the network the user is looking at, through a query-first or filter approach [2], [101], or to aggregate the network into a more manageable size [26], [111]. These data operations can be done through either preprocessing of the network or interactive operations directly in the visualization.

Juxtaposed views scale to large networks and many attributes, since they allow for a topology-only view that can optimize the layout and an attribute-only view well suited to display multivariate data. However, having separate views for topology and attributes comes at the cost that these systems often perform poorly on tasks that require integration between attributes and topology.

A scalable technique with regard to the number of node and edge attributes and node-/edge heterogeneity is *integrated views* that align a node/edge with a tabular representa-

Table 4.1. Scores for how well each technique performs on different network types and different multivariate network tasks. **0** means **does not support at all**, **1** is **supports poorly**, **2** encodes **supported but with limitations / may require interaction**, and **3** means **well supported**.

		Size			Type				Node Attr.				Edge Attr.				Top. Struct.				
		Small <100 nodes	Medium (<1,000)	Large (≥1,000 nodes)	Complex (sparse)	Complex (dense)	Layered/K-Partite	Trees	Few (<5)	Several (≥5)	Homog. (1 type)	Hetero. (>1 type)	Few (<3)	Several (≥3)	Homog. (1 type)	Hetero. (>1 type)	Single node/edge	Neighbors	Paths	Clusters	Entire/sub network
Node-Link Layouts	On-node encoding	3	2	1	3	1	3	3	2	1	3	2	2	1	3	1	3	3	2	2	2
	Attr. Facet.	3	1	1	3	1	3	1	3	1	3	3	2	1	2	1	3	2	1	1	1
	Attr. Pos.	3	1	1	3	1	1	1	3	1	3	1	2	1	2	1	3	2	1	1	2
Tabular Layouts	Adj. Matrix	3	1	1	2	3	2	1	2	3	3	2	3	2	3	2	3	3	1	3	2
	Quilts	3	1	1	3	1	3	3	3	3	3	3	3	3	3	2	3	3	2	2	2
	BioFabric	3	1	1	3	3	2	1	3	3	3	3	3	3	3	3	3	1	1	1	2
Implicit	All nodes	3	2	1	0	0	0	3	3	1	3	1	0	0	0	0	3	3	3	0	3
	Leaves	3	2	2	0	0	0	3	3	1	3	1	0	0	0	0	3	2	1	0	3
View Ops.	Juxtaposed	3	2	1	3	1	3	3	3	3	3	3	3	3	3	3	2	1	1	2	2
	Integrated	3	2	1	3	1	3	3	3	3	3	3	2	2	3	3	3	3	3	1	2
	Overloaded	3	2	1	3	1	3	3	3	1	3	1	1	1	1	1	3	3	2	3	2

tion. These techniques are easily combined with all tabular methods or can be combined with node link layouts when leveraging interaction. Along with juxtaposed views, integrated views are suitable for networks with several attributes. The tight visual coupling of the topology and attribute of the network make integrated networks the highest rated techniques for performing MVN tasks on non-tree networks. We recommend adjacency

matrices with integrated attribute views for dense networks with many attributes, and node-link networks, combined with integrated attribute views and interactive data operations, for small networks.

Interaction is useful for most techniques, but it is particularly important for node-link representations when combined with juxtaposed or integrated coordinated views. Juxtaposed views rely heavily on interaction to link the components in the topology and attribute views through linking and brushing. Integrated views, on the other hand, are based on a linear ordering of the nodes in the network. This linear ordering can be the product of user selection by picking subsets of the networks, such as spanning trees or paths. For these subsets, integrated views provide excellent integration of topological features and attributes.

4.5 Conclusion

In this chapter we presented a typology of techniques for visualizing multivariate networks. The typology was generated through a qualitative analysis of 210 papers published within the visualization research community and lays out the four dimensions along which MVN visualizations can be characterized. We believe the typology will be useful for practitioners to understand and choose appropriate multivariate network visualization techniques, and for researchers to identify areas that would benefit from further development.

We hope this typology allows visualization designers to more easily compare existing approaches to MVN visualization, as well as to leverage our guidelines on which techniques are best suited for given network types and specific exploration tasks. MVN visualization is a constantly evolving field, providing visualization researchers rich and interesting open problems.

CHAPTER 5

DESIGN STUDY — LINEAGE: VISUALIZING MULTIVARIATE CLINICAL DATA IN GENEALOGIES

Studying ancestry and familial relationships, i.e., genealogies, is both a pastime enjoyed by amateurs and a research area for professionals [186]. It is hence not surprising that numerous tools are available to record and visualize genealogies. Yet, most of these tools focus on analyzing family structures for historical purposes, and only a few target a clinical use case of analyzing genealogies in the context of complex, hereditary diseases. Geneticists, on the other hand, have long used genealogical networks to study how a genetic disease manifests itself in families. They use drawing conventions and standardized symbols to show both the family structure and the phenotype, i.e., the observable characteristics of an individual [187]. These charts can provide insights into the heritability and segregation patterns of genetic diseases. In their current form, however, they are predominantly useful for Mendelian diseases, or genetic diseases caused by a small number of mutations. Complex diseases such as cancer, autism, diabetes, obesity, and psychiatric conditions such as depression or suicide are known to have hereditary components that are regulated by a multitude of genes, each having a modest effect on risk, and also to depend strongly on environmental conditions and chance. When studying these conditions in a population, it is imperative to simultaneously consider genetic similarities, shared characteristics of the phenotype, and environmental conditions. Also, for these polygenic conditions, one needs to consider significantly larger populations to reason about hereditary relationships and pursue the discovery of genetic risk mutations.

Current medical or historical genealogy visualization tools are ill equipped to help researchers find patterns in these large, highly multivariate networks of families and their rich medical histories. In this chapter, we present a novel genealogy visualization tool that

we have developed in collaboration with psychiatrists and geneticists studying the genetic underpinnings and the environmental factors of suicide and autism. We use data from the Utah Population Database, a uniquely rich resource for population-based analysis of hereditary diseases.

We contribute a novel technique to visualize large, tree-like networks (rooted, directed networks that have some cycles but are predominantly in tree form) associated with rich numerical, categorical, and textual attributes. Our approach leverages the tree-like structure of the networks to produce a linearized layout that enables the direct association of the nodes with rich attributes in a tightly integrated tabular visualization. We address the issue of scalability by introducing novel forms of degree-of-interest-based aggregation that preserve the structure of the network, and, if desired, also provide an overview of the attributes of aggregated individuals. We demonstrate our technique in the context of genealogical data, and we argue that it can be equally applied to other multivariate trees or tree-like networks.

We also contribute a detailed characterization of the domain problems and the domain data as they are encountered when analyzing large, clinical genealogies[1] and a set of task and data abstractions derived from these characterizations. Finally, we contribute the open-source Lineage visualization tool, shown in Figure 5.1, which implements the technique, and describe multiple design decisions tailored to genealogical data visualization.

Lineage is in the process of being adopted by our collaborators, and has undergone iterative design refinements. We have also demonstrated it to other research groups working with genealogical and genetic data and have encountered overwhelming enthusiasm. We validate this work in an illustrative usage scenario and through qualitative user feedback from domain experts.

5.1 Domain Background and Data Characterization

Our collaborators study the genetic underpinnings and the environmental factors influencing psychiatric conditions, such as autism and suicide, using detailed genealogical, clinical, and genetic data. In this chapter, we will focus on suicide, yet our methods are

[1]The terms genealogy and pedigree can be used interchangeably in this context. However, for simplicity, we will always use genealogy.

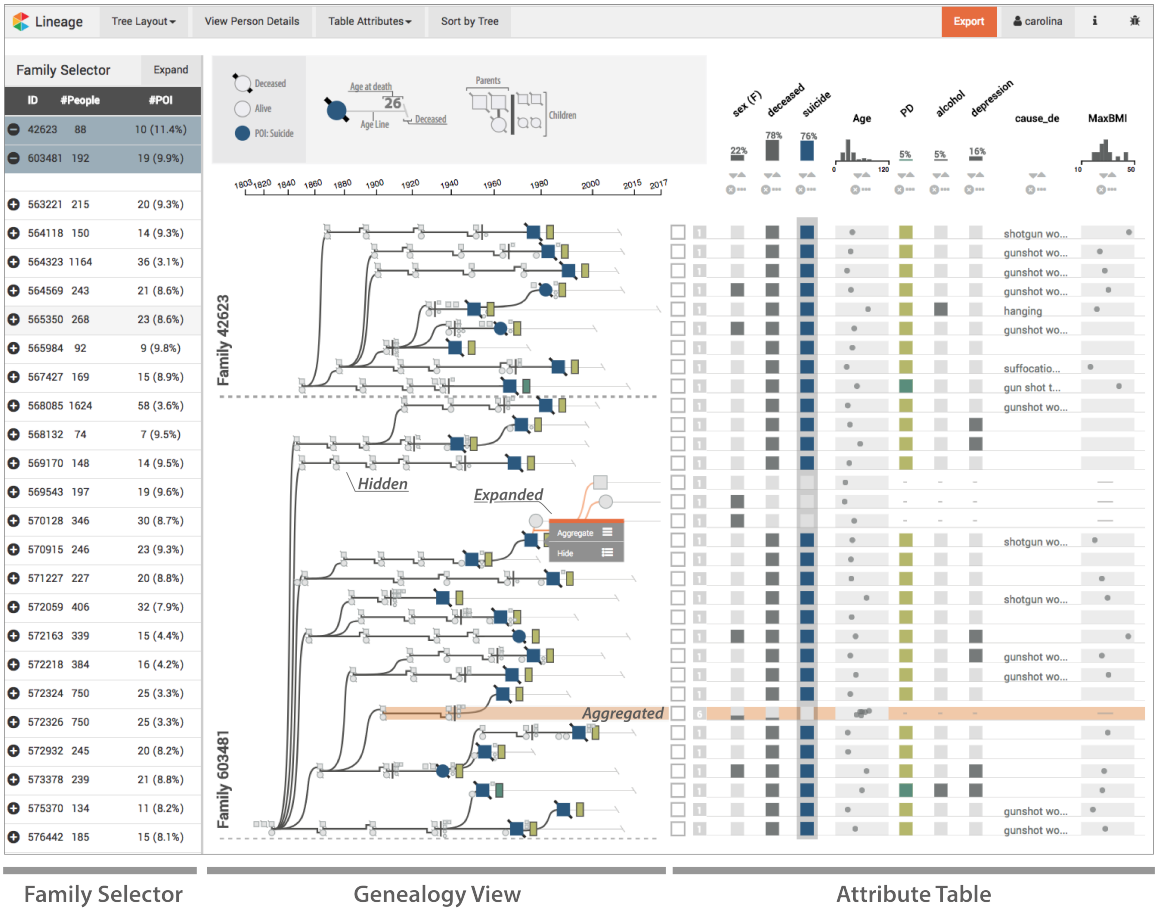


Figure 5.1. Lineage visualizing the genealogy of two families with increased numbers of suicides. The genealogy view shows the family relationships in a linear tree layout, where each node corresponds to a row in the associated table. Suicide cases are highlighted in blue, and a glyph next to the nodes indicates whether individuals were diagnosed with a personality disorder. The branches use different levels of aggregation (hidden, aggregated, expanded). The table shows detailed attributes about individuals, or, when branches are aggregated, for groups of individuals.

easily transferable to other complex, multifactorial conditions and diseases. Suicide is a high-impact application, as it is one of the leading causes of life-years lost [188] and the 10th most common cause of death in the United States [189]. Suicide is believed to be caused by a complex combination of risk factors, including environmental stressors and genetic vulnerability. Aggregated data across multiple large studies have produced a heritability estimate of completed suicide of 45% [190], [191]. Genetic risk factors for suicide are complex and can be classified as multiple subtypes. These subtypes often are characterized by co-occurring psychiatric conditions (comorbidities) and/or a combined risk of psychiatric diagnosis. For example, the genetic risk for schizophrenia is also asso-

ciated with a risk for suicide [192].

Our collaborators have compiled a unique dataset of suicide cases, including DNA and clinical profiles on 4,017 cases. These cases are linked to the Utah Population Database (UPDB), which provides genealogical data. Genealogies describe the familial relationships of individuals across multiple generations.

Figure 5.2 shows two genealogies using standardized drawing conventions [187]. Females are drawn as circles, males as squares. Couples are connected by an edge, and children connect to this edge using orthogonally routed links. The vertical position of nodes is given by their generation. A phenotype of interest is marked by a filled-in node. When studying family relationships, a common approach is to draw family trees considering the ancestry of an individual. Figure 5.2(a), for example, shows the family of the woman marked in black. The genealogy includes her two siblings and traces her family tree for two generations to include parents, uncles and aunts, and grandparents.

In contrast, our collaborators are interested in understanding genetic relationships between individuals afflicted with a condition and hence care about individuals who share genetic variants. They select families for study that have a statistically increased rate of a condition. These family trees are constructed by tracing cases back to a "founder," as illustrated in Figure 5.2(b). The underlying hypothesis is that such founders have genetic risk variants that they passed on to their descendants. Within the genealogy, the likelihood of genetic homogeneity is increased, and is more easily detected through

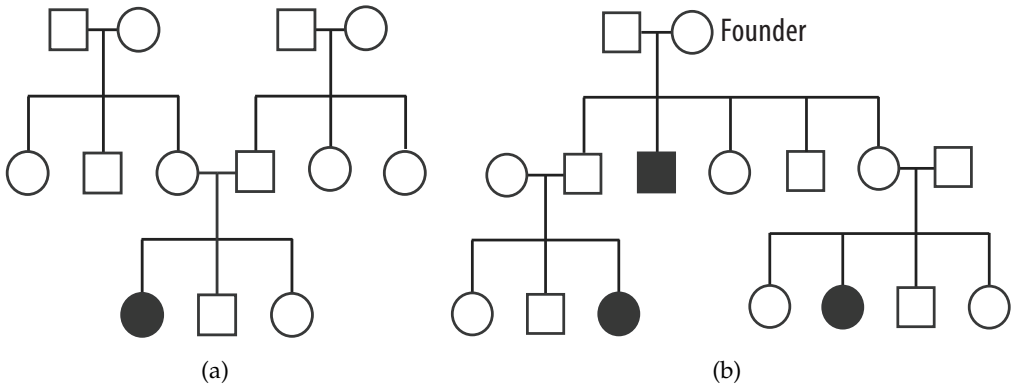


Figure 5.2. Two genealogies using standardized symbols for different aspects of the family structure. Females are shown as circles, males as squares. Individuals with a phenotype of interest are filled in in black. (a) A genealogy showing the family of the female in black, including siblings, parents, uncles/aunts, and grandparents. (b) A genealogy based on a founder, tracing down generations to include the families of individuals with a phenotype of interest (black).

the repeated occurrence of the genetic risk variant in the familial cases. Note that this genealogy contains only individuals who are descendants of one founder and his or her spouse, with the exception of spouses of descendants. Also, the dataset contains only individuals with direct links to a case, i.e., siblings, descendants, and direct ancestors are included, whereas, for example, uncles/aunts and cousins are not.

The dataset our collaborators have compiled contains about 19,000 suicide cases, including 4,585 recent cases with detailed data, backed by family structures made up of 118,000 individuals from 551 families. Suicide is frequently associated with psychiatric comorbidities, i.e., co-occurring chronic conditions, such as depression, bipolar disorder, substance abuse, PTSD, or schizophrenia [192]. Also, nonpsychiatric conditions such as asthma [193] may play a role in some cases. Environmental factors, such as socioeconomic status, pollution, and seasonality, are also known to be factors in suicide [194]. To capture this information, our datasets include demographic variables such as gender, race, age at death, method of death, family demographics (marriage, divorce, number of siblings/children), and place of residence at the time of death. The datasets also include records of other diagnoses captured as codes from the International Classification of Diseases (ICD) systems, the frequency with which these diagnoses were made, and the time of the first diagnosis.

To summarize, each of our many networks describes a family, with individuals as nodes and family relationships as edges. Since the networks are constructed by tracing ancestry to a founder, they are predominantly tree-like, but they do include cycles, for example, when two cousins have offspring. In addition, we have attributes on the individuals/nodes in the networks of various data types, including numerical, categorical, temporal, geographic, and textual data. These attributes are often sparse because only about 10% of individuals in the dataset have committed suicide, and our detailed records extend to only about 2% (4,017) of individuals across all families. These detailed records capture about 3,000 dimensions that contain demographic information and information about the manner of death, but predominantly contain comorbidities in the form of disease codes and the time and frequency of these diagnosis. These dimensions are often sparse because a diagnosis such as “depression” can be recorded using one of over 30 ICD codes.

5.2 Goals and Task Analysis

This project is rooted in a collaboration with faculty, clinicians, analysts, and graduate students in the Department of Psychiatry at the University of Utah. Six domain experts participated in the project. We loosely followed the design study methodology proposed by Sedlmair et al. [195]. Our "discover" phase consisted of multiple meetings with individual collaborators and with the whole group as a team, studying the domain literature and the tools they currently use.

We also ran a creativity workshop, specifically the *wishful thinking* component described by Goodwin et al. [196], involving all the collaborators. In the workshop, we asked participants to think about the analysis of suicide data and then discuss in small groups and take notes on post-its about what it is they would like to *know*, *see*, and *do*. This idea-generation phase was followed by a phase in which the teams had to prioritize their insights and then finally give the whole team an overview of their key ideas. We recorded the workshop and transcribed both the audio and the post-its. We then coded the artifacts and three themes emerged: the *data*, the *factors involved in suicide*, and the *analysis tasks*. The insights on the data and the factors involved in suicide were described in the previous section.

The overarching goal of our collaborators is to gain a better understanding of the determining or associated factors of suicide. Our collaborators classify the factors associated with suicide into comorbidities and demographic, genetic, and environmental factors. Specifically, they are interested in identifying and defining detailed phenotypes associated with suicide and the degree to which these phenotypes are familial. By finding people who are similar to each other in a relevant way, our collaborators hope to reason about genetic homogeneity, i.e., shared genetic factors contributing to suicide. They currently rely only on familial structure as a proxy for genetic homogeneity. However, they recognize that this approach is limited both as too broad — it is possible that they should consider only a part of a family — and as too narrow — people outside a family who have a similar phenotype could also have a similar genotype. Robust and detailed phenotypes are, of course, also interesting by themselves, because they can be used, for example, as part of a risk assessment in a clinical context.

It is important to note that the contextual knowledge of a researcher is beneficial to the task of classifying a phenotype. For example, a diagnosis of depression is weighted

differently if it is diagnosed dozens of times and was first diagnosed early in a patient's life. Similarly, a young person who commits suicide in a rural community is unlikely to have a detailed medical history. Hence, such a case could be similar to others, even if certain phenotypes are not recorded, if other factors, such as a close familial relationship, indicate it.

Our collaborators need a visualization tool that is embedded in a larger analysis process, one that includes calculating statistically significant familial risk (upstream) and searching for shared genetic variants (downstream). They need a tool that focuses on finding individuals and families that are "interesting" with respect to both their relatedness and their attributes, which can then be used in further analysis and validation.

We identified the following domain tasks as the most important aspects in the workflow of our collaborators:

T1 Select families of interest. The analysts want to select a family by browsing, by selecting a specific family based on prior knowledge, or in a data-driven way. An example of the data-driven approach is to find families with high rates of suicide or with individuals for whom suicide co-occurs with bipolar disorder.

T2 Analyze individual case. Our collaborators need to investigate the context of a case. For example, a potential genetic component contributing to suicide is judged differently if the person had many psychiatric comorbidities and committed suicide at a young age, compared to a late-life suicide of a person with a terminal disease.

T3 Compare cases. This task encompasses comparing individuals and identifying shared attributes to characterize a potentially meaningful shared phenotype. It also pertains to analyzing how the individuals are related, which can indicate the likelihood of shared genetic traits. Insights on shared environmental factors can be gleaned from both the family structure and the attributes. For example, siblings are likely to be exposed to the same environment in their childhood, whereas cousins might not. Similarly, two people living in the same area are potentially of similar socioeconomic status.

T4 Judge prevalence and clusters of phenotype. The number of suicide cases and the prevalence of comorbidities vary greatly between families and between branches of

a single family. Judging how common a phenotype is in a family or in part of the family is helpful in identifying subsets of interest for further study.

T5 **Compare families.** Once an interesting observation has been made in one family, our collaborators want to be able to investigate whether similar cases also appear in other families. For example, when an association of asthma with suicide is discovered, they want to know whether it is isolated in one family or occurs in multiple families and/or individuals.

T6 **Quality control.** Although not an analysis task per se, our collaborators also need to judge the quality of the data and report errors back to the central database. A common data error we have seen is disconnected components or detached nodes, which are caused by missing information about an individual's mother or father.

Most of these domain tasks rely on both studying the topology of the network, i.e., the family relationships, and investigating the attributes associated with the individuals. For example, the "compare cases" task (**T3**) relies on both the network and the attributes to, for example, reject an outlier in an otherwise well-defined phenotype within a family, if that outlier is only distantly related to other cases.

5.3 Related Work — Genealogy Visualization

Genealogical charts, as shown in Figure 5.2, are widely used in genetic counseling and the literature on genetic diseases. They are well suited to visualize a single phenotype of interest, but they are not suitable to map a complex phenotype to the node. Our collaborators currently use Progeny [197], a commercial genealogy drawing tool that closely follows the standard for visualizing genealogies [187], [198]. (See the supplementary material for an example figure created with Progeny.) Although Progeny is well suited to draw these standard genealogies for use in presentations, it is ill suited for exploratory tasks, mainly because of its inability to efficiently encode attributes in the network.

Interactive genealogy visualization tools that are designed to analyze disease clusters and to see disease propagation within families include PedVizApi [199], CraneFoot [200], Haploview [201], PediMap [202], and HaploPainter [203]. HaploPainter [203] visualizes genealogies and genetic recombination events below the individuals' nodes. Although it

shares the approach of showing metadata as rows associated with nodes with Lineage, it does not take a linearization approach to make values of different generations easy to compare, it does not aggregate the network, and it does not visualize different types of attributes. McGuffin and Balakrishnan [204] describe layout algorithms for complicated genealogical trees and introduce aggregation methods for subtrees, which we adopt.

Among tools that do not use the standard genealogical drawing conventions are Fan Charts [205], which uses the SunBurst technique to visualize genealogical trees, and the work by Mazeikla et al. [206], which employs a force-directed layout that considers similar phenotypes as additional attracting forces. Tuttle et al. [207] use an H-tree layout for scalable genealogy visualization, with the founder at the center and successive generations radiating out based on a fractal pattern. Ball [208] employs the idea to not represent generations as discrete units but to use time to position the nodes, and also to draw a person's life span. Kim et al. introduce TimeNets [209], a technique also focused on the temporal aspects of a genealogy. Although TimeNets is well suited to observe temporal changes in relationships between individuals, relationships between generations are harder to trace. The recent work by Fu et al. [210] focuses on visualizing the distribution of tree structures in many families. The tool combines a Sankey diagram showing properties of tree structures with explicit node-link diagrams on demand, but does not consider attributes of the nodes.

GenealogyVis [211] is a recent tool for visualizing genealogies to study historic data. Although it visualizes multivariate attributes, it addresses different needs — those of historians — and uses different approaches. Unlike in Lineage, attributes of individuals are not shown; rather, the focus is on demographic trends in (parts of) the network. Supplementary views, such as scatterplots and maps, allow historians to study, for example, migration patterns.

Genealogy visualization tools for animal genealogies face a different set of challenges compared to those for human genealogies, as the number of descendants sired by individual animals can be large, and complex interbreeding is common. Consequently, tree-based approaches are not well suited for these genealogies. Examples include CoVE [212] and VIPER [213]. VIPER introduces a sandwich view that CoVE also adopts. The sandwich view scales well to many descendants of an individual, but only explicitly encodes the re-

lationships between parents and their children. More distant relationships can be revealed through highlighting. Helium [214] is a visualization technique for plant genealogies, which commonly have complex crossing. It uses color coding and scaling of nodes to encode up to two attributes.

GeneaQuilts [138] is a matrix-based technique where each row constitutes a person and each column a nuclear family. In early stages of our design process, we considered using a GeneaQuilt instead of our node link design, since GeneaQuilts produces a linearization of the network that would be suitable for associating attributes. We ultimately decided against it because: 1) the data we consider for the analysis of genetic relationship is predominantly tree-like, and hence, the complex design of GeneaQuilts that is necessary to accommodate general genealogical networks is not justified for our simpler, tree-like datasets; 2) a key analysis task for the network view is to judge the degree of relatedness between two nodes, which is not well supported by GeneaQuilts without interaction; and 3) our design for aggregation is more suitable for node-link diagrams.

A different approach to analyzing relatedness is to calculate "kinship coefficients" between individuals, i.e., to calculate path-based metrics for relatedness and visualize them in a matrix [27]. Although this approach is scalable, it is not suitable for reasoning about all patterns of inheritance.

A related tool that is concerned with visualizing phenotypes of patient cohorts is PhenoStacks by Glueck et al. [215]. PhenoStacks uses a tabular approach similar to what we use for our table.

5.4 Methodology

The core idea behind Lineage is to linearize the family tree and then juxtapose a table of attributes. With linearization, the space taken up by a given tree is greater than a nonlinear version, so we employ different modes of aggregation to collapse the tree in a meaningful way. In this section, we describe the steps taken to linearize the tree, followed by different aggregation methods employed to deal with the scalability issue.

5.4.1 Linearization Approach

The overall linearization approach consists of two steps, decycling and then assigning a linear order to the nodes in the tree.

5.4.1.1 Decycling

In the first step, we remove cycles from the directed network, transforming it into a tree, by duplicating the node that completes a cycle, similar to the approach of Mkinen et al. [200]. If the duplicated node has children, we attach all children to one instance, while the other instance remains a leaf. Figure 5.3(a) shows a tree-like network with one cycle, and Figure 5.3(b) shows the resulting tree, where node 7 is duplicated. Although this duplication strategy works for general directed networks, it is most useful for directed networks with a defined root and few cycles, since in these cases most of the topology is retained, and the number of additional nodes is negligible with respect to scalability.

5.4.1.2 Linearization

In most tree layouts [216], associating the nodes with rows in a table by position is impossible. The tree in Figure 5.3(b) is compact, yet would require, for example, curved links to associate the nodes with a table row. To make this association between nodes and rows

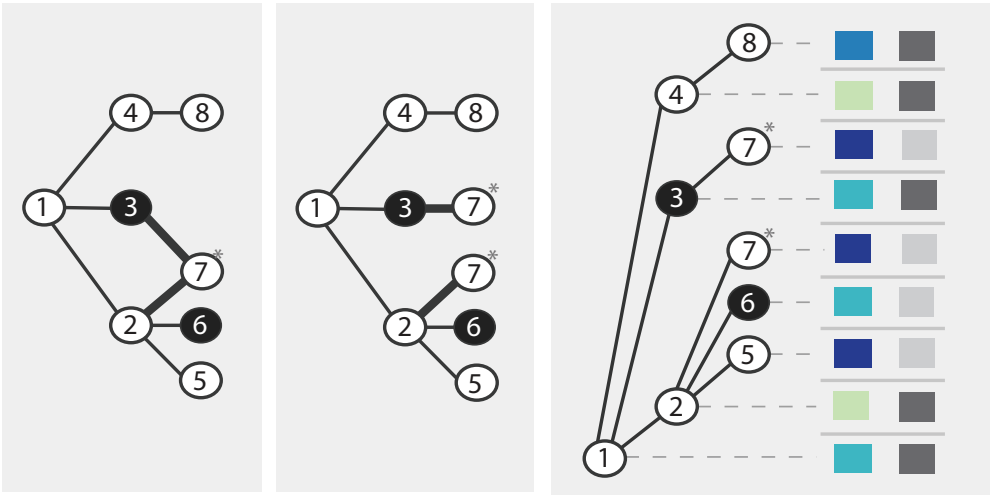


Figure 5.3. Decycling and linearization. (a) A directed, rooted network with one cycle ending in node 7. (b) We remove the cycle by duplicating the last node in the cycle (node 7). (c) The tree is linearized so that each node is assigned a distinct row. Leaves are rendered above their parents. This row-based, linear layout enables an unambiguous, position-based association with a table visualizing attributes.

of a table intuitive, we use a linearization strategy that assigns every node a distinct vertical position (i.e., a "row"). The position of the node alone thus unambiguously associates the node with a row in a table (see Figure 5.3(c)). Note that although we assume a left-to-right tree layout here, a top-to-bottom layout would work equally well for associating a tree with table columns.

Linearized tree layouts are based on tree traversal strategies. Although various strategies, such as breadth-first (level-order) or in-order depth-first-search, are possible, we found that a preorder depth-first-search works well for our purposes, since it results in a crossing-free layout and keeps leaves in subsequent rows.

Following the in-order strategy, we recursively place the descendants of a given node directly above their parents. Note that a top-down strategy would also be possible. We assume that an order of leaves can be defined, e.g., based on the attributes. If not, using a random order is possible.

Figure 5.3(c) illustrates the results of this algorithm when applied to the tree in Figure 5.3(b) and also shows how to easily associate a table with the tree. Note that the duplicate node also is duplicated in the associated table.

5.4.2 Aggregation

Although linearizing the tree allows for a direct, position-based association of the nodes and their attributes, the resulting layout uses more space than a compact layout. However, due to their hierarchical structure, trees are well suited for aggregation. Degree-of-interest (DOI) functions [183] have been widely applied to trees. In our design, we use the generalized idea of degree-of-interest functions by Furnas [183], [217].

We let analysts define a degree-of-interest function based on the attributes of the nodes, which we call the **phenotype of interest (POI)**. Nodes that have the POI are referred to as **nodes of interest**. In contrast to the original formulation of a degree of interest, our POI function is binary (i.e., a node is either of interest or not) and does not consider a distance to a selected node. An example for a POI is "committed suicide," which marks all nodes representing individuals who committed suicide to be of interest, or "has a maximum BMI of higher than 30," which would consider all obese individuals to be of interest. POIs that are a compound of multiple attributes (high BMI and suicide) are possible.

Based on this degree-of-interest function, we introduce two approaches to aggregation that vary in how they trade off compactness and preservation of the attributes of the nodes: 1) attribute-preserving aggregation; and 2) attribute-hiding aggregation. These aggregation approaches can, of course, be applied not only to the whole tree, but also to selected subtrees, or to both.

5.4.2.1 Attribute-Preserving Aggregation

Here we introduce an aggregation strategy for linearized layouts that preserves both the structure of the tree and the attributes of all the nodes. Nodes of interest are assigned a row of their own, whereas other nodes are aggregated into a single row. Figure 5.4(a) shows an example of this strategy applied to the tree shown in Figure 5.3(c). This layout emphasizes the nodes of interest, while preserving both the structure of the network and the attributes of the other nodes.

Our algorithm recursively follows a (sub)tree down a branch by assigning a new row to each inner branch. Inner branches are branches that do not end in a leaf after the first edge, i.e., an edge that directly connects to a leaf is not an inner branch. If no node of interest

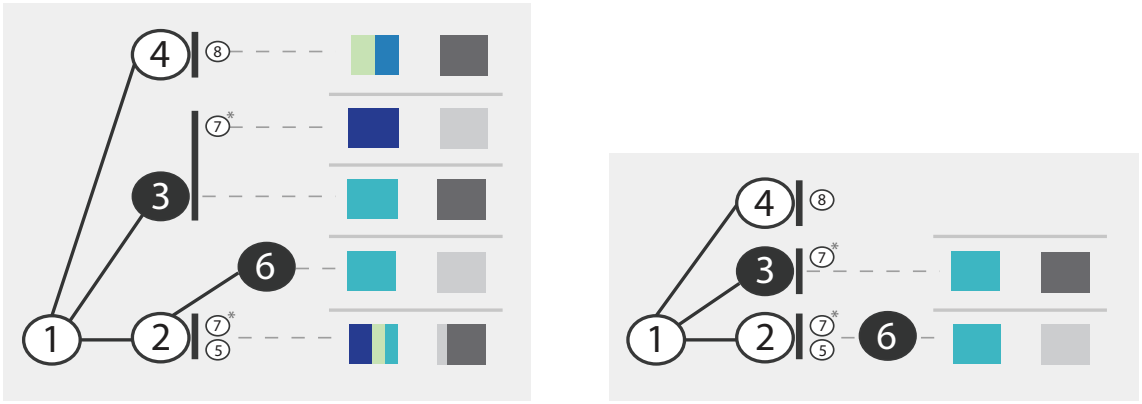


Figure 5.4. Aggregation approaches demonstrated using the tree in Figure 5.3(c). A filled-in circle indicates a node of interest. (a) Attribute-preserving aggregation. Each node of interest (shown in black) is in a separate row. Branches without nodes of interest are aggregated into one row, yet all attributes are preserved in the aggregate representations in the table. Notice how the two children of node 2 who are not affected are shown using an implicit encoding, which we refer to as a "kid grid." (b) Attribute-hiding aggregation. The branches leading to nodes of interest are hidden behind them. Only nodes of interest and branches with no nodes of interest have a row of their own. Only the nodes of interest are represented in the table.

is encountered, the algorithm continues to the leaves, placing all nodes of the branch in the same row. Multiple leaves that are not of interest are placed in a *kid grid*, an implicit encoding of the leaves as small nodes to the right of their parent. These nodes retain all visual encodings (e.g., shape for gender, crossed-out for deceased). We chose this approach for the representation of kid grids over alternative designs such as a numeric labels or bar charts since it is consistent with how individuals are represented in other places in the tree. An example is visible in the bottom branch in Figure 5.4(a), where nodes 1, 2, 5, and 7 are on the same row, and the leaves (5, 7) are in a kid grid. If a node of interest is encountered, we distinguish two cases. If the node of interest has children that are leaves and that are not nodes of interest themselves, they are added to a kid grid, which is placed in the next row (see node of interest 3,) and its descendant (node 7), which is placed in a kid grid in Figure 5.4(a)). If the node has children that are inner nodes, the algorithm is applied recursively. The result of this algorithm is a layout that has N rows, where N is the sum of:

- the number of nodes of interest,
- the number of inner branches that do not end in nodes of interest (case for node 4 in Figure 5.4(a)),
- the number of nodes of interest that have children that are leaves (case for the child of node 3 in Figure 5.4(a)).

The result in the associated table visualization is that each node of interest has a separate row, and the aggregated branches are represented in aggregated rows. In practice, we use visual encodings for aggregates and individual rows that can faithfully represent the data but are also comparable. For details on the table design, see Section 5.5.2.

5.4.2.2 Attribute-Hiding Aggregation

This form of aggregation also preserves the complete structure of the tree, but it does not preserve attributes of nodes that are not of interest. The result, illustrated in Figure 5.4(b), is a scalable approach that can be used to address tasks that are concerned only with the attributes of the nodes of interest and their connectivity, but not with the attributes of the other nodes.

The main difference compared to the attribute-preserving aggregation is that nodes of interest are not assigned to a new row when they are encountered. The algorithm again

recursively follows a (sub)tree down a branch by assigning a new row to each inner branch. If no nodes of interest are encountered while traversing the branch, the leaves are placed in a kid grid. If a node of interest is encountered, the next step depends on whether it has children that are inner nodes or not. For the node's children that are leaves, a kid grid is used, but no new row is started. For all other branches, the algorithm is applied recursively.

The resulting layout has M rows, where M is the sum of:

- the number of inner branches,
- the number of nodes of interest that have at least one child that is an inner node.

Here, only nodes of interest and inner branches that do not end in a node of interest are assigned their own row. For consistency, we do not represent branches that do not end in a node of interest in the table.

5.5 Lineage Design

Here we describe the design decisions that are specific to the use case of visualizing genealogies and that we realized in the Lineage prototype. To address the tasks of our collaborators, Lineage provides three views, shown in Figure 5.1: the genealogy network view, the closely synchronized table view, and a family selection view, which allows analysts to select one or multiple families.

5.5.1 Genealogy Network

An important difference between genealogical trees and general trees is that nodes have not one but two parents. To address this, we introduce the concept of a couple, indicated by a line connecting the partners (see Figure 5.5(a)). As is common in genealogical network layouts, the children of a relationship then connect to the line representing the couple instead of directly to the parents. We also adopt some of the conventions for drawing genealogical networks: males are drawn as rectangles, females as circles. Deceased individuals are crossed out. In Figure 5.6(b), for example, the topmost node represents a female who is alive and has the POI, but the other nodes with the POI are deceased. Nodes that have the phenotype of interest are filled in.

As discussed in the previous section, the phenotype of interest can be defined dynam-

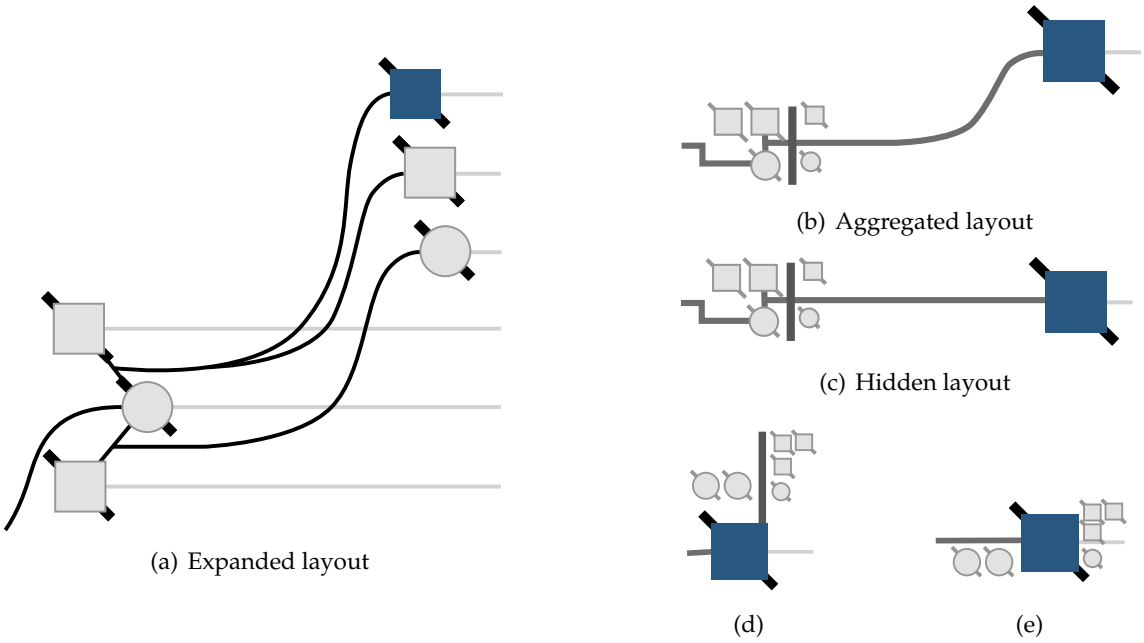
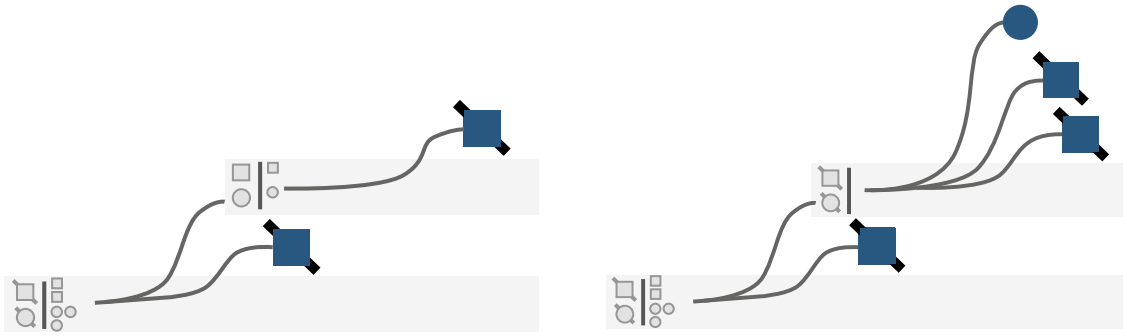


Figure 5.5. Different aggregation cases. (a-c) A family where one woman has children with two men. One of the children committed suicide. (a) No aggregation: every person is in his or her own row. (b) Attribute aggregation: the suicide case is in its own row; the rest of the family is aggregated. Notice the family grid with two male and one female parents, and one daughter and one son. The second son is not in the kid grid because he is a node of interest. (c) Attribute hiding: the family is hidden behind the suicide case. Only the attributes of the suicide case will be shown in the table. (d-e) A different family, where the node of interest has children, leading to special cases. (d) Attribute aggregation: the spouses and children are moved to their own row. The line connecting spouses spans two rows. (e) Attribute hiding: the spouses are placed to the left of the suicide case, the children to the right.

ically, based on either combining categorical values or brushing a range of a numerical variable. Figure 5.6 shows the effect of two different POI functions on the same subtree.

The modifications to the layout algorithm to accommodate couples are minor: couples are always placed in consecutive rows to avoid long, vertical parent edges. When one of the spouses has offspring with multiple partners, we place all partners in consecutive rows. In the case of two partners, we place the person with multiple relationships in the center to avoid edge crossings. Figure 5.5(a), for example, shows a woman who had children with two partners. For more than two spouses, however, or spouses who had children with different partners in alternating order, edge crossings are often unavoidable. Similar to Mkinen et al. [200], we use arrows to indicate that a node is duplicated and to point toward the duplicate. To resolve any ambiguities, we draw an edge connecting the duplicates



(a) POI: Suicide (b) POI: Age < 40

Figure 5.6. Different POI functions applied to the same aggregated subtree. (a) Suicide as a categorical POI. (b) Age < 40 as a numerical POI.

when hovering over the arrow (see Figure 5.7).

In contrast to traditional genealogical networks, we do not lay the nodes out by generation, but use the birth year to position the nodes horizontally [208], as shown in Figure 5.1. This approach avoids ambiguities about the birth order and encodes a vital attribute directly in the network. We also use curved splines instead of the traditional orthogonal edge routing, because continuous edges are easier to follow [218].

5.5.1.1 Aggregation Layouts

With respect to aggregation, the algorithm is extended only by first looking for spouses before descending into a subtree. If both are nodes of interest, each spouse is assigned his or her own row.

We previously introduced the concept of kid grids for aggregated nodes. Indicating hidden nodes using a glyph has been done before for network layouts, most notably by McGuffin and Balakrishnan [204], who use dots to indicate children in genealogical networks. Our layout for aggregated genealogies, however, goes beyond a basic indication of existing nodes as they encode both topological information and attributes. First, we extend the notion of a kid grid that encodes children to a family grid that encodes all members of a family. Figure 5.5 shows multiple examples. A family is separated by a vertical line into parents and children. This vertical line represents the line used to connect spouses in expanded mode. Parents are placed on the left of the line. In addition to the node shape, we also redundantly encode sex by position, placing the nodes representing

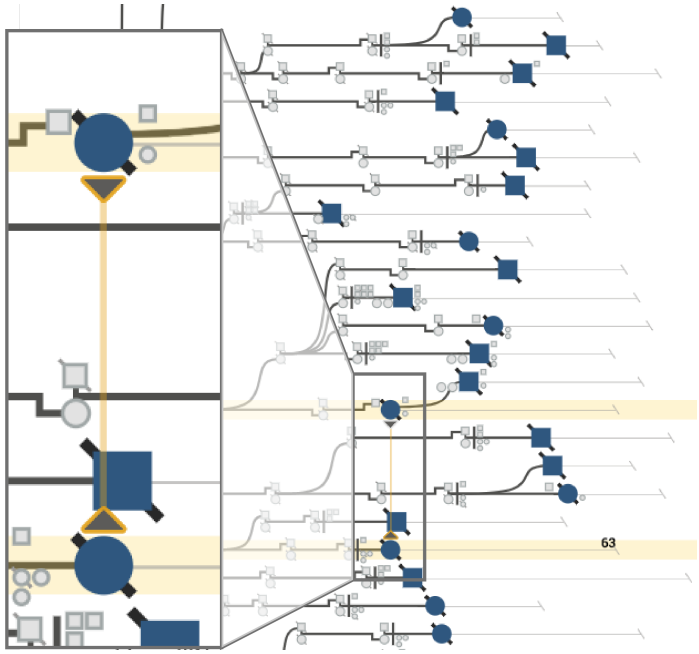


Figure 5.7. Visual encoding of nodes that were duplicated in the process of removing cycles from the network. The arrow glyph, which is shown at all times, indicates both the presence of a duplicate and its direction in the network. Hovering over the arrow draws a line connecting the node to its duplicate and highlights the corresponding rows in the table.

males on top and the nodes representing females below. In families with multiple partners, we place all partners in the same family grid, so that, for example, a family with a woman who has children with three partners is represented by three squares on top and one circle at the bottom.

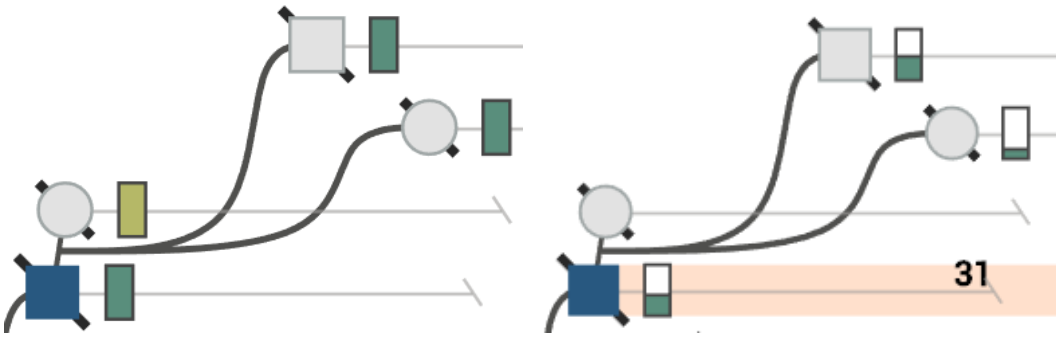
Note that aggregation results in some information loss. For families in which individuals have offspring with multiple partners, the exact association between children and parents is lost. Also, the attributes for all aggregated nodes in a row are displayed together in the table, removing the exact association between individuals and their attributes; instead, the distribution of values in that aggregate is emphasized. When hiding is used, the attributes are removed entirely from the table. We found that neither of these drawbacks is a problem since these design decisions align with the analysis tasks outlined earlier: analysts at first are often interested in nodes with the POI. When they want to consider other nodes in detail, they can deaggregate on demand.

It is important to note that we break with the convention of placing nodes based on their birth-year for aggregated families. Instead, we place the whole family based on the birth year of the parent with a blood relationships to the ancestors.

5.5.1.2 Encoding Attributes in the Network

Although we address the problem of encoding multiple attributes for nodes using our linearization approach, direct, on-node encoding of a small number of attributes provides the best bridge between attribute-based and topology-based tasks. We already discussed how sex (shape), deceased/alive (crossed out), birth year (horizontal position), and POI (fill) are encoded directly in the network. To enable our collaborators to view an additional variable in the network, we introduce a glyph, rendered to the right of the nodes, as shown in Figure 5.8. When the attribute is categorical, we color-code the glyph; for numerical attributes, we show a small bar. In both cases, the color coding is also used in the table to highlight the relationship (see the matching colors for bipolar disorder in Figures 5.8(a) and 5.9). When data are not available for a node, no glyph is shown.

Finally, we also encode the age of individuals by drawing a line from the node, which is placed at the year of birth, to the year of death, or to the current year (see Figure 5.8). These age lines conveniently encode an important variable in the existing coordinate system. We found that the age lines also help to perceptually connect the nodes to the rows in the table. Since we do not draw age lines for aggregates, we found it necessary to indicate the connection to the table using a light-gray background.



(a) Categorical attribute (b) Numerical attribute

Figure 5.8. Attribute encodings in a family tree. Age lines visualize the lifespan of individuals. Age lines for people who are alive continue until the present. Age lines of deceased individuals are terminated at their year of death. We can see that the individual represented by the node of interest died at age 31, and his spouse died shortly thereafter. Selected attributes can be visualized next to the nodes in glyphs (green rectangles). (a) The categorical variable bipolar disorder is encoded by a dark-green color. (b) The numerical variable number of bipolar diagnoses is encoded as a bar chart.

5.5.2 Table Visualization

The attribute table is designed to visualize both rows representing individuals and aggregates representing multiple individuals in the same space. The attributes visualized in the table can be chosen using the *Table Attributes* menu in the tool bar. As shown in Figure 5.9, we use dot plots to encode numerical data. Combined with transparency and jitter, dot plots can also be used to encode aggregate rows. For categorical values, we distinguish between binary categories, such as deceased or alive, and multivalued categories, such as race. We encode binary categories in a single column, as can be seen for "sex (F)", where a dark cell corresponds to true and a light cell corresponds to false. For multivalued categories, we use a method commonly employed by Bertin [20], and use one binary column for each category instead of, e.g., using color to encode categories. We represent aggregates of binary or categorical values as stacked bars, which are scaled according to the number of individuals in a category. Text labels and IDs do not have adequate visual representations for groups of elements, so we display an ellipsis (...) for aggregates. Missing values are rendered as a dash to distinguish them from zero or false values. We also provide a column that shows how many people are in a given row.

We avoid color to encode data, so we can employ it to highlight elements of interest,

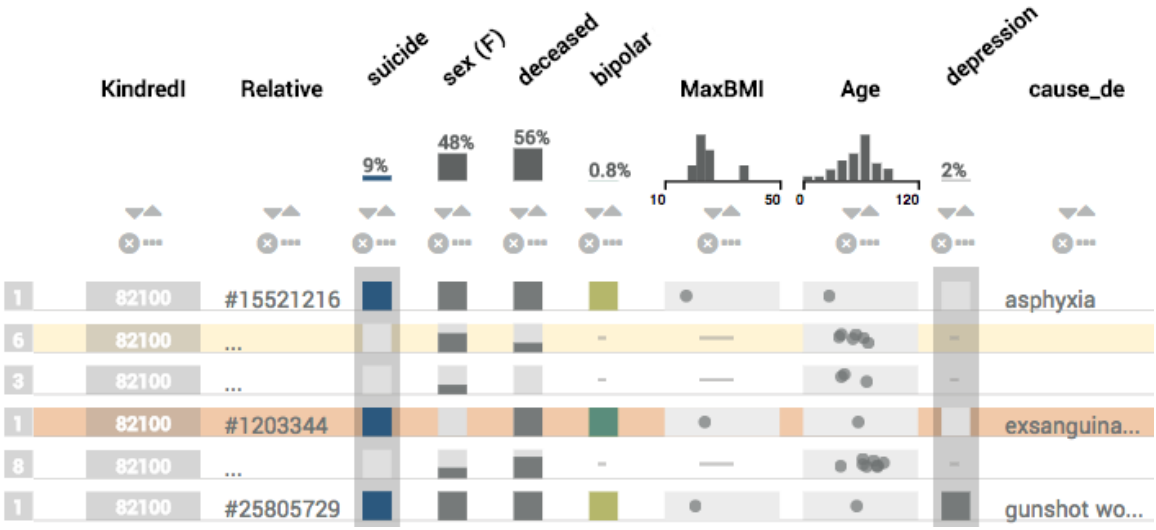


Figure 5.9. The table view in Lineage. The first column encodes how many individuals are aggregated in that row. Binary categories are represented as present/absent (e.g., sex). Aggregates of binary variables show the proportions of the variable in stacked bars. Numerical values are encoded using dot plots, which are also used for aggregates. The POI is highlighted using a gray background. The depression column is starred, also indicated by the gray background.

such as to highlight selected rows and to indicate the column that encodes the user-selected phenotype of interest and the primary attribute. In Figure 5.9, the selected attribute (bipolar) and the POI (suicide) are rendered in color. A menu in the columns allows analysts to set the POI, set an attribute as a primary attribute, and *star* an attribute. Starring an attribute adds it to the family selector table.

These features, in combination with the network, allow analysts to address the tasks related to analyzing individuals (**T2**) and comparing cases (**T3**).

Finally, we also allow analysts to sort the table based on any column, which enables them to to easily identify clusters of similar items (**T4**). However, sorting by attribute removes the close association with the network. To partially remedy this problem, we draw slope charts, similar to what is used in LineUp [219], to relate the rows of the table to the rows of the network. These connection lines work well for a small number of rows, but often result in significant crossings when dealing with many rows. In that case, interactive highlighting helps to trace the lines. Lines that end in an off-screen location are not rendered. Instead, an icon indicates the direction of their corresponding row. Clicking on the icons automatically scrolls to the location of the corresponding row in the table. Figure 5.10 shows an example of a partially aggregated network sorted by suicide.

5.6 Evaluation — Case Studies

We present two forms of validation for Lineage: case studies, a method employed widely to demonstrate the fitness for use of visualization design studies [195], [220], and informal usability testing and analyst feedback, which is described in the next section.

The case studies outlined below were conducted with Dr. Hilary Coon, a psychiatry researcher and principal investigator studying the genetic and environmental factors in suicide. Dr. Coon, who is a coauthor of the paper this chapter is based on, also participated in analyst feedback sessions and contributed to the design and development of Lineage. She was familiar with the interface from the earlier analyst feedback sessions as well as through demonstrations during the design phase. For the case study, we deployed Lineage on a secure, password protected and HIPPA compliant server instance on Amazon Cloud Services, which allowed her to access the tool from her personal computer, in her own work environment, and at a time that was convenient.

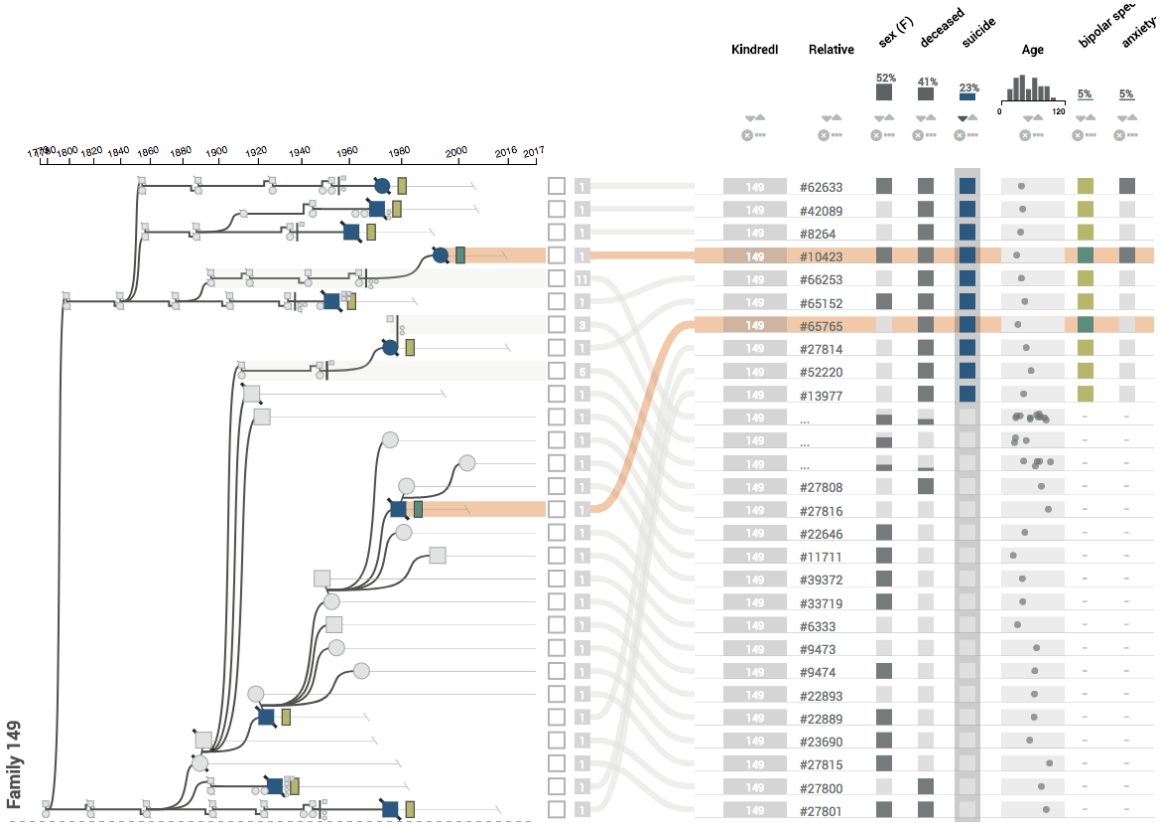


Figure 5.10. Connection between sorted table and network view. The table is sorted by suicide, which causes the rows in the table to be in a different order than the rows in the network view. The association between the two is retained by the curves connecting them.

For the case studies, Dr. Coon used the full dataset described in Section 5.1 and performed analysis tasks representative of her research. She documented her analysis process with notes and screenshots, which she shared with the team. Sections 5.6.2 and 5.6.3 were written by Dr. Coon and reflect her detailed accounts of the analysis workflow. These case studies were subsequently used in a research grant proposal.

These case studies demonstrate how Lineage allows our collaborators to complete several tasks, including: 1) selecting families most likely to point to genetic factors of suicide, and determining additional defining characteristics of cases in these families for familial analysis (**T1**); 2) visualizing aspects of familial cases contributing to genomic shared regions with genome-wide significant evidence (**T3**); 3) searching for supporting evidence by visualizing additional families with evidence of familial sharing in the same genomic region (**T5**); and 4) prioritizing new families for additional analysis based on visualization of risk and clustering of defining characteristics (**T4**).

5.6.1 Prioritizing Families for Analysis

Dr. Coon uses the suicide dataset described in Section 5.1 for her analysis. Using established familial relative risk methods [221], she ascertains a large number (>200) of extended multigeneration high-risk families for analysis [222]. Limited time for analysis and resources requires her to prioritize these families based on data visualized in Lineage. Our collaborator's goal is to select a promising family for the computationally expensive analysis of shared genomic segments (SGS) [223]. SGS investigates the significance of genomic segments shared between distantly related affected cases. If an observed shared segment of the genome is significantly longer than expected by chance, then inherited sharing is implied. A greater distance separating cases (i.e., path-distance in the genealogy graph) translates to the increased statistical power of the method: chance inherited sharing in distant relatives is improbable. Finding cases with shared sequences that also have a shared phenotype (suicide, plus potentially other comorbidities) allows Dr. Coon to reason that the sequence is a factor in these phenotypes.

Previous analyses exploring the statistical power of SGS using simulated high-risk families indicated that power is determined by familial distance (path length) between cases with genomic data for analysis, indexed by counting the total number of generations separating the cases (meioses). In this study, if families had at least 15 meioses between cases, then 3-10 families were sufficient to gain excellent power (>80%) to see at least one true positive within any given family [224]. For all scenarios considered, genome-wide association studies (i.e., studies that do not consider familiarity) would have negligible power to detect the simulated variants. This study therefore dictated that extended high-risk families should be selected with the highest familial relative risk, and with the largest number of cases with genotyping available for analysis through interrogating numbers of cases with DNA, and familial distance of these cases from one another. Familial distance is visualized in the graph view in Lineage and serves as a useful predictor of the statistical significance of shared regions among cases. However, additional data about the cases shown in the table view of Lineage are useful in later prioritization of genes in regions with significant evidence of sharing. These attributes include gender, young age at death, and clustering of comorbidities.

Figure 5.11 shows two families of high interest (709, 42623), based on significant fa-

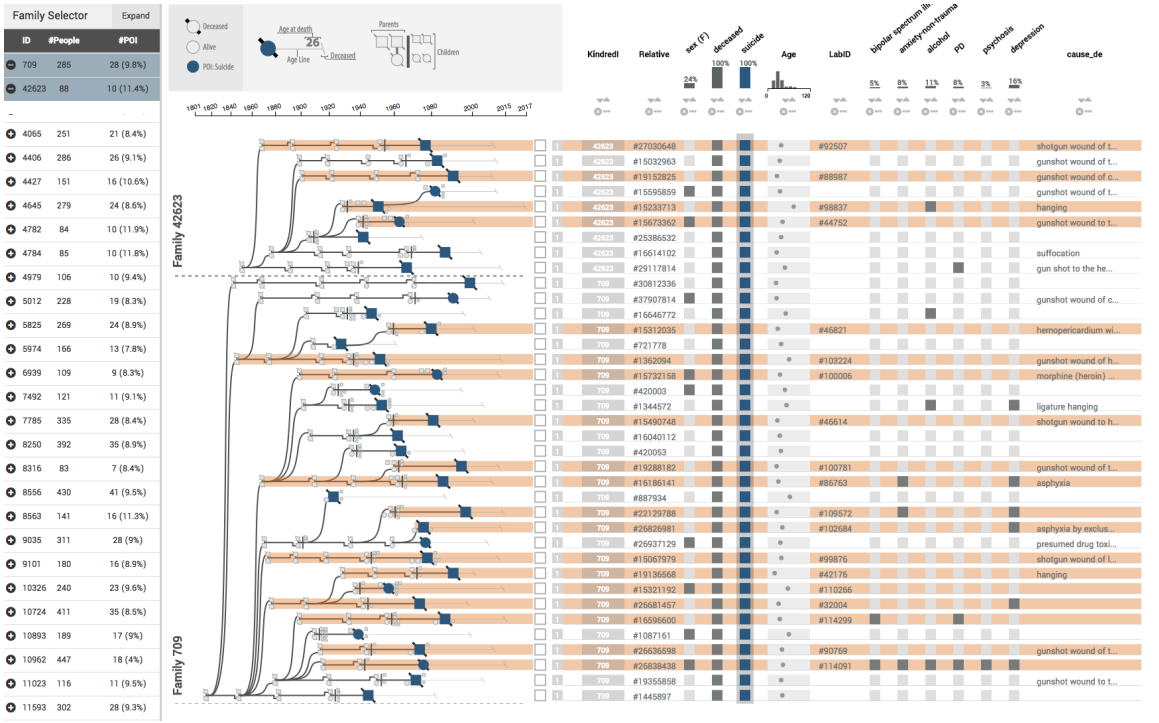


Figure 5.11. Two families with high familial risk for suicide. Our collaborator uses Lineage to prioritize families for an analysis of shared genomic sequences of suicide cases. Although both families seem promising based on risk alone, Lineage reveals that the few cases with genetic material (indicated by the presence of a *LabID*, highlighted in orange) in Family 42623 are too closely related. Family 709, in contrast, contains 15 cases with genetic material available that are also widely separated. Furthermore, a large cluster of relevant comorbidities (depression, anxiety, alcohol abuse, bipolar disorder, personality disorder) indicates a likely genetic component of suicide in this family.

milial risk ratios ($p<0.0001$).These familial risk ratios are calculated with a separate tool. Significant families are selected by ID in Lineage. Both families have more than three cases available for analysis, as indicated by the *LabID* attribute. However, 42634 was not chosen for analysis given that cases are not sufficiently separated (of the four cases with a LabID, two have a parent-child relationship, for instance), and little co-occurring diagnostic information is apparent in this family. Lineage reveals that family 709, in contrast, suggests clustering of multiple conditions, and has 15 suicide cases with genetic material available that are also adequately separated (**T4**).

Our collaborator hence selected family 709 for the computationally intensive analysis of shared genomic segments. Creation of files for this analysis was facilitated by exporting the LabIDs from the family, which serve as the input "proband list."

5.6.2 Identifying Comorbidities of Cases Contributing to Significant Familial Genomic Sharing

Analyses of selected families to date have produced significant evidence of genomic sharing. An example is a shared region in family 601627 with genome-wide significance (p=1.94E-10). A single gene in this region, *neurexin 1* (*NRXN1*), has evidence from the published literature of involvement in psychiatric risk [225] and inconclusive evidence for risk of suicide [226]. The shared region in family 601627 is shared by six cases, shown in Figure S4 in the supplemental material of Nobre et al. [1]. Dr. Coon used Lineage to interactively explore demographic attributes and clustering of clinical co-occurring conditions (**T3**). Cases contributing to sharing had young age at death, ranging from age 17 to 39; average age at death in the research cohort is 40. The clustering of co-occurring depression is not completely unexpected, as approximately half of the suicide cases show evidence of depression. However, the multiple cases with personality disorders (PD) are more unexpected. These disorders include less commonly used diagnoses such as antisocial personality, borderline personality, conduct disorder, and obsessive personality.

This association was previously unknown to Dr. Coon, and would have been very difficult to identify given the number of cases in the family (831) and the number of case attributes. Knowledge of clustering of attributes in these cases with evidence for this particular genetic risk will direct our collaborator in her selection of additional families for resource-intensive replication studies. The attributes will also guide selection of other nonfamilial cases from their much larger cohort and of external cohorts for further replication. If replication is achieved, knowledge of case attributes could also drive the design of additional targeted studies in specific high-risk subgroups.

Based on these discoveries, our collaborator then extended her search for other families associated with shared regions that are associated with the gene *NRXN1* (for details, see Section 2 in the supplementary material of Nobre et al. [1]). She found two more families associated with the gene *NRXN1* and consistent demographic attributes and comorbidities (Figure S5 in the supplement of Nobre et al. [1]). Across all three families, the occurrence of PD in the cases supporting the genomic regions was 8/19=42%. Once the three families had been visualized, another feature was observed (**T5**). Of the 19 cases across all families, 6 were female (31.6%). The percentage of female cases in the overall cohort was 20.8%. The

occurrence of this trend for this important demographic attribute will help with selection of additional families to continue to follow up evidence for this and related genes.

5.6.3 Prioritizing Additional Families

Our collaborator has found Lineage to be particularly helpful in prioritizing additional families to follow up initial results based on clustering of important attributes. For the case study above, the important attributes were young age at death, presence of PD, enrichment for female cases, and possibly presence of depression. Within these selection attributes, families had to also meet minimum requirements of number and separation of cases with available data, as discussed in the first case study (**T1**).

Our collaborator started by sorting the families by the occurrence rate of personality disorders (PD) within Lineage, which resulted in relatively few families with a concentration of cases with these disorders. She looked not only at overall percentage of suicide cases in the families with PD, but also at numbers of cases with DNA for analysis; even if a family has overall high enrichment of this diagnosis, if only one or two cases have genetic data, the family will be of little interest. Given this initial filter, three families were prioritized at the top (**T1**): families 540781 (7 PD cases total, 3 with DNA), 10724 (6 PD cases total, 4 with DNA), and 565350 (5 PD cases total, 3 with DNA). Figure S6 in the supplement of Nobre et al. [1] shows the attribute table for these families. As Dr. Coon was performing this initial task, she noticed that a high number of the PD suicide cases were also women, even in families not at the top of this follow-up priority list (**T4**). Although the association between these attributes was present in the family described in the previous case study (6 of the 19 PD cases supporting sharing in the initial three families were female), this signal was even stronger in these families identified based on the phenotype characterized. In families 540781, 10724, and 565350, the proportions of female PD cases were 4/7, 2/6, and 5/5, respectively. Taking all families together, the proportion of female cases among those with PD was 11/18 = 61%. Given that female suicide makes up only about 20% of cases overall, this observed pattern is striking. Our collaborator is now interested specifically in this association as it may relate to *NRXN1* genetic risk, but also more generally in the association of these attributes and how this may relate to other phenotypic and genetic aspects of the research sample.

It became apparent to her that the initial prioritization already fulfilled another criterion, enrichment for female suicide, likely due to the association between this characteristic and the occurrence of PD. Note that this enrichment for female suicide is apparent only when also considering the PD attribute; taken as a whole, these families are not significantly enriched for female suicide.

Another attribute of interest was age at death. Taking all age at death for cases with DNA in families 540781, 10724, and 565350, Dr. Coon found averages of 32.86 (sd=12.48), 43.42 (sd=13.98), and 26.33 (sd=9.06), respectively (**T3**). This factor suggests that families 565350 and 540781 may be most interesting for computationally intensive follow-up of the initial findings, and will help her and her team make decisions regarding which cases to select for expensive molecular sequencing.

Overall, the case study revealed several ways in which Lineage aided in the analysis process, as detailed above, as well as a few limitations of the tool. One such drawback is the need to identify families of interest prior to using Lineage in order to select relevant families for comparison within the tool. Another limitation is the need to manually count selected individuals in order to calculate the percentage of a certain family represented by a selection. These limitations are addressed in further detail in Section 5.8.

5.7 Discussion

Although details of our design study and our implementation, such as how we display parents and family grids, are specific to genealogies, we argue that our linearization and attribute-driven aggregation approach can be applied broadly when analyzing multivariate trees or tree-like graphs, such as phylogenies or file directories. The species and their relationships depicted in phylogenies, for example, are associated with vast numbers of attributes capturing traits (is flightless, has tail, color, etc.), and judging which attribute is inherited at which point in the tree of life is crucial for understanding the process of evolution. Answering these questions is important in a basic science context as well as in a human health context. An example for the latter is the study of the development of viruses such as influenza, Zika, or HIV [227]. Our approach also has the potential to be combined with more generic graph-to tree extraction approaches, as discussed by Lee et al. [89]. Using their method, arbitrary multivariate graphs could be converted into trees

and explored in Lineage.

Lineage as a clinical genealogy visualization tool specifically can be applied to study other diseases with a major impact on human health. We have already deployed Lineage with an autism dataset (see Figure S3 in the supplementary material of Nobre et al. [1]), which has characteristics that emphasize the importance of attribute visualization of non-leaf nodes. In this dataset, attributes are also available for parents and other relatives of autism cases.

We also argue that our strategy of combining explicit node-link layouts with the implicit layout of the family grids is transferable to other application scenarios.

Our described linearization approach makes the association between nodes and attributes obvious and enables a tight integration of attribute-based and topology-based graph analysis tasks. Both aggregation methods described serve to reduce the space usage of the linearized tree while preserving the topology and the desired level of information about the attributes. The aggregation is based on two principles: assigning nodes to be aggregated to the same row and combining the explicit node-link layout with the implicit encoding for aggregated nodes and their leaves (family grids).

The Lineage genealogy visualization tool can be broadly used with other genealogical datasets, e.g., to study autism, diabetes, or cancer. Many groups at the University of Utah make use of the Utah Population Database, and we have already established contact with other potential collaborators who are in need of a clinical genealogy visualization tool. Some of these datasets also have detailed attributes for nonaffected cases, which will make our attribute-preserving aggregation approach more valuable. Although our data are unique with respect to scope, detailed genealogical datasets are becoming more common because they have shown immense potential for population genetics [228]. We believe that our approach could also be adapted to datasets containing many small families (siblings, parents, grandparents of affected individuals) since they are commonly collected to study the genetic disease of one family member.

5.7.1 Scalability

In contrast to other tools, such as the DOITree [184], our aggregation approach preserves all the structure of the tree, which is suitable for trees with hundreds of nodes,

but not for trees with tens of thousands of nodes or more. To scale to larger trees, our algorithms could be combined with hiding parts of the tree. Also, although our algorithms work for any tree and any phenotype of interest, they are most efficient if the number of nodes of interest is small compared to the number of nodes in total. A common phenotype of interest for our collaborators is suicide, and the typical genealogies they study contain between 5-15% suicide cases. For these conditions, we found the resulting layouts to be compact and useful.

We found Lineage to scale well to families with about 1500 individuals, which covers most families in our collaborators' dataset (547 of 550 families have fewer than 1000 individuals). We also experimented with the largest families in our dataset, which contain about 2500 individuals. For these families, we observed several seconds of wait time until the decycling and the layout were computed. We anticipate addressing these performance limitations through precomputing and caching initial layouts.

In terms of the scalability of the visual encodings, we argue that Lineage produces a more readable layout in less space than Progeny, the tool that is currently used by our collaborators for displaying genealogies. Note that Progeny has only very limited capabilities for showing attributes by encoding attributes directly on the nodes and displaying text underneath nodes, and attributes cannot be dynamically selected or manipulated. For a comparison between Progeny and Lineage, please refer to the supplementary document of Nobre et al. [1]. When using suicide as a POI (the most common use case) and when using attribute-hiding aggregation, a family with about 400 individuals fits onto a single screen without scrolling. Larger families, attribute-preserving aggregation, or no aggregation more commonly require scrolling.

The number of attributes that can be displayed for each individual is limited by the horizontal screen size. On a large, 2560x1600 pixel display, about 20-40 dimensions can be shown, depending on the type (text and numerical columns need more space than binary categorical, for example). We found that this number typically exceeds the number of attributes our collaborators would like to study simultaneously.

5.8 Conclusion

In this chapter, we introduced a novel approach for visualizing multivariate trees and tree-like graphs using a linearization approach. We demonstrate the usefulness of our approach by realizing it in the Lineage system, which is designed for the visualization of genealogies in a clinical context. Using Lineage, our collaborators are now able to efficiently explore the structure of large families and even multiple families at the same time, in addition to analyzing dozens of attributes for the individuals in these families. They can use Lineage to identify phenotypes of interest that appear in multiple families, and then use this knowledge to inform and narrow down their search for genetic variants.

Lineage in its current form is already useful to our collaborators, but there are many directions in which it could be extended. Specifically, we currently deal with only a selected subset of the 3000 dimensions that are available for each of our cases. We plan to develop integrated visual and analytical methods to select dimensions of interest for any given subset of patients. For example, the system could identify that for a given family, PTSD is a common comorbidity and suggest that the analyst add PTSD to the table. Such an approach will be especially important when we start to integrate the detailed genetic data that are available for many of these cases.

The case studies and the feedback session also revealed areas for future work. First, it is desirable to integrate search and filter functionality, so that analysts can quickly identify families of interest based on attribute data. In our current implementation, participants used an external spreadsheet with statistical information about the families, and combined it with the browser search feature to find families of interest. A second aspect is a panel that displays basic information about a selection, such as the number of selected individuals, as well as the percentage of the total family selected. In our case study, these numbers were achieved by manual counting.

CHAPTER 6

MVN VISUALIZATION TECHNIQUE — JUNIPER: A TREE+TABLE APPROACH TO MULTIVARIATE NETWORK VISUALIZATION

Network visualization is a challenging problem, especially when the size of the network exceeds a few hundred nodes. This lack of scalability is exacerbated when rich attributes for the nodes and/or the links need to be considered when analyzing a network. Such multivariate networks are common across domains: biologists, for example, need to explore canonical pathways in the context of experimental data, to judge whether a pathway is valid for a given tissue or organism; social scientists may need to study whether a tight group of friends are all in the same age group and went to the same school. The difficulty of visualizing multivariate networks arises from two conflicting goals that need to be reconciled: visualizing topology and visualizing node and edge attributes. The visualization community has a good understanding of how to visualize either the topology of a network or the multidimensional data that are associated with the nodes and edges, yet addressing both topology-based tasks and attribute-based tasks at the same time is still an open research problem. Although there has been progress on visualizing aggregate attributes for the larger structure of a network [17] or on visualizing attributes for special network structures such as trees [1] or paths [58], [62], we are not aware of a scalable, multivariate network visualization technique that excels at supporting *focus tasks*. Here, we introduce such a technique.

We use the term focus tasks to refer to tasks where the details of individual nodes, edges, and their neighborhood matter, as opposed to the global structure of the network. Focus tasks commonly require readable labels and a detailed understanding of a node's attributes. These tasks include identifying adjacent nodes (who are my friends?), iden-

tifying nodes that are accessible from another node (where can I fly to from this airport within at most one layover?), finding short paths (what's the best route to go from A to B?), etc. Examples for focus network tasks on multivariate networks include investigating congestion and latency in a computer network or exploring how a mutated gene influences activity levels of the genes in its neighborhood. It is worth noting that these focus network tasks are equally important in both large and small networks.

Our primary contribution is Juniper, a new interactive technique that is tailored to address focus tasks when visualizing large, multivariate networks (Figure 6.1). The core idea is to extract a spanning tree from a subnetwork that is the result of a query of a larger network. The spanning tree is grown from a node of interest and laid out in a linearized tree, where every node can be unambiguously associated with a row in a table. This table is used to visualize topological properties of the tree, such as the degree of the nodes and their adjacency to selected other nodes, and to show rich attributes.

We also contribute an implementation of this technique, which enriches this basic concept with user interactions to restructure the tree to best answer the analyst's question, expose additional topological information such as edges not included in the tree, identify shortest paths between nodes, explore interdependent attributes along paths in the network, aggregate groups of nodes to save space, expand the network on demand, filter nodes by type, or sort them based on attributes.

Juniper is tailored to address focus tasks related to the details of a large network. We argue that this class of tasks is important in many practical applications and complementary to overview tasks that are better addressed with other techniques.

6.1 Related Work

Juniper is inspired by and contributes to multiple subfields of network visualization. Here we discuss how our work relates to tree-based network visualization and to query-based visualization of large networks.

6.1.1 Tree-Based Network Visualization

The idea of tree-based network drawing goes back at least two decades. Munzner uses a spanning tree as the structure to lay out a network in hyperbolic space [229] and shows

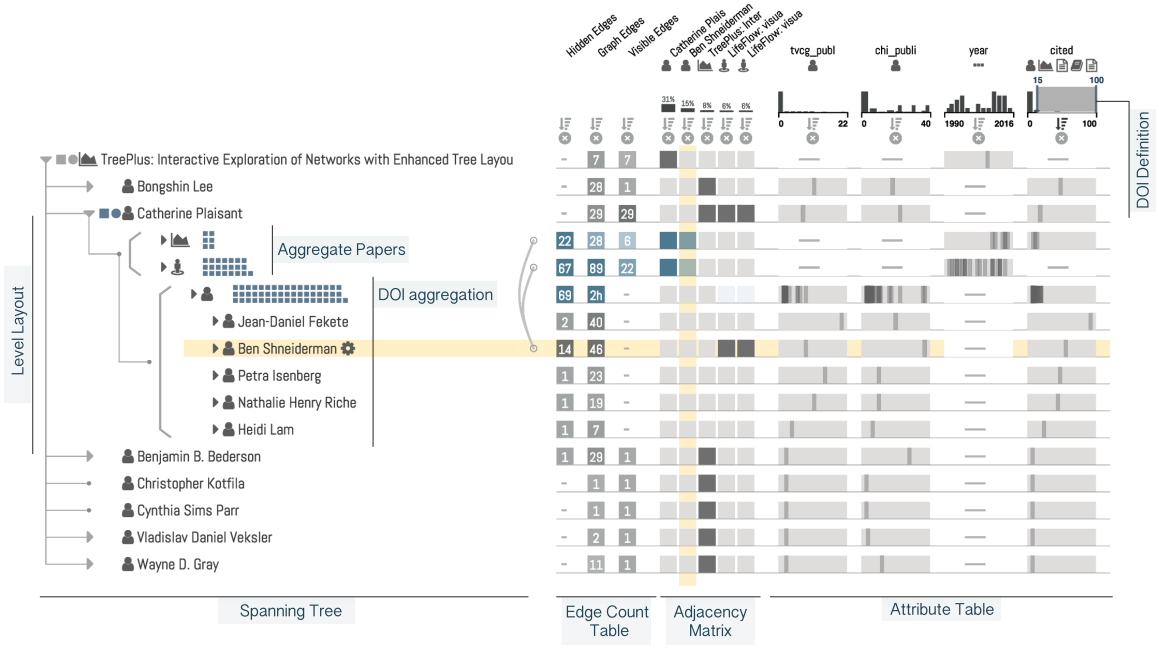


Figure 6.1. Juniper visualizing a coauthor network starting at the TreePlus paper as a spanning tree. The network is extended for Catherine Plaisant to include all her papers and coauthors. The papers are shown in aggregate form and faceted by CHI and TVCG. Most of the tree uses a conventional layout, but the descendants of Catherine Plaisant's node are shown in level layout, which groups nodes by distance to the branch root. Nodes in this branch are aggregated, with the exception of prolific authors, which are revealed using a degree-of-interest function. Ben Shneiderman is highlighted; two hidden edges originate at his node. The edge-count table shows a summary of the connectivity of each node. The adjacency matrix shows explicit connections to selected, highly connected nodes. The attribute table shows attributes about the authors and papers for individual as well as aggregated rows.

links that are not part of the tree on demand. Hao et al. [230] take a similar approach, but they also introduce duplicates to resolve some ambiguities. Similarly, Ontorama [231] uses a hyperbolic layout for a spanning tree and supplements it with a second view showing a linear tree that allows duplicate nodes.

Yee et al. [232] introduce a radial layout for networks based on spanning trees. A focus node is used as the root of a spanning tree and shown at the center, immediate neighbors are shown circling the focus nodes, neighbors once removed are shown on a second circle, etc. The edges of the spanning tree and other nontree edges are shown in a different color. Animated transitions are used to dynamically update the focus node. MoireGraphs [18] follow the same principle but combine the radial layout with rich on-node attribute visualizations.

The works most closely related to ours are TreePlus by Lee et al. [89] and the application-specific variant of TreePlus, GOTreePlus [233]. TreePlus introduces the "plant a seed and watch it grow" principle. Based on an initial, user-chosen node, analysts can grow the spanning tree by successively revealing subtrees. TreePlus shows hidden links between the tree nodes on demand using a combination of highlighting, a separate view of neighboring nodes, and explicit cross-links. Lee et al. evaluated TreePlus by comparing it to a traditional node-link diagram in a controlled study and found that TreePlus outperforms the node-link layout for most tasks and is preferred by most participants. For a detailed discussion of the differences of TreePlus and Juniper, refer to Section 6.5. Most of these techniques, including Munzner's hyperbolic tree, the radial layouts, and TreePlus, also encode node attributes, but they limit attribute visualization to on-node encoding of one or few attributes.

Another type of technique visualizes compound networks that have both a tree and a secondary network structure. Fekete et al. [177], for example, visualize a tree structure in a compound network as a tree map and render cross-links between the tree nodes on top of it. Holten [135] uses a compound network as an example for his hierarchical edge bundling technique. Gou and Zhang [234] render a tree structure in a sunburst layout and supplement edges connecting different levels of the layout.

Although Juniper builds on this rich body of prior work, it is unique with regard to several aspects. Juniper leverages novel interactions and the close integration of tree-based network visualization with an adjacency matrix to better support topology-based tasks in tree-based layouts. However, the main distinction of Juniper is the integration of an attribute table to support attribute-based tasks. The tree-based network visualization techniques discussed here are limited to one or two attributes, in contrast to Juniper, which is the first tree-based network visualization technique designed to handle highly multivariate networks.

6.1.2 Query-Based Visualization of Large Networks

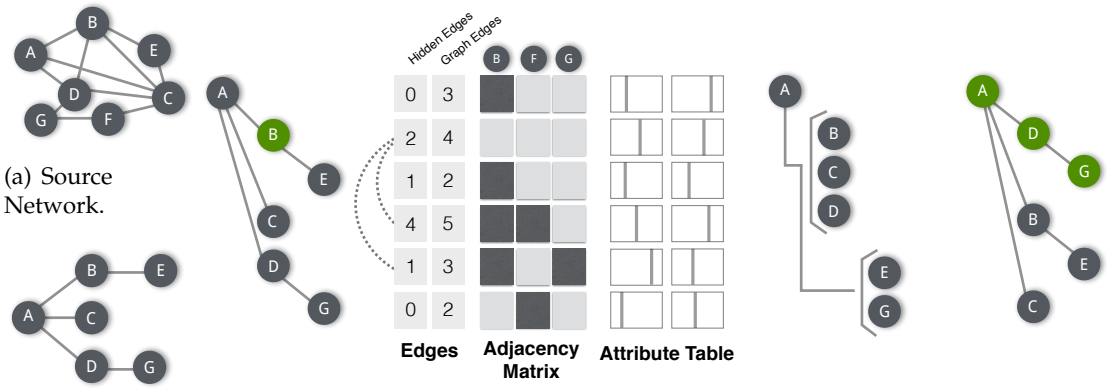
A common strategy to explore large networks is a bottom-up approach, where the analysis begins with a search or a query, and then more context is added as needed [5], [102]. Flavors of this approach range from explicitly revealing neighborhoods of nodes [13], [89],

to querying for paths or connectivity in a network [27],[62], to querying based on a degree-of-interest function [102], to associative browsing and complex queries [101], [235]. All these examples are designed to return or expand a single subnetwork, in contrast to techniques such as VIGOR [167] that are used to analyze (typically structural) queries that return many different subnetworks. Although we do not contribute novel concepts to network querying methods, we make use of many of these approaches.

6.2 Methodology

In this section we introduce the concept of tree-based exploration of multivariate networks. Details on our implementation of this concept and a number of design decisions can be found in Section 6.3.

The idea that we follow is to: 1) extract a subnetwork from a larger, underlying network; 2) calculate a spanning tree from the subnetwork; and 3) linearize this tree. The linearization enables us to juxtapose the tree with a table, as illustrated in Figure 6.2. This tree+table approach, in turn, allows us to visualize additional topological information, such as node adjacency, and to show associated attributes of the nodes. Although the first two steps are common in other systems, as discussed in Chapter 2, Juniper is the first technique to make use of a dynamically extracted tree to visualize multivariate attributes.



(a) Source Network.

(b) Spanning Tree.

(c) Linearized Tree Layout with Tables.

(d) Level Layout.

(e) Path-Based Sorting.

Figure 6.2. From (a) a network, to (b) a spanning tree of a subnetwork. Note that node F is not included and that several edges are missing (e.g., B-D). (c) Linearization of the tree shown in (b). The linear tree layout allows us to juxtapose a table showing hidden edges and overall node degree, an adjacency matrix, and a table showing rich node attributes. Hidden links are shown for the selected node B. (d) Level layout of the same tree, where all nodes at the same distance from the root are grouped together. (e) Node sorting to ensure that all nodes on path A-D-G are in sequence.

Figure 6.2(a) shows an example network. In practice, this network can be larger than can be conveniently displayed, can have different types of nodes, and can have rich attributes associated with it. Following the "search, show context, expand on demand" principle [102], we extract a subnetwork from the larger network — either in bulk or iteratively — and calculate a spanning tree for that subnetwork using a breadth-first search (Figure 6.2(b)). If a subnetwork is added in bulk, a key decision in this process is the choice of the root node, since the tree-based approach works best for tasks related to the root (e.g., it is trivial to see all neighbors of the root). We assume that analysts will want to manually specify a root in most cases; if no root is specified, we choose the node with the highest degree. The order in which nodes are visited at a given level by the breadth-first search algorithm also has an impact on the resulting tree, as nodes visited first will likely have more of their neighbors available to be attached. In Juniper, the order is driven by a user-defined sorting function; sensible options include lexical ordering of node labels, ordering by degree, or ordering by attributes.

6.2.1 Layout

Once a spanning tree is calculated, we linearize the tree using one of two complementary layout algorithms. We produce a traditional **tree layout** using a depth-first search algorithm, where every node is assigned a unique vertical position (see Figure 6.2(c)). The order of nodes for layout purposes is again defined with a sorting function.

An alternative layout is the **level layout**, shown in Figure 6.2(d). In the level layout, all nodes of a level are shown next to each other, followed by all nodes of the next level, etc. Again, sorting of nodes is driven by a user-specified function.

Level layout and tree layout have complementary strengths. The tree layout is well suited to investigate precise relationships to the root node. For example, in the bipartite coauthor network, if we start with an author, we can expand all her publications, and then expand all the coauthors on each of these publications, giving us a sense of who collaborated on which paper. The level layout, in contrast, allows us to ask a different question. In the level layout, the root author would be at level one, all her papers at level two, and all her coauthors at level three. In this layout, we can easily see and compare all the coauthors of the root author; they will be next to each other, and we can use the table

to sort the nodes, to identify, for example, the author with the most papers. In general, the tree layout can be used to answer questions about specific topology, whereas the level layout can be used to evaluate all nodes at a certain distance. Note that level and tree layouts can be separately defined for each branch.

Both level and tree layouts are well suited to support one of our main tasks: understanding attributes in the context of neighborhoods. To support our other main task — understanding attributes in the context of paths — we introduce path-based node sorting as illustrated in Figure 6.2(e). In this example, the tree shown in Figure 6.2(b) was reordered to guarantee that all nodes along the path A-D-G are in the sequence of the path, thereby supporting the analysis of its attribute in sequence and enabling analysts to make judgments about path effects.

6.2.2 Reshaping the Tree and Revealing Hidden Edges

The crossing-free and easily readable layout achieved by using a spanning tree comes at a cost — both tree and level layouts hide edges. A simple way to reveal all edges and neighbors of a node is to make it the root. This, however, changes the layout drastically, which can be disorienting for an analyst. An alternative is to *gather all children* of a node. In that case, all nodes that have an edge to the target node are attached as children to this node, with the exception of its ancestors. We choose not to attach ancestors as children because it would lead to similar layout changes as the make-root operation.

In addition to reshaping, we use three strategies to visualize edges that are not part of the tree. First, hidden edges are drawn for user-selected nodes. In Figure 6.2(c), hidden edges are drawn for node B, which has edges to nodes C and D, in addition to the edges to A and E that are part of the tree. This strategy is common to most tree-based network visualization techniques (e.g., [89]).

Complementary to showing hidden edges on demand, we also show a table visualizing counts for hidden edges (the number of hidden edges in the subnetwork) and network edges (the degree of the node in the underlying network), as shown in Figure 6.2(c). Whereas the former allows analysts to judge connections that are not apparent in the tree, the latter can be used to judge the node relative to the whole network, and also give analysts a sense of how many nodes would be added if the neighbors of the node were to

be added to the subnetwork.

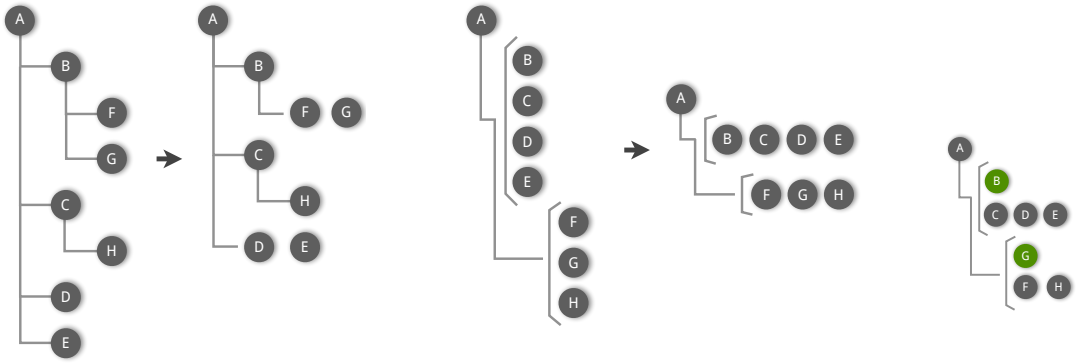
The third strategy to visualize topology is an adjacency matrix that is fully integrated with the tree, resulting in a hybrid node-link/matrix layout. The matrix is not meant to show all nodes in the subnetwork, as this would likely result in a sparse matrix and require considerable amounts of screen-space. Instead, similar to the rationale behind NodeTrix [24], the matrix is designed to show connectivity for highly connected nodes. The integration of the node-link tree and the matrix allows analysts to quickly judge the relationships of these nodes with nodes in the tree. Note that any node can be included in the adjacency matrix, not only those that are part of the subnetwork. Figure 6.2(c), for example, shows node F in the adjacency matrix, which is not included in the subnetwork. The adjacency matrix can be useful, for example, when exploring an author's papers and coauthors. Adding the author's PhD and postdoc advisors to the matrix is useful since she has likely collaborated on many papers. Using the matrix, an analyst can quickly judge which papers were written in collaboration with whom, which would not be easy to see in the tree visualization alone.

6.2.3 Hiding and Aggregation

Although a tree-based linear layout has many advantages, it also limits the number of nodes that can be concurrently displayed on the screen. To counteract this limitation, we introduce two approaches to selectively reduce the number of nodes: branch hiding and branch aggregation.

Branch hiding is common to most tree visualizations. It allows analysts to selectively hide branches of a tree that may not be relevant for the task at hand. Although it excels at saving space, the downside of hiding is that analysts no longer have access to any information about the hidden nodes.

Our second, less aggressive, approach is aggregation. This approach has the advantage of preserving both topological and attribute information in aggregate form. Aggregation is available in both tree and level modes. In tree mode, illustrated in Figure 6.3(a), only leaves are aggregated; the backbone of the tree and hence all the topological structure of the tree remain visible. When aggregating in level mode, as shown in Figure 6.3(b), all the nodes of one level are aggregated into a single row, resulting in a very compact layout.



(a) Aggregating a Tree Layout.

(b) Aggregating a Level Layout.

(c) Aggregating with DOI.

Figure 6.3. Aggregation strategies. (a) Aggregation in tree layout: leaves of the same parent are aggregated by placing them in the same row. (b) Aggregating in level mode: nodes of the same level are aggregated into a single row. (c) Aggregation with a degree-of-interest function, shown in a level mode. Nodes B and G (green) are considered to be of interest based on a degree-of-interest function, and hence are placed in their own row.

Aggregation as described above can be controlled using the tree's topology, i.e., analysts can choose to represent individual branches in aggregated mode. However, it is a common task to look for nodes with certain attribute characteristics among such a large, aggregated set. To address this, we introduce a binary degree-of-interest (DOI) function [183]. Figure 6.3(c) illustrates the effect of a DOI function on level-based aggregation. Here two nodes, B and G, shown in color, are considered of interest and hence retain their own row, whereas the others are aggregated. An example for the coauthor network would be to look for all highly cited papers of a network of prolific authors, in which case highly cited papers would be afforded their own rows, whereas papers with few citations would be aggregated.

Note that the visualization of hidden edges, the adjacency matrix, and the attribute visualization can easily be adapted to support both individual and aggregated rows.

6.2.4 Attribute Visualization

In line with the "topological attributes" shown in the edge count table and the adjacency matrix, we can leverage the linearized layout to visualize arbitrary node attributes, as illustrated in Figure 6.2(c). A variety of visualization options can conceivably show different data types in either individual or aggregated form [236].

The key benefit of the integrated attribute visualization, as opposed to a separate linked view, is that the topology of the tree can be used to sort and group the elements, reveal-

ing, e.g., dependencies along a path, or shared characteristics of all neighbors of a node. Equally valuable is the opposite approach: the attribute visualizations can be used to influence the tree layout, through both sorting and DOI functions. The columns representing attributes are well suited to interactively define such a sorting, or a data range of interest for a DOI, as shown in Figure 6.4.

6.3 Juniper Design

We implemented the concept described in the previous section in an interactive web-based tool. Here we report on the design decisions that went into realizing this tool.

Juniper has two views: 1) a query view that is used to search for individual nodes or to query for subnetworks; and 2) the main tree+table view, which contains the network and attribute visualizations. A toolbar at the top allows analysts to switch to a force-directed layout and to choose a dataset. In addition, node-type specific menus allow analysts to add attributes to the table and to filter nodes by type.

6.3.1 Querying

The query interface is the starting point for any exploration in Juniper. Analysts can browse or search for nodes in the query view and add them to the tree+table view (see Figure 6.1). The search and browsing interface is faceted by node types: when, for example, text is entered in the search field, all matches are shown in separate, type-specific facets. The faceting enables analysts to quickly find nodes of interest, even with an incomplete query. The interface also shows the degree of the nodes, so that highly connected nodes can be readily identified. A node can be added individually or together with all its neighbors. Nodes can also be added to the adjacency matrix. It is possible to add multiple roots/trees to the tree+table view simultaneously.

The query view also provides an interface to write Neo4J Cypher queries (a query language for the network database we use). Although this is an expert option, it enables analysts to retrieve arbitrary subnetworks considering both topological features and attributes.

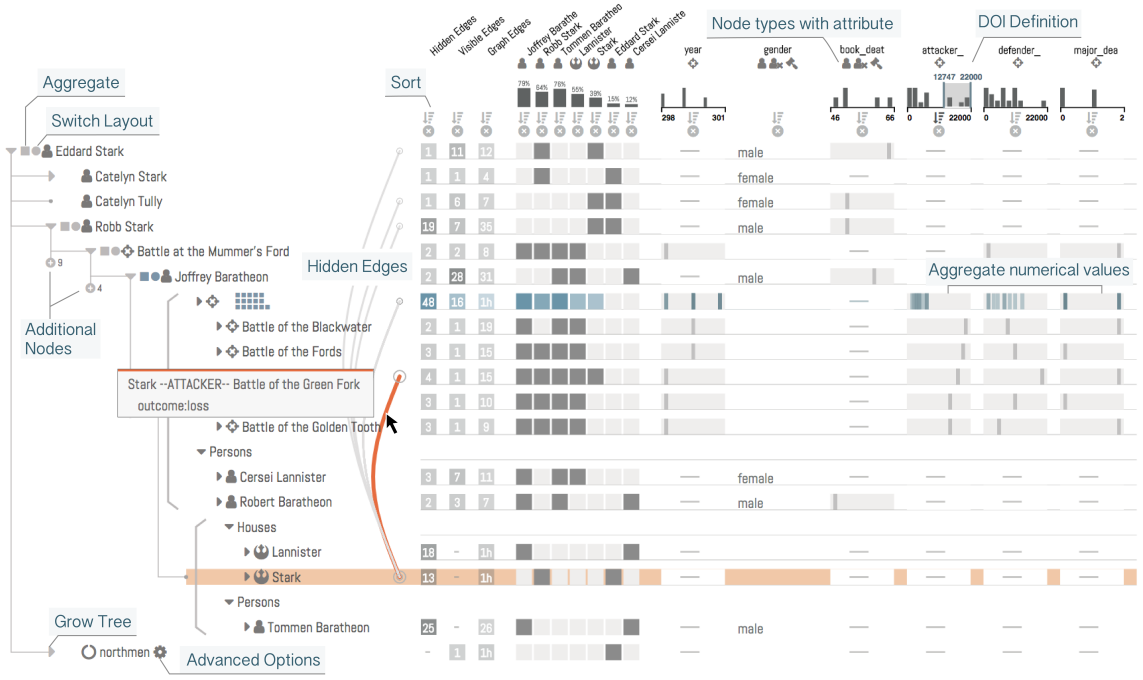


Figure 6.4. Juniper design overview using a *Game of Thrones* dataset, rooted at *Eddard Stark*, and expanded in tree layout up to *Joffrey Baratheon*. Descendants of the node Joffrey Baratheon are shown in level mode. A DOI function reveals battles with an attacker size of 10,000 and larger in the otherwise aggregated set of battles. The associated table visualizes edge counts (hidden, visible, and network edges) for both individual nodes (gray) and aggregates (blue). The adjacency matrix was autopopulated with the most connected nodes in the subnetwork. Again, individual rows are shown in gray, aggregates in shades of blue. The attribute columns are specific to node types, as shown in the column header. Aggregated rows use compact visualizations showing the values of all contained rows, where appropriate. Hidden edges are shown for the selected *House Stark*. The edge from House Stark to the *Battle of the Green Fork* is highlighted and a tooltip with information about the edge type and direction is shown.

6.3.2 Tree View

The tree view implements the concept outlined in Section 6.2. Nodes at each level are given ample space for labels, which is a common limitation in force-directed layouts. We also distinguish between different node types by showing a custom symbol for each type. Edge types and directions are shown as tool-tips where available.

The network can be grown organically by revealing neighboring nodes. In cases in which a node has more neighbors than are currently shown, a small plus sign is shown below the node that can be used to add those missing nodes, as shown in Figure 6.4.

In terms of tree-restructuring, our prototype supports the previously discussed *make root* and *gather children* operations, in addition to selectively removing nodes/branches,

and explicitly reattaching a branch at a different node, based on a hidden edge.

Hidden edges are shown for the selected node, *House Stark*, in Figure 6.4.

6.3.2.1 Layout and Aggregation

Figure 6.4 shows the implementation of the previously described layout strategies. The example shown is a *Game of Thrones* network that contains many different node types. The tree originates at a person, *Eddard Stark*, who has a connection to the *Battle at the Mummer's Ford* via an intermediate person, *Robb Stark*. The layout at the root is a tree layout, but descendants of *Joffrey Baratheon* are shown in level layout, which is indicated by the brackets replacing direct connections. Note that the branch starting at Joffrey is aggregated. We show aggregated nodes as little squares, which allows easy size estimation, and facet them by node type, so that analysts can quickly see how many nodes of a certain type are in each aggregate. Tooltips on the aggregated nodes reveal the node title. Persons and Houses are manually deaggregated; for Battles we use a degree-of-interest function to partially deaggregate battles above a certain *attacker size* (see label "DOI definition" in Figure 6.4).

6.3.3 Edge Count Table and Adjacency Matrix

The table for edge counts described in Section 6.2 is realized with a redundant encoding using color saturation and exact numbers, as shown in Figure 6.4. Numbers with three or more digits are shortened to the most significant digit plus 'h' for hundred and 'k' for thousand. Since aggregate values and individual nodes are commonly of different scales, we use separate color scales for individual rows (gray) and aggregates (blue). The color scales are defined independently for each column, since the number of hidden edges is expected to be much smaller than the number of network edges, for example.

The purpose of the adjacency matrix is to further expose the connections in the network that are not captured in the tree. As discussed in Section 6.2, we do not show all nodes in the matrix column, but rather selected nodes that complement the tree well in a hybrid node-link/matrix layout [24]. Nodes can be added to the table from the query view or the tree. We also autopopulate the matrix with the most connected nodes in the tree since highly connected nodes are likely to have many hidden edges. As in the edge count table, we use grayscale (binary in this case) for individual rows and a blue color scale

for aggregate rows. In contrast to the edge count table, the color scales are normalized on a per-row basis to account for aggregates of different sizes.

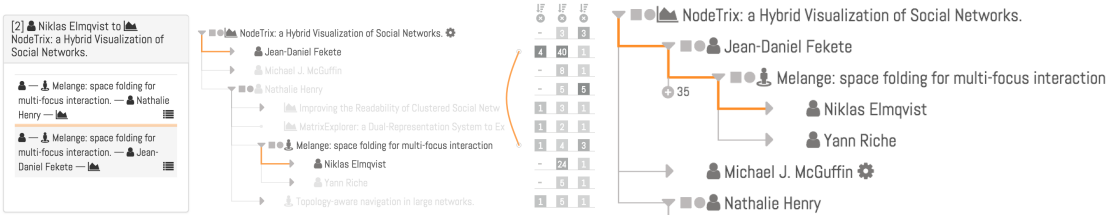
6.3.4 Attribute Table

The attribute table can be used to visualize rich data associated with the nodes. Each column in the table corresponds to an attribute for one or multiple node types. Most attributes are defined for only one node type, which can result in a sparse table if a network contains many different node types. For numerical data, we use a vertical line placed along a scale, as it uses position, the most powerful visual channel available. Exact values are shown on hover. We visualize aggregate rows by drawing multiple lines in the same cell, as shown for the aggregate cell for ✥ battles and defender size column in Figure 6.4, for example. By using transparency, we can ensure that overlapping lines are noticeable.

The attribute table, the edge count table, and the adjacency matrix also serve as interfaces for sorting and defining degrees of interest, as shown in Figure 6.4. Sorting is applied only within the levels of the tree to avoid edge crossings. Columns can be arranged arbitrarily through drag and drop.

6.3.4.1 Path Visualization

A common task in networks is to find a short(est) path between two nodes [62]. Since shortest paths can be hidden when using a tree-based layout, Juniper provides an explicit path search feature to quickly identify all shortest paths between two nodes (Figure 6.5). A dedicated view lists all the paths of the same length. Note that this list is limited to paths



(a) Path Preview (b) Sequential Path

Figure 6.5. Shortest path search and visualization. (a) A shortest path search between *Niklas Elmqvist* and the *NodeTrix* paper reveals two paths, shown in the view on the left. Both paths go through the *Melange* paper, but one continues through *Nathalie Henry*, the other through *Jean-Daniel Fekete*. The path via Jean-Daniel Fekete uses a hidden edge, which is shown when hovering over the path. (b) The selected path was laid out sequentially.

in the subnetwork, but it could easily be extended to the whole network. When hovering over a path, the path is highlighted in the tree and shows hidden edges if necessary. On demand, analysts can enforce that all nodes in a selected path are laid out sequentially in the tree and, by extension, in the table (Figure 6.5(b)). This is an example of how topological features can be used to lay out the attribute table to study potential network effects in attribute space.

6.4 Evaluation — Use Cases

Here we show several explorations using Juniper for focus tasks. We demonstrate how both attribute and network data are used in conjunction to gain insights.

6.4.1 *Game of Thrones* Network

The network we use in this example is based on the popular books and television show *Game of Thrones* or *A Song of Ice and Fire* by George R. R. Martin. The dataset is available on Kaggle [1]. We followed instructions to import them into Neo4j [2]. The network captures several types of relationships among story characters, noble houses, battles, books, cultures, etc. The network contains about 2,500 nodes, 17,000 edges, 18 attributes, and 11 node types.

We start our exploration with one of the main characters in the show, *Eddard Stark*. We see that he is associated with a handful of people, as well as all five of the books. The books are a hub of connectivity with ties to most characters in the story. As this is not helpful for our investigation, we filter out nodes of type book. We know that Eddard was killed by *Joffrey Baratheon*, so we add him to the tree through the search interface. Surprisingly, the dataset does not capture this direct connection between Eddard and Joffrey. We see that they are connected through a set node (both are nobles) instead, which is not interesting, so we filter out these nodes, leaving us with a connection between the Starks and Joffrey through the *Battle of the Mummer's Ford*, as shown in Figure 6.4. Next, we are interested in seeing all of Joffrey's connections, so we use the gather children operation. We see that he is connected to several battles, a few people, and to *House Lannister*.

---

[1] `https://www.kaggle.com/mylesoneill/game-of-thrones/`

[2] `https://tbgraph.wordpress.com/2017/06/25/neo4j-game-of-thrones-part-3/`

We want to get a better understanding of the battles and what the role of the opposing houses of Stark and Lannister is in them, so we switch the branch starting at Joffrey into level mode, which groups all of his node's descendants by their type. We then aggregate battles, add several attributes related to battles to the table and inspect them. We see in the *attacker size* aggregate cell that battles seemed to fall into one of two groups: a few large battles with an attacker size of over 10,000 people and many smaller battles. We are particularly interested in understanding the large battles, so we use a brush on the histogram for that column to set a DOI function to reveal all battles with an attacking force of more than 10,000 people, as shown in Figure 6.4.

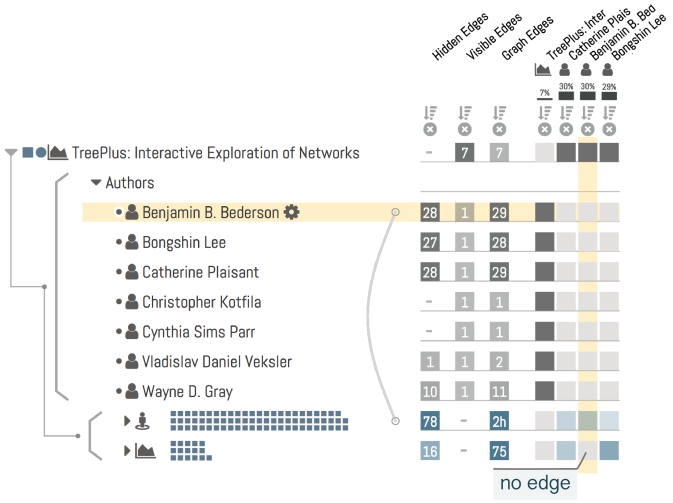
Because we are interested in the involvement of the Stark and Lannister houses in these large battles, we hover over each independently to see their connections to the battles. We see that the Stark house is associated with only one of these large battles — the *Battle of the Green Fork*. Hovering over this edge reveals that House Stark was the attacker and lost this battle. Inspecting the adjacency matrix cell that connects House Stark to the aggregated smaller battles shows us that House Stark was associated with 9 of the 16 battles with an attacker size of under 10,000. Clearly, the direct interaction between Lannisters and Starks in battle happened mainly in smaller battles.

6.4.2 Exploring a Coauthor Network

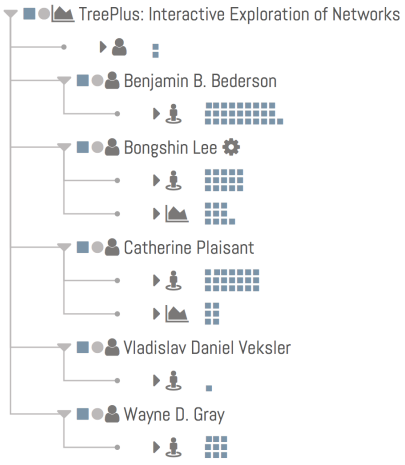
We curated the coauthor dataset introduced previously by retrieving a list of papers from DPLP. [3] We extracted all papers published at ACM CHI and IEEE TVCG up to 2015. We have also included additional attributes about papers based on the visualization publication dataset compiled by Isenberg et al. [237]. We used this information to also compile aggregate citation counts for authors.

We start by querying for a paper that is relevant for this manuscript: the *TreePlus* paper by Lee et al. [89]. By expanding its neighbors (Figure 6.6(a)), we reveal that the paper has seven coauthors. Several of the authors are familiar names, but we would like to see which ones are the most prolific scholars. To answer this, we scan the *network edges* column, which corresponds to the number of papers these authors have published at CHI and TVCG combined. We see that *Catherine Plaisant* and *Ben Bederson* have published 29

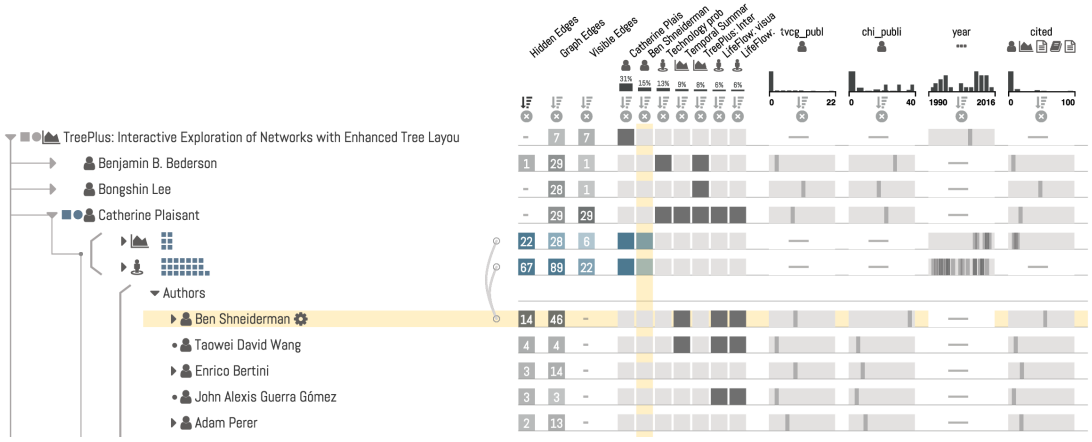
[3] `https://dblp.uni-trier.de/`



(a) Distribution of papers by authors of TreePlus.



(b) Individual paper distributions.



(c) Frequent coauthors of Catherine Plaisant.

Figure 6.6. A use case for exploring the relationships between scholars and papers. (a) Distribution of papers of authors of the *TreePlus* paper across CHI and TVCG. These authors have published about five times as much at CHI as in TVCG. In particular, *Ben Bederson* has no TVCG paper other than the TreePlus paper (which is the root), as is evident from the matrix and the missing link from his node to the TVCG aggregate. (b) Distribution of papers for individual authors. *Bongshin Lee* has published most evenly between TVCG and CHI. (c) Frequent coauthors of *Catherine Plaisant*. Authors are sorted by the number of hidden edges. *Ben Shneiderman* is a frequent collaborator, but he has also published many papers with others. *Taowei David Wang* has published all his papers with Catherine Plaisant.

papers each, and *Bongshin Lee* has published 28. But does this group combined publish more at CHI or at TVCG? To answer that question, we expand all neighbors of these authors (their papers) and put the tree into level layout in order to group together all papers and to group them by type. However, as this combined list is quite long, we aggregate the papers, as shown in Figure 6.6(a). We see that, overall, there are about five times as many papers at CHI as in TVCG for these authors. By looking at the adjacency matrix cell for the TVCG papers and Ben Bederson, we can see that he does not have a TVCG paper, other than TreePlus, but he is a very prolific author at CHI. We confirm this by hovering over his node, which reveals an edge to the CHI aggregate but not to TVCG.

Next, we focus on the other authors' papers. Have they published exclusively at CHI, in TVCG, or both? By switching from aggregated level to tree mode, we see that Catherine Plaisant and Bongshin Lee have published frequently at both CHI and TVCG, as shown in Figure 6.6(b). We hone in on Catherine Plaisant's papers by gathering all her coauthored papers, and then we look at the aggregate information for the years she published (see Figure 6.1). We see that she was immensely successful at CHI in the 90s and continued publishing at CHI afterwards, and that her TVCG publications start in 2008, soon after the first VIS publications appeared in TVCG.

Now we want to learn about Catherine Plaisant's coauthors. Has she published with many different people, or does she have consistent collaboration partners? We set the branch starting with her to level mode, expand all of her children, and aggregate the papers. Next we sort her coauthors by the number of hidden edges — those edges correspond to papers these authors have coauthored with Catherine Plaisant (Figure 6.6(c)). We quickly identify that she has collaborated extensively with Ben Shneiderman. She has written 14 papers at TVCG and CHI together with him. We also see that Ben Shneiderman has published many more papers at CHI (38) than in TVCG (8). We discover that her second-most frequent coauthor is Taowei David Wang, who has published all of his four papers together with her.

Cahterine Plaisant has also published with other prolific scholars, which we can identify by sorting the authors by citation. To clean the list up, we aggregate and set a DOI to show only authors with more than 15 citations (Figure 6.1; note that these citation

counts reflect only citation included in this dataset [237]). We see that, in addition to Ben Shneiderman, *Jean-Daniel Fekete*, *Petra Isenberg*, *Nathalie Henry Riche* and *Heidi Lam* are in this list. However, looking at the hidden edge column, we see that these have been coauthors on only one or two papers in our dataset.

6.5 Discussion

Juniper is designed for the tree-based network exploration of highly multivariate networks. As discussed in Chapter 3, Juniper addresses focus tasks, such as those related to adjacency and paths, yet always in the context of the relationship between the topology and attributes of a network.

As far as the analysis of topology is concerned, tree-based network exploration has been shown by Lee et al. [89] to perform better than force-directed layouts for various tasks, including path-based and connectivity- based tasks. The same study also showed that tree-based network exploration leads to significantly higher confidence and that users preferred the tree-based layout.

Our goal in the development of Juniper was to (1) improve on the current state of the art in tree-based network exploration, by providing novel visual encodings and interactions, such as topology/path-based sorting, DOI-based aggregation, attribute-driven sorting, and the combination of an adjacency matrix with the tree-layout, and (2) leverage the tree-based layout to visualize attributes, tightly integrated with topology. We argue that Juniper is well suited to address our key tasks: understanding attributes in the context of paths in a network and understanding attributes in the context of neighborhoods.

One of the downsides of using a tree-based layout is that it is difficult to understand cycles in a network. Although we believe that the visualization of hidden edges makes it possible to do that in Juniper, it is not the ideal solution because it requires interaction to uncover a cycle. We are considering various strategies to address this problem, including a supplemental view for cycles, a special encoding along the tree, or breaking with the tree-convention for selected nodes.

Our technique targets focus tasks, yet overviews can be useful in some scenarios. Although we chose not to include an overview visualization to be concise and focused, we envision that a production-quality network analysis system would also provide overview

techniques as, for example, described by van den Elzen and van Wijk [17]. In such an integrated tool, analysts could seamlessly switch between representations optimized for overview and focus tasks.

6.5.1 Scalability

Since Juniper is a bottom-up network visualization technique, the size of the underlying network is limited only by the capabilities and performance of the graph database. The largest network we currently include in our demo (the co-author network) has about 34,000 nodes and 90,000 edges and results in no noticeable delays for common queries.

The scalability of the subnetwork is limited by the number of rows that can be simultaneously displayed. On a large desktop screen, we can show about 50-60 rows. The number of rows, however, corresponds to the number of nodes only when no aggregation is used. We found that the use of aggregation combined with a DOI function is a very efficient way to explore subnetworks with a few hundred nodes, depending on the properties of the network. In cases in which more rows are displayed than can be fit on the screen, we use scrolling. However, we currently do not provide a good solution for linking to off-screen content; hence working with many more rows can be tedious.

In terms of the number of attributes, Juniper is exceptionally scalable compared to other multivariate network visualization techniques. We consciously reserve a sizable portion of the available screen for making long node labels, such as paper titles, readable. Even with that much space dedicated to labels, we can display $10 - 20$ additional attribute columns on a desktop display. Our current visual encodings for attribute visualization favor precision and details over compactness; more compact representations, such as those used in enRoute [58], are conceivable and could increase that number considerably.

6.5.2 Comparison to Related Techniques

We compare Juniper to TreePlus [89], since TreePlus is the most comprehensive of all the tree-based network visualization techniques discussed in Chapter 2. Although TreePlus shares the basic idea of tree-based network exploration with Juniper and was an important inspiration for our work, Juniper introduces several novel concepts that go significantly beyond the capabilities of TreePlus. The **key distinction from TreePlus is our ability to visualize rich attribute data**, due to the linearized layout and the juxtaposition

with the table. However, we also argue that Juniper is at least competitive with TreePlus when considering only topological tasks. Even though the linear layout needs more space than the layout chosen in TreePlus, we argue that our aggregation methods counteract the increased spatial demand of the linear layout, and that our DOI-based deaggregation is effective at revealing relevant nodes and connections even when aggregation is used. TreePlus also does not have a level layout, which can be used to quickly identify nodes at a certain distance from the root.

Lineage [1] is a domain-specific tool developed for visualizing clinical genealogies and was published recently by some authors of this manuscript. Although **Lineage** shares the idea of using a linearized tree to visualize multivariate attributes of that tree in a table, it **is in fact a tree visualization tool, and not a general network visualization technique**, like Juniper. The genealogies that can be visualized in Lineage have to be tree-like, i.e., have to trace back to a single founder. Rare cross-links within the tree are removed by duplicating the nodes in a preprocessing step. Since Lineage visualizes trees, it has no notion of reshaping a tree based on a network, does not show topological context by combined a node link and a matrix layout, and can represent only static instances of a tree, instead of growing a tree from a subnetwork that is dynamically extracted from a large network. Lineage does not support path-linearization and has no level mode. Lineage is an important tool in its niche application area. Juniper, in contrast, is a general purpose multivariate network visualization technique with the potential for applications in many domains.

6.5.3 Evaluation

We considered various strategies to evaluate Juniper against the claims we make: that it is a well-suited technique for focus tasks in multivariate network analysis. We considered qualitative/usability evaluation, case studies, and insight-based evaluation, which we have rejected for different reasons. Ultimately, we decided between: 1) quantitative evaluation of task performance (time, correctness) and 2) evaluation by justifying the design rationale and providing usage scenarios.

With regard to quantitative evaluation, the key choices to make in the study design are: 1) the tasks to use in the evaluation and 2) the comparison target. A comparison to

a state-of-the-art system like Cytoscape introduces confounders and makes comparisons between a tool specifically designed for certain tasks with a general purpose tool. The best approach to quantitatively evaluate the core contribution of a complex technique such as Juniper would be to implement a reasonable alternative in the same general framework. We could compare Juniper to an MCV system using our attribute table and a force-directed layout. However, although we provide a simple force-directed layout for illustration in our prototype implementation, we do not provide advanced features such as aggregation, expanding or collapsing branches, etc. Since we have no canonical way to implement these features, this approach would require significant effort in designing such a system. It would also introduce additional potential confounders, because it matters how (well) these features are implemented. We instead opt for evaluation by providing a detailed rationale for our design [238] and demonstrate its usefulness in use cases. In line with recent successful papers (e.g., [17],[135],[239],[240]), we believe that design arguments and demonstrations through use cases provide excellent evidence for the utility of complex, interactive visualization techniques.

6.6 Conclusion

We believe that Juniper is widely applicable to different network datasets across various domains. Juniper's strengths are in interactive exploration and in supporting focus tasks for multivariate network visualization. In the future we hope to develop techniques for connecting to off-screen nodes, which is a current scalability limitation. Also, allowing duplicate nodes could be helpful for certain tasks, yet node duplication requires careful encoding to not confuse users. Juniper currently also does not support rich edge attributes. Edge attributes of visible edges could be seamlessly integrated into Juniper, as the tree structure guarantees that each node has exactly one incoming edge. As a result, we could add rows representing edges to the table right above the destination node and add columns for the edge attributes. A drawback to this solution is that it cannot be used for hidden edges. Attributes of multiple hidden edges could be shown as extra columns in the table with their destination node, visualizing attributes from multiple edge in each cell. Finally, for networks with many different node types, the attribute table can be sparse.

CHAPTER 7

EVALUATION: CROWDSOURCED COMPARISON OF NODE-LINK DIAGRAMS AND ADJACENCY MATRICES

In our typology of MVN visualizations (Chapter 4), we discuss 11 types of multivariate network (MVN) visualizations and give recommendations of when to use which visualization technique. These recommendations, however, are mostly based on visualization best practices and rely to only a small degree on empirical data, perpetuating the notion of visualization as an "empire built on sand" [241]. How do we know whether a particular visual encoding or an interaction technique is better than the many alternatives available? The visualization literature largely relies on two approaches: controlled experiments that measure correctness and time on simplified tasks for well-defined stimuli [220] and evaluations with experts, such as insight-based evaluation [242], [243] and case studies [195]. Carpendale [244] discusses the conflict between evaluating visualizations with real users and real datasets, with high ecological validity, and controlled experiments that require recruiting a large enough participant sample to draw quantitative conclusions. Running controlled experiments requires abstraction and simplification of real-life tasks and careful variation of a small number of design factors [220].

In this chapter, we attempt to answer questions about the merits of two multivariate network visualization techniques by pushing the limits of controlled experiments for complex, interactive visualization techniques. We compare two MVN visualizations: node-link (NL) diagrams with on-node encoding and adjacency matrices (AM) with a juxtaposed tabular visualization. Testing these two conditions required the design and implementation of two functional prototypes. We designed and implemented these techniques based on existing guidelines, and followed a multistage testing and piloting process. We also

elicited feedback from experts on network visualization to validate and refine the designs. The aim of these activities was to ensure that the two conditions in the experiment resemble visualizations that might be used in practice. We postulate eight hypotheses and evaluate them with a set of 16 tasks, which we derive from the task analysis of Nobre et al. [3]. We also report on insights generated in an open-ended task.

Our contributions are twofold: We provide the first set of empirical evidence on the performance of two important MVN visualization techniques for different tasks, and we develop and reflect on an approach for controlled experiments using complex, validated visualization techniques.

7.1 Related Work

Here we provide an analysis of prior evaluations of network visualization techniques, followed by a discussion of the current landscape of crowdsourced evaluations of interactive visualization techniques. For more general prior work regarding MVN visualizations, we refer to Chapter 2, which contains the related work for this dissertation.

7.1.1 Network Evaluation Studies

There is a long history of work that has evaluated the merits of different network representations [245]. However, most of this work focuses on network topology and treats node and edge attributes only cursorily, if at all. We limit our discussion to larger studies that evaluate NLs and/or AMs.

Several studies compare NL and AM representations [23], [29], [246]–[248]. Ghoniem et al. [23] assessed the performance of both approaches using seven topology-based tasks on graphs of sizes between 20 and 100 nodes and densities from 20% to 60% of all possible edges. Interactivity was limited to selecting nodes and links. They found that NL outperformed AM in small and sparse graphs, but for larger or more dense graphs, AM produced more accurate results. The exception is path-based tasks, where node-link diagrams outperformed the adjacency matrix regardless of network size or density. In a follow-up study, Keller et al. [246] evaluated six tasks on a domain-specific, directed network, using both NL and AM representations. They confirmed the results of Ghoniem et al. for the investigated network types. Okoe et al. [29] also reproduced the Ghoniem et al. study

for larger networks (up to 330 nodes), but for much sparser graphs with densities of 1.6% to 3.2% of edges. They used a more diverse set of tasks in a large, crowdsourced study. Interactions included panning and zooming, moving nodes in the NL condition, and highlighting nodes and incident edges. Color was used to encode clusters that were detected algorithmically. Their work confirms earlier findings on each technique and reveals that adjacency matrices perform better on cluster tasks.

Ren et al. [248] compared NL with two sorting variants of AMs for networks of 20 and 50 nodes in a large, crowdsourced study. They found that NL resulted in higher accuracy and faster response time, but that this difference diminished as participants became familiar with the visualizations.

Christensen et al. [249] compared NL, AM, and Quilts [151] for path-finding tasks in layered networks. They showed that, for layered networks, participants performed best with quilts, followed by NL with AM as the slowest of the three representations. Chang et al. [247] ran a study where both a node-link diagram and a matrix representation of a weighted networks (including edge attributes) were displayed to the user simultaneously or individually. They found that combined views did not perform any worse than individual views, but postulated that this may not hold for more complex tasks or more experienced users.

In a recent survey, Yoghourdjian et al. [245] gave an overview of empirical studies on node link diagrams. They found that mostly a small range of graph sizes and structures — small and sparse graphs — have been used in experimental evaluations of network visualization techniques. Studies that use larger networks typically test interactive techniques that include a query interface, so that only a small portion of the network is visualized at any given time.

Three studies evaluated approaches for visualizing two or more edge attributes. Alper et al. [143] found that for tasks involving the comparison of weighted graphs, AMs outperformed NLs. Abuthawabeh et al. [250] ran a user study comparing AMs with a technique that depicts multiple types of edges in separate, parallel, node-link diagrams. They found that participants identified the same graph structures with both visualizations. Schoeffel et al. [251] encoded edge attributes as bars directly on the edges in NLs. Their study on a small network (10 nodes, 10-15 edges, up to five attributes) revealed that this can be useful

for small graphs with no edge crossings.

The existing work on network evaluation studies thus far has mostly focused on the topology of the network. No studies currently consider node attributes beyond a simple, topology-derived attribute, such as cluster membership or node degree, which means we currently have no guidance based on empirical data about which network visualization technique to use when both node attributes and network topology are relevant to an analysis.

7.1.2 Evaluation of Interactive Visualization Systems

User studies are among the most common forms of evaluation within the field of information visualization [220]. Lam et al. [220] have categorized user studies that are aimed at evaluating user performance into two types: 1) understanding the limits of visual perception and cognition for specific visual encodings; and 2) assessing how one visualization or interaction technique compares to another. User studies can be carried out either in a controlled lab setting or by using a crowdsourcing platform. Although lab studies afford the most control over participant selection, participant attention, training, and the testing environment, they also incur a high cost of recruiting users, as well as an inherently limited participant pool [252]. Crowdsourcing platforms offer a potential solution to this problem by providing access to a much larger group of participants. The existing user performance work on evaluating interactive visualizations is characterized primarily by validating a new approach comparing it with existing ones [93], [253], which is almost exclusively carried out in a lab setting with a smaller number of participants. Even though crowdsourced studies have been frequently used for performance evaluations of the perceptual type (e.g., [254]–[256]), they are relatively scarce for evaluating interactive visualization techniques (e.g., [29], [257]). This scarcity may stem from the perceived challenges of using a remote group of nonexpert participants to evaluate an interactive system — a topic we investigate in this work. Additionally, very little work has compared two or more existing **complex, interactive** techniques.

7.2 Challenges

The visualization community has embraced a wide set of quantitative and qualitative evaluation methodologies, ranging from quantitative experiments conducted in a lab or on crowdsourcing platforms, to qualitative studies, to insight-based evaluation, to case studies [220]. In this work, we conduct a crowdsourced study with two complex, interactive visualization techniques. Developing such techniques requires many design decisions. How can we know that any effects we observe are not confounded by any of these design decisions? Carpendale [244] describes three factors to consider: generalizability (can a study be applied to other people and situations), precision (can a study be definite about the measurement and can it account for the factors), and realism (is the context of the study like the context in which it will be used). However, current evaluation methodologies typically cannot satisfy all three simultaneously. A common approach to designing a quantitative study is to carefully modify selected factors or variables, so that the effect of each factor can be isolated. This approach leads to studies that are precise, but frequently not realistic. The need to isolate factors and variables significantly limits the progress we can make with one study and reduces realism. In this work, we take a different approach: we compare two techniques that are distinct in many ways simultaneously. Although this approach poses a threat to generalizability, we mitigate this threat by employing a rigorous design process following existing evidence and our own expertise, and by validating our designs in a multistaged qualitative process. Being able to compare two complex techniques increases realism since we can include factors that a real system would have. By simultaneously designing both techniques, we can better account for confounding variables than if we compared two existing systems that were designed separately.

A related challenge is that in a crowdsourced study, we have to use novices as proxies for experts. Complex and interactive visualization techniques designed for experts are frequently not immediately intuitive and require learning before they can be useful. Obviously, it would be best to validate a system with the user group it was designed for. In practice, however, recruiting researchers knowledgeable about network analysis is difficult, and becomes de facto impossible when a study aims to increase precision by increasing the number of participants. In our study, we attempt to educate participants about a visualization technique through an extensive training program (by crowdsourcing

standards). We argue that careful training makes it possible to study interactive visualization techniques that would otherwise be suitable only for experts.

7.3 Visualization and Interaction Design

Several approaches are available for encoding both the topology and the attributes of a network [3]. For this study, we chose two common encodings: a node-link diagram with on-node/on-edge encoding (NL), and an adjacency matrix with embedded edge encoding and a juxtaposed table for the node attributes (AM). We chose these two techniques because: 1) they are the most common generic network visualization techniques; and 2) several previous studies have compared NL and AM for topology-centric tasks [23], [29], [245], [248], [258], which allows us to use the findings of these studies to design our techniques based on these empirical recommendations.

When designing the techniques, we made choices that we justify in this section. We adopted the following design principles: 1) Use the most perceptually efficient encoding available for data, given the affordances of the technique. 2) Follow common practice in network visualization systems, such as Cytoscape of Gephi. 3) Provide a set of common, technique-specific interactions. In order to ensure our design decisions for each visualization technique were appropriate for experimentation, we used a multistage validation approach with expert evaluations and three rounds of pilots. Both designs can be viewed at `https://vdl.sci.utah.edu/mvnv-study/`, or by running the supplementary code.

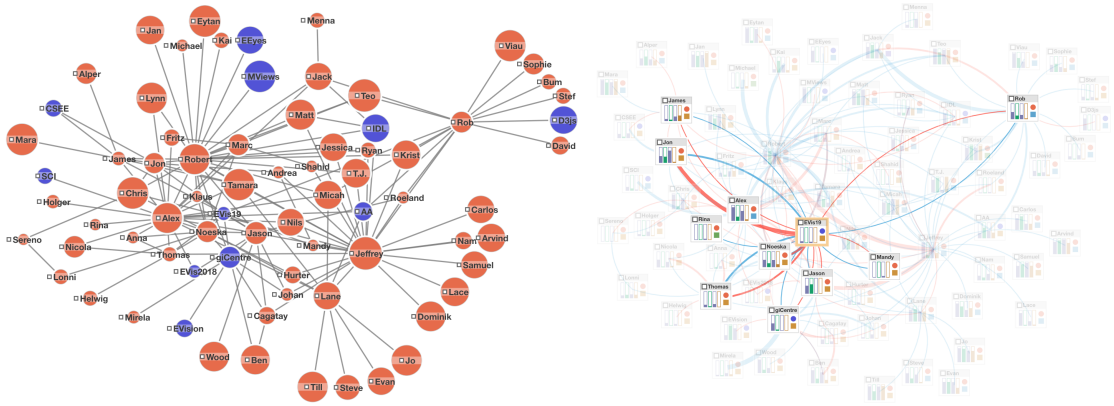
For the feedback from visualization experts, we followed the heuristic evaluation method proposed by Wall et al. [259] and added a set of custom questions tailored to our design decisions. We ran a pilot with one expert to evaluate our survey, which led to design recommendations we implemented but also to issues identified in our survey. The final survey had 69 questions that consisted of ratings on a seven-point Likert scale (high marks are good) and free-text questions asking for design suggestions and criticism. We then asked 10 experts to explore the two techniques with a set of four representative tasks and then to fill out the survey. Although responses were helpful, we received only two responses within two weeks and one more response after we had already conducted the main study. The pilot and the two timely responses yielded multiple suggestions for improvements, which we incorporated, or, if not, discuss in this chapter. Both the survey

and the results are included in the supplemental material of Nobre et al. [4]. Overall, experts rated the version of the tool they saw (before we implemented their suggestions for improvements) critically. The average rating for the heuristic evaluation question for the node-link diagram was 3.85; the average rating for the custom questions was 4.47. For the matrix, the average heuristic score was 5.69; the custom questions were scored at an average of 5.03. We attribute these low scores, particularly for the node-link condition, partially to the difficulty of framing the questions. We attempted to get ratings for our design decisions, assuming the two techniques and the dataset as given, i.e., the best designed node-link diagram should get a perfect 7 on all questions. We found, however, that the experts mostly judged the techniques globally.

7.3.1 Node-Link Design

We studied node-link layouts with on-node encoding, as defined by Nobre et al. [3], since this is a widely used encoding and is supported by common network visualization tools. A key decision was to use traditional network layouts, including node positioning, exclusively for visualizing topological structure. We considered alternative layouts, such as positioning nodes by attribute values, as a set of separate techniques beyond the scope of our study. We experimented with many different layout algorithms and parameterizations. We calculated layouts using Cytoscape's implementation of the Prefuse force-directed layout, which we optimized manually.

We designed two versions of **node-attribute** encodings, shown in Figure 7.1, which we used depending on the number of attributes being encoded at once. Both versions always show all node labels. For conditions involving one numerical and/or one categorical attribute, we encoded the numerical attribute with the size of a circle and the categorical attribute as color, as shown in Figure 7.1(a). Both of these choices are the highest ranked available visual channel for the respective data type [260]. This design has the advantage that both a numerical and a categorical attribute can be visualized simultaneously with little interference, and global judgments of the individual attributes are easy. For conditions involving more than two attributes, we used nested bar charts for numerical values and colored glyphs for categorical values, as shown in Figure 7.1(b). These encodings again use the most expressive visual channels available and support the comparison of multiple



(a) On-node encoding with color and size.

(b) On-node encoding with nested bars and glyphs.

Figure 7.1. Node-link designs. (a) We use node size and color for conditions with one categorical and/or one quantitative attribute. (b) We use bar charts and colored glyphs for more than two attributes. Figure (b) also shows edge types (color) and edge weights (thickness), in addition to neighborhood highlighting: only neighbors of the selected node *EVis19* are rendered opaquely.

values within a node well. For this encoding, global judgments are more difficult due to the smaller mark size and the many marks present. As the nodes have to be bigger to accommodate multiple glyphs, the network topology is less apparent than in the simpler configuration. We include a legend showing the meaning of all visual encodings in a given configuration.

We visualize **edge attributes** using color and/or edge thickness for categorical and quantitative data, respectively. Figure 7.1(b) shows two edge types with different edge weights. We use straight lines for edges for conditions with only a single type, but use separate, curved edges when edges of multiple types are present. There is conflicting evidence about the merits of curved edges [122], [261].

We limited **interactions** to a small set that we considered essential for the techniques. For both techniques, we provided a label-based search, highlighting of individual nodes, tool-tips, and highlighting of neighbors of nodes. To highlight neighbors of a selected node, we faded all non-neighbors out, as shown in Figure 7.1(b). We found the fade-out necessary in the node-link diagram due to the high amount of visual clutter caused by crossing edges and colorful nodes. Previous work found that without neighbor highlighting, users quickly became frustrated [23], [29]. We also provided the ability to drag nodes, which is essential to disambiguate edge crossings.

7.3.2 Adjacency Matrix

Adjacency matrices are the second major class of network visualization. Network topology is encoded in the matrix, where filled-in cells indicate the presence of an edge. Matrices rely on seriation or sorting algorithms to reveal topological patterns of interest, although neighborhoods are also easily identified by scanning a row or column. A key benefit of matrices is their ability to visualize dense networks, since every possible edge is already represented on the screen.

Edge attributes are commonly encoded using brightness/saturation or color for quantitative and categorical data types. We use a gray color scale to encode numerical values. Multiple edges are more difficult to encode, since the space available for edges in a cell is small. Alper et al. [143] discuss various encodings and argue for nested rectangles of different color, where the brightness encodes a numerical attribute for two edge types with weights, an approach we adopt.

A simple but efficient way of encoding **node attributes** in adjacency matrices is using juxtaposed tables. We juxtapose a tabular visualization [236], [262] with the matrix, and keep the rows consistent between the matrix and the table [27], [263], [264], as shown in Figure 7.2. As a result, we can use highly efficient encodings for the attributes. We employ

Figure 7.2. Adjacency matrix design. Two types of edge attributes are encoded using nested rectangles (red and blue). The color saturation encodes edge weights. Attributes are visualized in the juxtaposed table. The matrix is sorted by clusters, and the selected node's neighbors are highlighted in green. This figure also shows the study interface with the task instructions and the answer field, the search interface, and the legend.

aligned bars for quantitative attributes, which use position and size redundantly, and spatial region and color for categorical data. As for the node-link, we provide **interactions** such as highlighting, tool-tips, and node search. On cell/edge selection or hovers, we highlight the row and column at this intersection, and the equivalent row/column on the other side of the diagonal. We also highlight neighbors; however, in this case we use simple (green) color highlighting (see Figure 7.2, as the matrix does not suffer from cluttering. As a (more powerful) alternative to moving nodes, we introduce sorting. The matrix can be sorted interactively by node label, a seriation algorithm [148], by the neighborhood of a node, or by any attribute.

7.3.3 Discussion

It was our goal to develop two techniques that we can compare fairly by carefully designing each technique to use the best possible visual encodings and interactions. We would then be able to measure the inherent benefits and drawbacks of the techniques. Even though we believe that the visual encodings are largely equivalent, our expert evaluators argued that the interactions are not. In particular, interactive sorting in the adjacency matrix does not have an equally powerful counterpart in the node-link diagram. However, we argue that the set of interaction techniques we provide is consistent with what is commonly expected of the respective visualization techniques, and sorting/seriation is essential but also natural in matrix layouts. One expert suggested we also include sorting for nodes, resulting in attribute-driven positioning of the nodes. Even though this operation is reasonably equivalent to sorting in the adjacency matrix, it leads to significant clutter and overlap in the node-link diagram. Another suggestion from an expert was to provide a query and filter system, for example, using scented widgets in the legend. We decided against enabling filters and queries, as we would mostly be testing that query and filter system, not the inherent qualities of the alternative network visualization approaches.

7.4 Study

We aim to investigate the strengths and weaknesses of AMs and NLs for a diverse set of tasks on multivariate networks. Our study used a between-subjects design in which each participant was randomly assigned to either the AM or NL condition.

7.4.1 Procedure

In each condition, participants were assigned 16 tasks, with the first 15 presented in random order, and a final free exploration task presented at the end of the study. We recruited participants on Prolific, a crowdsourcing platform with a research focus. Based on completion times of pilot experiments, each participant was paid $12.50 USD, for an estimated duration of 40 minutes, resulting in an hourly rate of about $15 USD. All participants viewed and agreed to an IRB-approved consent form. To be eligible for the study, participants had to use a laptop or desktop device with a resolution of at least 1400x850 pixels available screen space in the browser. Our procedure consisted of five phases: *Passive Training*, *Active Training*, *Trials*, *Study*, and *Demographics and Feedback*. The full study for both conditions can be viewed at `https://vdl.sci.utah.edu/mvnv-study/`. **Passive training** was a video introduction to the dataset, as well as an explanation of the technique (NL or AM) with which the participant would be interacting. Participants had to watch the video before being allowed to continue. The training also introduced analysis strategies that are useful for tracing paths or identifying clusters. **Active training** was achieved with a guided tour of the actual visualization and interaction mechanisms. Participants had to use interactions to be allowed to proceed. During **Trials**, participants had to correctly answer two tasks to proceed with the study. These tasks were meant to further train the user on the technique and to verify participants were attentive. During the **Study** itself, tasks were presented in random order to minimize learning effects across participants. Answers were given as a set of selected nodes, through a controlled or free text form, or as a combination of both. Each task was followed by a survey asking for confidence, perceived difficulty, and comments. The study interface is shown in Figure 7.2. A final form collected **Demographics and Feedback** on the study and the training.

7.4.2 Measures

Throughout the study, we collected a set of qualitative and quantitative measures. We captured the start and end time for each phase, as well as time browsed away from the study window, which allowed us to assess the average time spent on training and trials, as well as filter out participants who rushed through the study. During the **Trials** phase, we captured all submitted answers, including incorrect ones. During the **Study** phase, we

collected time spent on each task, the submitted answer, the confidence in the submitted answer, and the perceived difficulty of the task; the latter two on a 7-point Likert scale. Through the free-response questions, we collected qualitative feedback for each task. The final **Demographics and Feedback** form includes free-response questions where users provided feedback on the training material and on the overall study. We collected rich provenance data of users interactions, including searching for node, dragging a node, (un)selecting a node, clearing selected nodes, sorting operations, and hovering on a node (which shows a tool-tip).

When calculating correctness, we used nonbinary rules that map to a 0–1 scale. For example, we gave 0.5 points for an answer that contains the second-largest node, if the task asked for the largest. We provide details on our scoring method for each task in the supplementary material of Nobre et al. [4]. When calculating time to completion, we subtracted time spent away from a tab. Although this approach does not ensure a participant paid attention in all cases, it does reduce outliers.

7.4.3 Data

As our dataset, we used a Twitter network of interactions during the EuroVis 2019 conference (collected by J. Guerra-Gomez, used with permission). We chose a Twitter network because we expected participants to be familiar with social networks. The network has 75 nodes, 143 edges, an average degree of 3.81, and a density of 0.16. We also created a smaller version of this network, which contains 25 nodes, 56 edges, an average degree of 4.48, and a density of 0.3. Following Ghoniem et al. [23], we define density as $d = \sqrt{e/n^2}$ where e is the number of edges and n the number of nodes. For the nodes, we had the attributes *names*, # *followers*, # *following (friends)*, # *tweets*, # *likes*, *account age in days*, *node type* (person or institution, categorical), and *continent of origin* (categorical). For edges, we used a categorical attribute/edge type (retweeted or mentioned), and a numerical attribute for each type that contained the number of each action (retweet or mentioned). We modified the source data in several ways: we simplified names to shorten them and clipped outliers (for example, an extreme number of followers) so that numerical values are easily comparable on a single linear scale. We also manually added account type and continent of origin. Although retweets and mentions are directed on Twitter, we simplified

the network by treating them as undirected.

For network size, we chose the largest network that could reasonably be represented as a node-link or adjacency matrix on a standard-size display without zoom or panning, while still rendering multiple attributes. For example, a node-link diagram with on-node encoding can display only a limited number of nodes before occlusion and overlap render the technique inadequate [3]. By choosing a network that reaches these limits, we aim to draw conclusions about networks that are close to the "hardest" case that can be visualized with these techniques. Larger networks either require different approaches or must be filtered first. For tasks where we believed network size could have a technique-dependent effect on task performance, we included an equivalent task on a small network and a large one. For density, we consciously chose a sparse network, because we know from prior work that adjacency matrices outperform node-link diagrams in dense networks for most tasks [23]. If we can show that the adjacency matrix outperforms the node-link diagram in sparse multivariate networks, we can generalize that they also outperform them in dense networks. We discuss the implications of our choices on generalizability in the limitations section.

7.4.4 Tasks

We created the tasks based on the two recent taxonomies for MVN tasks [3], [47]. Our tasks cover the main topological structures outlined in both taxonomies: single nodes, neighbors, clusters, and paths. The tasks are listed in Table 3.1. Details about each task are described in the supplementary material of Nobre et al. [4].

7.4.5 Hypothesis

Prior to running the study, we developed a set of hypotheses about how the two visualization approaches would compare for different types of tasks. We present the hypotheses below and later use them to frame and discuss our results.

7.4.5.1 Distractor Effects Hypothesis

In the AM, accuracy and time will be resilient to the number of distractors (node attributes that are visualized but that are not necessary for the task). Distractors will have a negative effect in terms of performance and time in the NL, since adding many attributes to

the NL will make identifying topological structures more difficult, as the nodes get bigger, and the extraneous attributes will make it harder to isolate the attribute necessary for the task. We test the hypothesis with Tasks 1, 2, 4, and 5.

7.4.5.2 Attribute Sorting Hypothesis

The AM will perform better (accuracy and time) in tasks that benefit from sorting the matrix based on attributes, such as identifying an extreme node according to a numerical attribute (Tasks 1, 2).

7.4.5.3 Scalable Attributes Hypothesis

The AM is more scalable with respect to node attributes (Task 3). It will lead to faster and more accurate decisions when many attributes ($>$3) are present, except for tasks on topological structures ill suited for adjacency matrices, such as paths.

7.4.5.4 Common Neighbor Hypothesis

The NL will lead to more accurate and faster responses for all tasks that are concerned with common neighbors of two or more nodes. Although there is weak evidence that shows the AM to be advantageous for similar tasks [29], we believe that the known benefits of NL for path-finding will be relevant for tasks of this type (Task 8).

7.4.5.5 Within-Node Comparison Hypothesis

If the task involves comparing attributes of identical scale within individual nodes (Tasks 10, 11), the NL will lead to more accurate and faster responses. We believe this to be due to the layout of the bars, as the bars in the nodes use the same axis, whereas the attributes in the matrix use adjacent columns.

7.4.5.6 Cluster Hypothesis

The AM will lead to more accurate and faster results for tasks involving clusters (Task 12). We believe that clusters are difficult to spot in the node-link diagram, especially since node size is fairly large to accommodate attributes.

7.4.5.7 Path Hypothesis

As we know from previous studies, matrices are ill suited for path-based tasks [23],[29], [246]. Hence, this hypothesis states that the NL will perform more accurately and faster for all tasks related to paths (Tasks 13, 14, 15).

7.4.5.8 Insight Generation Hypothesis

When freely exploring the network (Task 16) with the AM, participants are more likely to have attribute-based insights. Conversely, the NL will lead to more topology-based insights, such as the presence or absence of connections among nodes.

7.4.6 Pilots, Analysis, and Experiment Planning

We conducted several tests and pilots to evaluate tasks, system usability, data collection modalities, measures, and our procedure. We estimated the number of participants required to uncover effects based on a pilot run on Prolific with 20 participants. We used a power analysis between the two conditions to estimate the variance in our quantitative measures, which we combined with our observed means to estimate the number of trials required. Due to the limitations of null hypothesis significance testing, we base our analysis on best practices for fair statistical communication in HCI [265] by reporting confidence intervals and effect sizes. We compute 95% bootstrapped confidence intervals [266] and effect sizes using Cohen's d to indicate a standardized difference between two means. For each task, we display the accuracy and time results in the form of a violin plot, which approximates the density distribution of accuracy and time on task for all participants. We superimpose the mean value with a 95% confidence interval error bar to facilitate comparison. Compared to just reporting confidence intervals, violin plots have the advantage of making the distribution of the data salient. Although we include p-values from Mann-Whitney tests (given the non-normal distributions of time and accuracy data) in our figures and highlight Bonferroni-corrected significant results (we consider a corrected threshold of p=0.003), these are only a supplement to the analysis.

7.5 Results

We recruited 322 participants for this study. Half the participants were assigned the node-link diagram (NL) and the other half the adjacency matrix (AM). After reviewing all

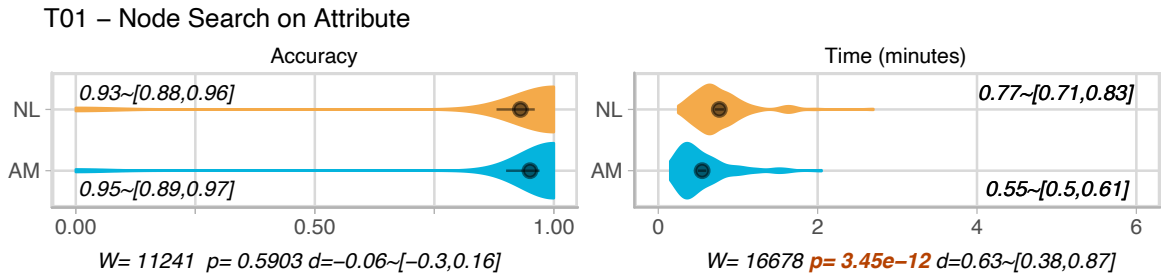
submissions, we excluded the responses of 10 AM and 9 NL participants due to low-effort answers or incomplete submissions. Submissions were classified as low-effort when the participant completed the study in under 10 minutes and had an average accuracy of under 30%. This left us with 151 valid AM and 152 valid NL submissions. Here we present a comparison of task accuracy and time to completion between the two conditions. We group the tasks according to the hypothesis they were intended to investigate. Refer to Table 3.1 and the supplementary material of Nobre et al. [4] for task descriptions, configurations, correct answers, and scoring methodology. The study data, results, and the analysis scripts are available at `https://github.com/visdesignlab/mvnv-study-analysis`.

7.5.1 Distractor Hypothesis

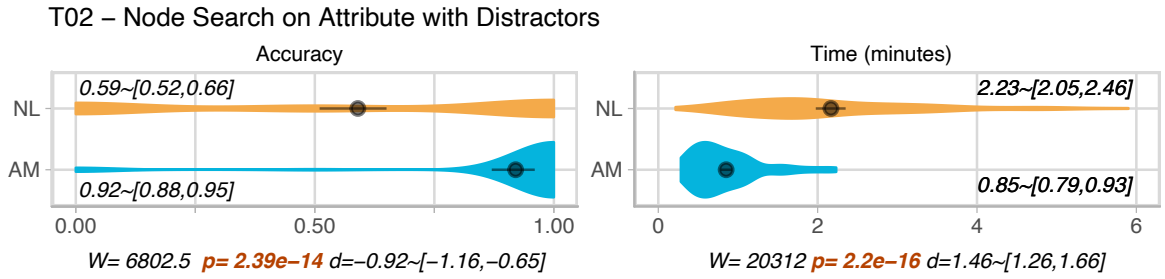
To evaluate our hypothesis that encoding nontask-essential attributes, or *distractors*, would hinder performance in the NL but not in the AM representation, participants were given two pairs of tasks, one with distractors and one without. The first of these task pairs targets single nodes.The second of these task pairs targets node neighbors.

The accuracy and time for all four tasks are shown in Figure 7.3. For the no-distractor task T1, there were no significant differences in task accuracy between NL and AM, with both conditions showing high accuracy. We found a significant but small difference in time, where AM leads to faster response times. The addition of distractors in T2 led to a strong decrease in accuracy in NL ($M = 0.59$ $[0.51, 0.65]$), but to only a small degree in the AM condition ($M = 0.92$ $[0.87, 0.96]$), resulting in a notable difference in accuracy between the two conditions. Likewise, there is a strong and significant difference in time to completion ($M = 2.23$ $[0.71, 0.83]$ for NL, $M = 0.85$ $[0.79, 0.93]$ for AM). These data confirm the distractor hypothesis for single node targets. The distractor effect does not hold for neighborhood tasks. The no-distractor condition T4 shows slightly better accuracy with the NL than the AM condition. When distractors are added in T5, both conditions decreased in accuracy, but a significant difference appears favoring the NL condition (NL $M = 0.95$ $[0.89, 0.97]$, AM $M = 0.82$ $[0.75, 0.87]$). NL leads to faster responses.

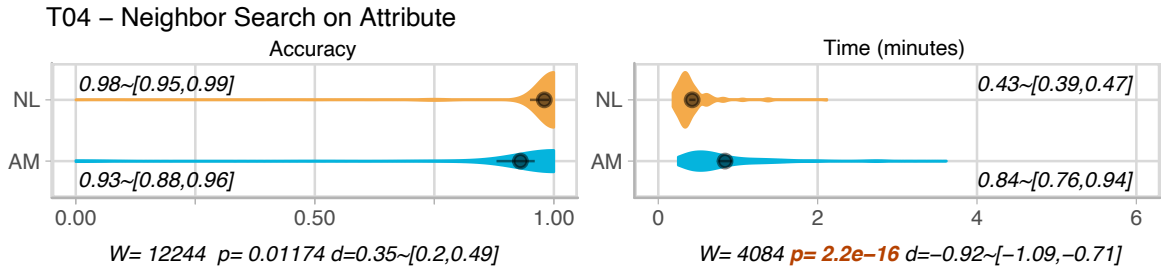
From these results, we can conclude that NL and AM are roughly equivalent for global search tasks on the nodes with a single attribute that is encoded as the node size (NL) or a sorted bar chart (AM). However, if distractors are present, the NL condition has to



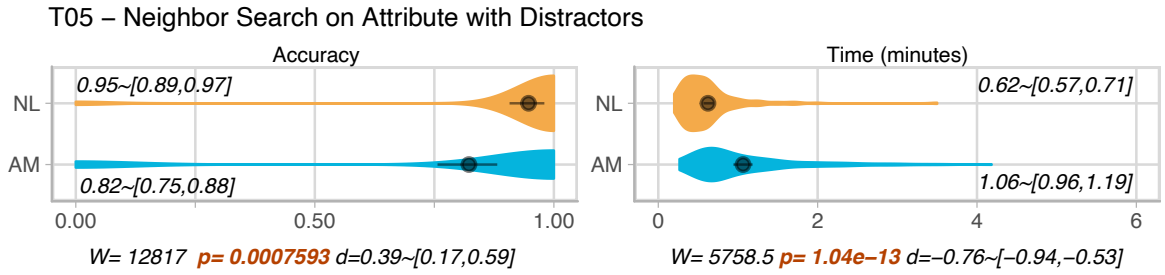
(a) Node search tasks without distractors (T1).



(b) Node search tasks with distractors (T2).



(c) Neighborhood search tasks without distractors (T4).



(d) Neighborhood search tasks with distractors (T5).

Figure 7.3. Task results for the distractor and sorting hypothesis.

accommodate the larger number of attributes with an encoding that is less amenable to global search — small nested bars — and strong differences appear in accuracy and time. In neighborhood tasks, where users first identify the node of interest through the search feature, this effect inverts, although performance in AM does not deteriorate quite as

much as does that in NL in global search. We speculate that the reason for this is that neighborhood highlighting is more efficient in NL, and that this effect drowns out any benefits the attribute representation in the AM might have.

7.5.2 Attribute Sorting Hypothesis

Tasks 1 and 2 were also used to investigate the hypothesis that tasks that relied on global maximum or minimum attribute values would benefit from the sorting inherent to the table in the AM configuration. The results (shown in Figure 7.3) indicate that when comparing bubbles in NL to sorted tables in the AM, no significant differences appear in task accuracy, yet a significant difference with a medium effect size ($d = 0.63$ $[0.38, 0.87]$) appears in time to completion. When comparing complex on-node encoding (nested bars) to sorted tables, as in T2, the effects are significantly and strongly in favor of sorted tables (accuracy: $d = -0.92$ $[-1.16, -0.65]$, time: $d = 1.46$ $[1, 26, 1.66]$). Hence, we can conclude that the attribute sorting hypothesis is correct.

7.5.3 Scalable Attribute Hypothesis

Task 3 is an advanced task designed to test the hypothesis that tasks that relied on several attributes and topology would perform better in AM condition. We speculated that the table would support easy comparison of multiple attributes across nodes, and that sorting on a single attribute could help. However, the results show that the NL condition was significantly more accurate than the AM condition, although both were of fairly low accuracy (Figure 7.4). The perceived difficulty was high, reported on average at about 5 in both conditions. Both conditions took approximately the same amount of time. Overall, this hypothesis is not supported. In retrospect, we believe that since this task

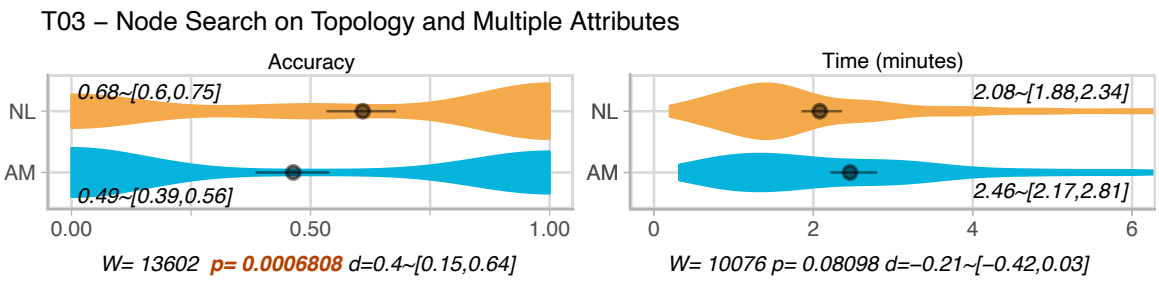


Figure 7.4. Results for scalable attribute hypothesis.

required integrating information from many different attributes and topological features, participants had to resort to a serial search, i.e., scanning all nodes, to answer it. In this case, the bars and sorting capability of the matrix did not offer a benefit.

7.5.4 Common Neighbor Hypothesis

Task 8 asks participants to select the common neighbor of two nodes with the most followers. User performance on this task (Figure 7.5) does not reveal significant differences in accuracy or in time between NL and AM, and hence the hypothesis does not hold. Overall, this outcome was surprising to us. We expected that the known benefits of path finding for node-link diagrams would extend to common neighbor tasks. Although the correct node was not directly in between the nodes in the NL condition, NL layouts in general cannot guarantee this. Also note that our scoring gave 0.5 points to the common neighbor with the second highest number of followers. The violin plot indicates that very few participants selected the second-best answer in the AM condition, whereas this was a common response in the NL condition.

7.5.5 Within-Node Comparison Hypothesis

T10 and T11 test the hypothesis that the node-link diagram is better suited to comparing attributes of the same scale within a given node, on large and small networks, respectively. Figure 7.6 shows a small but significant accuracy advantage of the NL over the AM for the large network. The smaller network task shows similar accuracies for both conditions. However, the NL condition resulted in much faster responses in both the large (NL $M = 0.87$ $[0.81, 0.96]$ vs AM $M = 1.79$ $[1.66, 1.93]$) and the small networks (NL $M = 0.71$ $[0.65, 0.83]$ vs AM $M = 1.25$ $[1.14, 1.41]$). We conclude that this hypothesis is

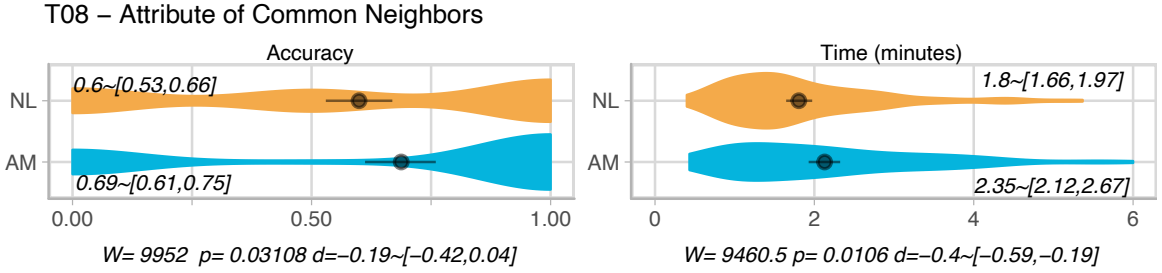


Figure 7.5. Results for common neighbor hypothesis.

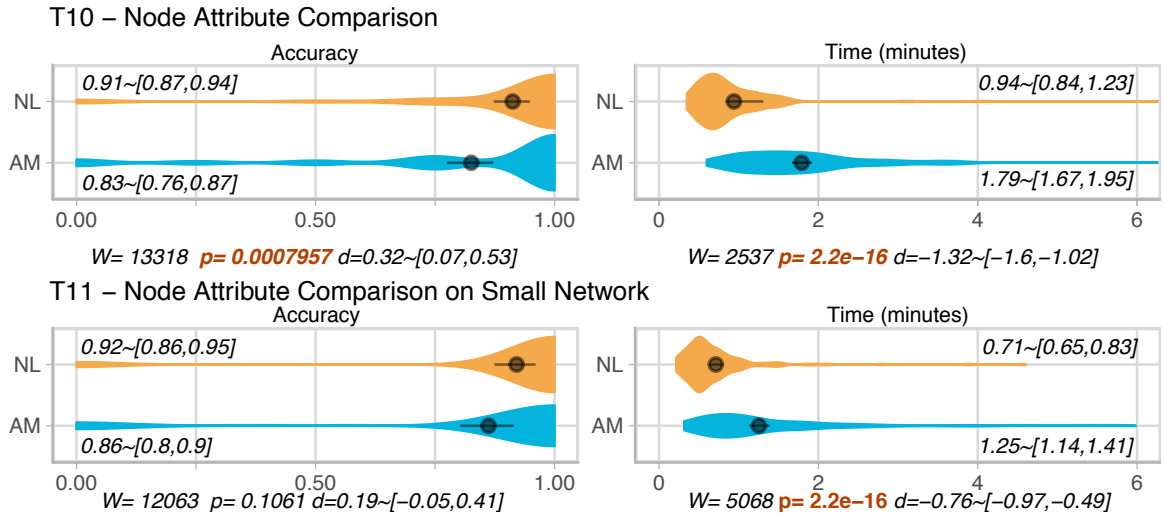


Figure 7.6. Results for within-node hypothesis.

supported, with the caveat that the task we used involved identifying neighbors of a target node. The results of Task 4 indicate that NL tends to have advantages when neighborhoods are involved. One possible reason is that a task on identifying a node globally would benefit from sorting strongly, i.e., we believe that the sorting hypothesis supersedes the node comparison hypothesis.

7.5.6 Cluster Hypothesis

To investigate our hypothesis that the AM would perform better on cluster tasks, T12 asks users to identify the nodes in a cluster, and then to average the number of followers in that cluster. Results (Figure 7.7) show significantly higher accuracies with a medium effect size for the AM condition than for the NL, and comparable time. Of note are the overall very low accuracies and the long time spent on task for both conditions. To investigate this result further, we also computed user accuracy for selecting the cluster separately from that for estimating the average attribute value. The results (Figure 7.7) indicate that users were equally able to estimate the average value of an attribute in the NL and the AM, but were better at selecting the cluster structure in the AM ($M = 0.2$ $[0.16, 0.26]$) than in the NL ($M = 0.04$ $[0.02, 0.07]$). Overall, we conclude that the hypothesis is supported.

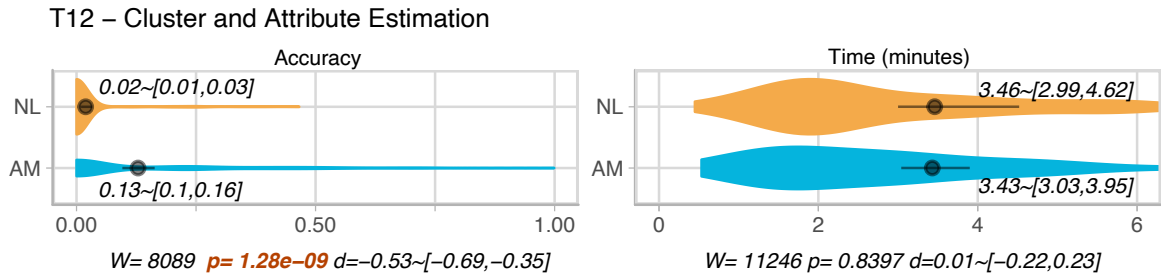


Figure 7.7. Results for cluster hypothesis.

7.5.7 Path Hypothesis

Our path hypothesis postulated that the NL would outperform the AM for all path-related tasks. T13 and T14 test a path task on a large network and a small one, respectively. The results, shown in Figure 7.8, confirm our hypothesis with the NL resulting in significantly higher accuracy for T13 and for T14, although the effect size is less pronounced in the small network. T15 is a more challenging path task and again shows a significantly higher accuracy with a large effect size in the NL vs. the AM. Participants also were more than a minute faster on average in the NL.

7.5.8 Edge Attributes

Tasks 6, 7, and 9 investigate the performance of each condition on tasks that relied on edge attributes (Figure 7.9). Task 6 and Task 9 require the user to select and inspect a single neighbor based on edge attributes. Task 7 is an overview task, requiring the user to summarize edge attributes for all edges incident to a node. All three tasks show no significant difference in accuracy or time to completion between the two conditions, possibly as a result of the visualizations' ability to highlight neighbors, thereby significantly reducing the search space for completing these tasks. Results plots for these tasks can be found in the supplementary material of Nobre et al. [4].

7.5.9 Insight Generation Hypothesis

Task 16 instructed participants to freely explore the network and report on any insights they derived from their exploration. In order to analyze the responses, we performed a qualitative coding of a sample of 120 responses (60 of each condition), categorizing insights

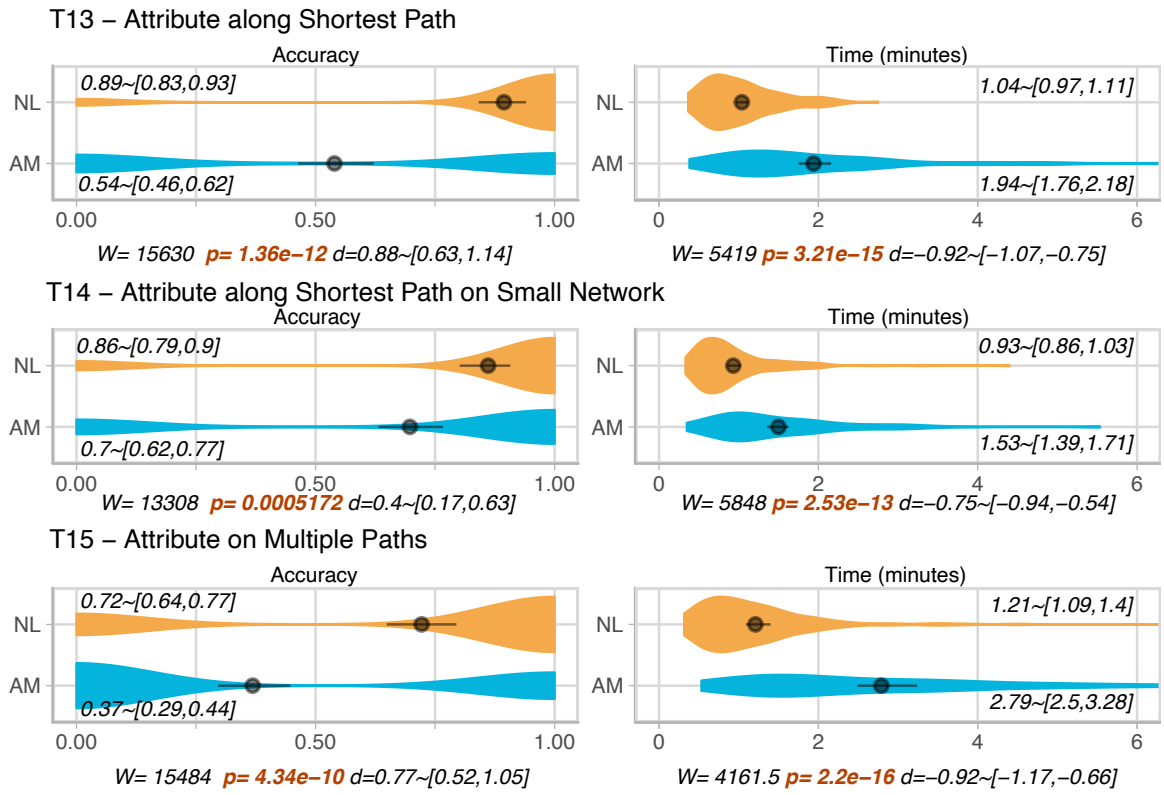


Figure 7.8. Results for path hypothesis.

into a set of codes that were derived by an initial open coding of the data. A full list of codes and their frequencies is provided in the supplementary material of Nobre et al. [4].

Two types of insight were markedly more common in the **adjacency matrix** condition: overview and ranked attribute insights. Overview insights were those that derived from an assessment of the entire network, such as "Institutions have much fewer tweets in general than a person's account." Ranked attribute insights were those that identified maximum or minimum values for one or more attributes, such as "MViews has the 2nd youngest account age; however it has the 4th largest follower count even though it also has the fewest tweets." Across-node attribute comparisons were also more common in the AM, including insights such as "Nodes with the most followers actually have much fewer tweets than those with far fewer followers."

Conversely, the **node-link diagram** favored topology-only, topology-attribute, and within-node-attribute comparison insights. Topology-only insights do not mention any at-

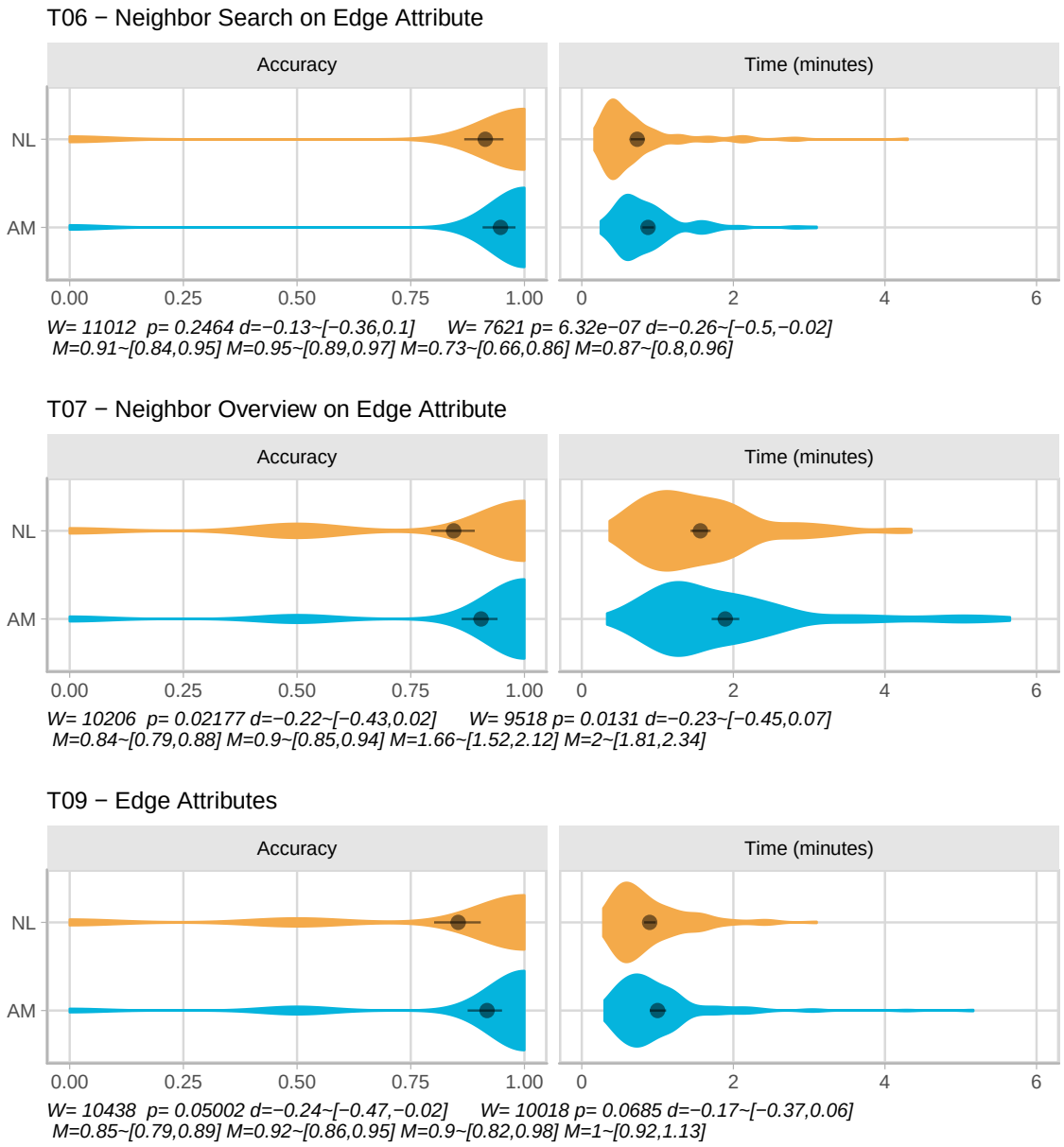


Figure 7.9. Results for edge attributes.

tributes and included "Steven, Evan, Jo, and Till have only ever had 1 interaction and they have all been with Lane." Topology-attribute insights were more common in NL and refer to those that comment on both the structure of the network and one or more attributes. Observed examples include "It does seem a bit odd that Jeffrey, Alex, and Rob have such large networks with their lower than average tweeting." Within-node attribute insights are those in which the comment relies on comparison of attributes for a given node, such

as "Jeffrey hasn't made many tweets (less than a thousand) yet has a lot of followers and a fairly long account age."

We were positively surprised by the extent and the quality of the insights, the engagement of the participants, and the ability of both techniques to reveal insights of various types, albeit with different frequency. We saw the biggest differences in overview-attribute insights (much more in AM) and ranked-attribute insight (exclusively in AM). Consequently, we consider this hypothesis to be confirmed.

7.6 Discussion

Overall, our results show that AMs are best suited for tasks on clusters, tasks that benefit from sorted attributes, and tasks that are performed in the presence of distractors and require scanning the entire network. NLs, on the other hand, are well suited for paths, tasks that rely on within-node comparisons, and tasks on neighbors given the ability of highlighting neighbors in the network. The disadvantage of the AM for performing path tasks is well known [23], [29], [47] and was once again confirmed in our study. Our results indicate that the suitability of AMs for cluster tasks, previously described by Okoe et al. [29], also holds for cluster tasks involving attributes. The cluster task in the present study incorporates attributes both in the cluster selection (only select nodes of type person) and in the attribute estimation component.

We could not find differences between the AM and NL in the ability to analyze edge attribute in three different tasks, which is in contrast to the results of Alper et al. [143], which showed that AM performed better than NL. Our NL edge design was different in that we used curved, separated edges instead of straight, bundled edges. The discrepancy in our findings may be due to the differences in study design, our different NL design, or the dataset and types of tasks used in both studies. All tasks used in the Alper et al. study involved comparing parallel edges between pairs of nodes, so the linear juxtaposition of edges in their design versus the curved edges in the present study could conceivably affect performance. **Based on these results, we conclude that adjacency matrices are at least as good for edge attribute tasks as node-link diagrams, even for the very sparse networks we tested.**

Okoe et al. [29] found weak evidence for AM to perform better in common neighbor

tasks. Our findings do not support this, if the tasks also consider attributes. Both the NL and the AM performed about equally well. User insights from the free-explore task corroborate the results from other tasks. For example, they show that AMs are particularly well suited for tasks on extreme values for one or more attributes, which is also supported by Task 1. Generally, insights related to attributes were much more common in the adjacency matrix, and insights related to topology were slightly more common in the node-link diagram. This finding highlights the respective strengths and weaknesses of the techniques. A discussion on implications for designing network visualizations can be found in the supplement of Nobre et al. [4].

7.6.1 Methodology Considerations

Creating an evaluation methodology for comparing complex interactive systems with nonexpert users resulted in a unique set of challenges and considerations that we outline and reflect on. First, we had to ensure participants were properly trained on the interactive techniques. We conducted two rounds of pilots to test and improve our training materials. Our first pilot revealed that the passive training videos plus trials were insufficient to recall all available interactions, particularly for the adjacency matrix. As a result, we implemented active training, which required users to perform all available interactions correctly before starting the study. Results from the second pilot were much improved, as indicated by both user performance and participant feedback.

Another challenge is the long study duration when compared to typical crowdsourcing tasks. We found that since we offered above-average hourly compensation and clearly specified the duration of the study, participants were willing to invest the longer time to complete the study. We recruited all 322 participants who completed the study in under two hours. Four hundred twenty-three people started the study but returned it before completion, usually within a few minutes. Feedback by participants indicates that our study was more involved than others on Prolific.

We logged all interactions with the system in order to track engagement and collect a rich dataset on how users leverage the features in each visualization to perform the tasks. We mostly used these data to verify the quality of each trial and to identify and reason about outliers. For example, we found that exceptionally long task completion time

often correlates with participants browsing away from the tab showing the experiment. An initial analysis of these data reveals visualization-specific interaction patterns: For example, participants highlight neighbors more often in the NL, suggesting that neighbors are easier to identify in AM without highlighting. We leave a detailed analysis of these data to future work.

7.6.2 Limitations

Our comparison of two complex systems makes it impossible to identify individual factors that contribute to the performance of the techniques. For example, we do not know whether it is interactive sorting or the encoding in aligned bars that leads to attribute-related insights in the AM.

One limitation of our study is that we were not able to compare multiple different network types in terms of size and density. Instead, we focused on testing a broad set of tasks on networks with varying attribute configurations. Although we cannot make claims as to whether our results would generalize to networks with significantly different topological characteristics, we based our decision to test two size variants of a single network on prior studies that investigated the effect of network size on task accuracy. The results of Ghoniem et al. [23] indicate that for networks under 100 nodes, there was no effect between size and user accuracy for all tasks except for overview tasks such as estimating the number of nodes or links in a network, where node-link diagrams performed better for small networks. For density, Ghoniem et al. [23] found that AM outperforms NL for dense networks in all tasks with the exception of path-finding, where NL representations were always better. By choosing a sparse network for this study, we were able to attribute performance differences between AM and NL to the controlled network attribute variations, and not to the inherent advantage of the AM over NL for dense networks.

7.6.3 Reflections

We believe that our study is unique in terms of the complexity of the techniques and the tasks that we evaluate. Reflecting on our approach, we ask whether such a study that cannot adequately separate individual factors is worth the effort. Aren't more qualitative approaches, like case studies, a better approach for these cases? We believe that a quantitative approach is appropriate because it can give definitive answers to questions about

the relative merits of two widely used interactive visualization techniques, and that it is not feasible to isolate all factors and test them individually. We believe that our results are impervious to subtle changes in the visualization or interaction design. e.g. we believe that if the embedded bar charts in the NL conditions were replaced with an alternative, equivalently powerful design, our results would still hold.

Second, should this method be the new gold standard for evaluating a novel technique? Absolutely not. It is important to use this approach only for techniques that have passed the test of time, and for which the design process is free of biases. We believe, however, that a broad class of published visualization techniques is amenable to our approach.

Reflecting on our validation process, we believe that the expert interviews helped us to design better techniques, yet we were frustrated by the low return rate and by our inability to have a discussion with the experts. We were largely unsuccessful framing our questions so that they were answered in the context of that technique. We believe that a dialog, potentially in a series of structured interviews, would have been more fruitful.

Regarding the use of crowdsourcing platforms for evaluating complex techniques, we found that users achieved remarkably high average accuracy (75.2% across both conditions). We attribute this to: 1) our extensive training and 2) our above average compensation, which we speculate motivated participants to pay attention and be willing to invest time. We have evidence for the former in the form of qualitative feedback on the study, where participants complimented the training. The latter is demonstrated by our high rates of useful comments on the open-explore tasks (80% of all answers contained insights). Given these data, we believe that the view of the visualization community on which kinds of studies can be run on a crowdsourcing platform might be too narrow, and that a broader range of systems is amenable to crowdsourced experiments.

7.7 Conclusion

Multivariate networks are a common and important data type. We present the first quantitative study of two complex, validated visualization techniques tailored to multivariate networks. These results serve to investigate and confirm our understanding of MVNs techniques and their strengths and weaknesses, as presented in Chapter 4. Our results reveal that the AM outperforms the NL for tasks on clusters as well as configurations

where nontask-essential attributes are encoded. NL are bested suited for paths and tasks that rely on within-node comparisons. We also reflect on our approach of using advanced visualization designs in quantitative studies. We highlight the importance of including active training, as well as above average compensation to ensure motivated users for a longer than average study.

CHAPTER 8

DISCUSSION AND CONCLUSION

In this chapter, we first reflect on the major challenges with MVN visualizations including how the work in this dissertation contributes to addressing them. We then summarize the contributions of the dissertation and conclude by describing interesting areas for future research in MVN visualization.

8.1 Discussion

As illustrated in the work presented, MVNs are common in the real world, particularly to represent domain knowledge in various fields. As a result of this ubiquity, research that focuses on improved ways of visualizing these networks can be very beneficial to both real world applications and other research areas.

However, the wealth and variety of information that can be stored in MVNs brings with it several challenges for visualization researchers. Among the main challenges is the sheer number and variety of different attributes that can be associated with each node and edge. Visualizations that properly capture these complex attributes must also take into account the limits of human perceptual and cognitive abilities. As a result, it becomes impossible to clearly show all the information at once in a way that is still meaningful and useful to the user. Careful design decisions that take into account the user's tasks as well as optimized encodings for the network are key to an effective visualization approach.

The complexity of real world MVNs is often accompanied by equally complex analysis tasks. Tasks often require knowledge of the topology of the network as well as one or more attributes of the nodes or edges at the same time. In order to support these complex tasks, MVN visualizations must be carefully designed and support both topology- and attribute-based analysis tasks. Prior knowledge of the specific types of tasks a target audience will perform allows for further customization of encodings to support those tasks, as well as the selection of the appropriate visualization technique.

Aside from the complexity of the network and associated tasks, MVNs, like general graphs, are often very large in real world applications. Scalability limitations that general graph visualizations contend with are further exacerbated with MVNs since additional space is required to encode the node and edge attributes. In order to support these large MVNs, a visualization system must allow for interactivity. With interaction, users can reduce the graph to a meaningful subset through filtering and querying, as well as selectively display details on demand for entities and attributes of interest.

Multivariate networks in the real world are generally visualized as node-link diagrams or variants thereof. The visualization research community has proposed alternate visualization approaches that can more effectively support different MVN analysis tasks, but their efficiency relies on users understanding the encodings used. The adjacency matrix, for example, is gaining popularity and familiarity outside of the visualization community but is still not immediately intuitive to nonexperts. As a result, aside from the challenges described thus far, in order for MVN visualization techniques to be effectively useful in the real world, users must first learn how to use them properly. Training material on novel visualizations is key to achieving this goal.

The work in this dissertation addresses the above challenges in several ways. Our survey of MVNV techniques and operations gives a broad overview of how current MVNV approaches address these challenges, as well as areas that are ripe for future development.

The two technique contributions address the specific challenges outlined above in ways that support their target users and tasks. The techniques are well suited for tasks that require a close integration of topology and attributes. For attribute- or topology- only tasks, users can interact exclusively with the view that encodes that information, such as the table of attributes. For tasks that require knowledge of both attributes and topology, such as finding all the neighbors of a node that have a certain attribute value, users can leverage the integration between views to carry out the task.

We address the multitude of attributes associated to the nodes and edges in two ways: 1) allowing users to select which and how many attributes to display at once and 2) using icons to represent domain relevant attributes such as node types in *Juniper*, or a person's gender in *Lineage*. These icons can also be thought of as "glyphs" for categorical values. Some categorical values — such as male/female — can easily be encoded with glyphs since

there are strong conventions for them, but others might be less immediately recognizable. Ultimately, this approach can be very effective in a system where the types of nodes and edges being visualized are predetermined.

To address the issue of scalability, Lineage takes an aggregation approach to the problem, where users can collapse entire branches of the family tree in order to optimize space usage for individuals of interest to the analysis at hand. Juniper takes a query-first approach to this problem, allowing the user to query the underlying graph to find and display only nodes of interest.

The importance of user familiarity with a given visualization technique became apparent to us in the work done for our user study. In order to assess how well the node-link diagram or the adjacency matrix supported a given task, we had to first ensure that our participants understood the visualizations and the ways in which they could interact with them. Although our users were probably on the extreme end of the novice spectrum as far as network visualization goes, our findings regarding the importance of training likely translate to domain specialists, who might be more familiar with a node-link diagram but may not have experience using other representations.

8.2 Summary

This dissertation provides a comprehensive treatment of visualization techniques for multivariate networks. We offer contributions along several dimensions, including a taxonomy of MVNV tasks and typology of MVNV techniques, two MVN visualization techniques, and a user study providing the first set of empirical evidence on the performance of two MVNV approaches.

Our **survey of MVNV techniques** [3] covered over 200 papers from 1991 to 2018, giving us a broad view of the different approaches to one of the main challenges of visualizing MVNs: supporting analysis of both topology and attributes simultaneously. We organized the design space of solutions to this problem into a typology of techniques, which will provide guidance to future practitioners visualizing MVNs. This survey also served to point out limitations of existing techniques, and areas ripe for future research, which we describe in Section 8.3. We also address some of these limitations in the two technical contributions of this dissertation, Lineage and Juniper.

Lineage is a novel visual analysis tool designed to support domain experts who study multifactorial diseases in the context of genealogies. Incorporating familial relationships between cases with other data can provide insights into shared genomic variants and shared environmental exposures that may be implicated in such diseases. We introduce a data and task abstraction, and argue that the problem of analyzing such diseases based on genealogical, clinical, and genetic data can be mapped to a multivariate graph visualization problem. The main contribution of this design study is a novel visual representation for tree-like, multivariate graphs, which we apply to genealogies and clinical data about the individuals in these families. We introduce data-driven aggregation methods to scale to multiple families. By designing the genealogy graph layout to align with a tabular view, we are able to incorporate extensive, multivariate attributes in the analysis of the genealogy without cluttering the graph. We validate our designs by conducting case studies with our domain collaborators.

With **Juniper**, we introduce a novel, scalable, tree+table multivariate graph visualization technique, which makes many tasks related to multivariate graph analysis easier to achieve. The core principle we follow is to selectively query for nodes or subgraphs of interest and visualize these subgraphs as a spanning tree of the graph. The tree is laid out linearly, which enables us to juxtapose the nodes with a table visualization where diverse attributes can be shown. We also use this table as an adjacency matrix, so that the resulting technique is a hybrid node-link/adjacency matrix technique. We implement this concept in Juniper and complement it with a set of interaction techniques that enable analysts to dynamically grow, restructure, and aggregate the tree, as well as change the layout or show paths between nodes. We demonstrate the utility of our tool in usage scenarios for different multivariate networks: a bipartite network of scholars, papers, and citation metrics and a multitype network of story characters, places, books, etc.

We also contribute an **empirical study** comparing node-link diagrams with on-node encoding and adjacency matrices with juxtaposed tables. We find that node-link diagrams are best suited for tasks that require close integration between the network topology and a few attributes. Adjacency matrices perform well for tasks related to clusters and when many attributes need to be considered. We also reflect on our method of using validated designs for empirically evaluating complex, interactive visualizations in a crowdsourced

setting. We highlight the importance of training, compensation, and provenance tracking.

Ultimately, we hope this dissertation will provide guidance for practitioners and domain experts using MVNs for real world exploration, as well as identify strategies for evaluating complex interactive visualizations.

8.3 Future Work

MVNs are becoming ubiquitous in a broad range of domains, and thus the demand for efficient ways of visualizing them is growing. The use of node-link diagrams to portray networks is by far the most common approach, but the limitation on the number of attributes that can easily be encoded on these structures has led to the popularization of coordinated views and other techniques that are more amenable to encoding multiple attributes, most notably, tabular approaches.

The work in this dissertation highlights areas that are ripe for further development, as well as challenges encountered by current techniques.

8.3.1 Tabular Techniques

Node-link layouts have received plenty of attention, but tabular techniques, such as adjacency matrices, have not been studied in depth in the context of attribute visualization. Tabular approaches have significant potential for visualizing MVNs due to their linear layout of nodes, which allows for trivial juxtaposition of both node and edge attributes.

8.3.2 Edge Attributes

Many solutions are available for visualizing node attributes, but the currently known techniques for visualizing edge attributes are limited. Even matrices, which are considered the best solution for visualizing edge attributes, can show only a handful. Visualizing edge attributes comes with its own set of challenges. Hence, dedicated research in the area is necessary. We identified tabular and integrated approaches as potentially fruitful areas of investigation for edge attributes. BioFabric provides a unique but so far underexplored opportunity for better edge attribute visualization, as edges are represented as columns in BioFabric, which could be leveraged for integration with edge attribute visualizations. Similarly, interactive integrated techniques provide opportunities for better edge attribute visualizations.

8.3.3 Interaction

With the increasing use of web-based systems, interaction has become a ubiquitous element in multivariate network exploration systems, allowing users to quickly and effortlessly filter and analyze the network, as well as modify the encodings used on demand. The study of interaction in MVN visualization techniques is an open opportunity.

8.3.4 User Studies

The field of MVN visualization would benefit from more user studies to rigorously investigate the benefits and trade-offs of different MVN techniques with respect to certain tasks and datasets.

8.3.5 Tools

Hundreds of papers have addressed multivariate network visualization techniques, but the few robust, well-documented, and well-maintained open-source software packages provide only basic multivariate attribute visualization capabilities. The scientific community needs new and better tools that integrate the scientific state of the art in well-maintained software.

REFERENCES

[1] C. Nobre, N. Gehlenborg, H. Coon, and A. Lex, "Lineage: Visualizing multivariate clinical data in genealogy graphs," *Transaction on Visualization and Computer Graphics*, vol. 25, no. 3, pp. 1543–1558, 2019.

[2] C. Nobre, M. Streit, and A. Lex, "Juniper: A tree+table approach to multivariate graph visualization," *Transaction on Visualization and Computer Graphics (InfoVis '18)*, vol. 25, no. 1, 2019.

[3] C. Nobre, M. Meyer, M. Streit, and A. Lex, "The state of the art in visualizing multivariate networks," *Computer Graphics Forum (EuroVis)*, vol. 38, no. 3, pp. 807–832, 2019.

[4] C. Nobre, "Evaluating multivariate network visualization techniques using a validated design and crowdsourcing approach," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI), to Appear*, 2020.

[5] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. van Wijk, J.-D. Fekete, and D. Fellner, "Visual analysis of large graphs: State-of-the-art and future research challenges," *Computer Graphics Forum*, vol. 30, no. 6, pp. 1719–1749, 2011.

[6] X. F. Wang and G. Chen, "Complex networks: Small-world, scale-free and beyond," *IEEE Circuits and Systems Magazine*, vol. 3, no. 1, pp. 6–20, 2003.

[7] E. Kruja, J. Marks, A. Blair, and R. Waters, "A short note on the history of graph drawing," in *Graph Drawing*, G. Goos, J. Hartmanis, J. van Leeuwen, P. Mutzel, M. Jünger, and S. Leipert, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, vol. 2265, pp. 272–286.

[8] K. Dinkla, M. Westenberg, and J. van Wijk, "Compressed adjacency matrices: Untangling gene regulatory networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2457–2466, 2012.

[9] C. Klapisch-Zuber, "The genesis of the family tree," *I Tatti Studies in the Italian Renaissance*, vol. 4, pp. 105–129, 1991.

[10] M. Lima, *The Book of Trees: Visualizing Branches of Knowledge*. New York: Princeton Architectural Press, 2014.

[11] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

[12] H. Neuweger, M. Persicke, S. P. Albaum, T. Bekel, M. Dondrup, A. T. Hüser, J. Winnebald, J. Schneider, J. Kalinowski, and A. Goesmann, "Visualizing post genomics data-sets on customized pathway maps by ProMeTra – aeration-dependent gene

expression and metabolism of Corynebacterium glutamicum as an example," *BMC Systems Biology*, vol. 3, no. 1, p. 82, 2009.

[13] J. Heer and D. Boyd, "Vizster: Visualizing online social networks," in *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005*, 2005, pp. 32–39.

[14] N. Gehlenborg, S. I. O'Donoghue, N. S. Baliga, A. Goesmann, M. A. Hibbs, H. Kitano, O. Kohlbacher, H. Neuweger, R. Schneider, D. Tenenbaum, and A.-C. Gavin, "Visualization of omics data for systems biology," *Nature Methods*, vol. 7, no. 3, pp. 56–68, 2010.

[15] M. E. Smoot, K. Ono, J. Ruscheinski, P.-L. Wang, and T. Ideker, "Cytoscape 2.8: New features for data integration and network visualization," *Bioinformatics*, vol. 27, no. 3, pp. 431–432, Feb. 2011.

[16] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *Third International AAAI Conference on Weblogs and Social Media*, 2009.

[17] S. van den Elzen and J. J. van Wijk, "Multivariate network exploration and presentation: From detail to overview via selections and aggregations," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '14)*, vol. 20, no. 12, pp. 2310–2319, 2014.

[18] T. J. Jankun-Kelly and K.-L. Ma, "MoireGraphs: Radial focus+context visualization and interaction for graphs with visual nodes," in *IEEE Symposium on Information Visualization 2003 (IEEE Cat. No.03TH8714)*, 2003, pp. 59–66.

[19] L. Wilkinson and M. Friendly, "The history of the cluster heat map," *The American Statistician*, vol. 63, no. 2, pp. 179–184, 2009.

[20] J. Bertin, *La Graphique et Le Traitement Graphique de l'information*. Nouvelle bibliotheque scientifique. Flammarion., 1975.

[21] C. Perin, P. Dragicevic, and J.-D. Fekete, "Revisiting bertin matrices: New interactions for crafting tabular visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2082–2091, Dec. 2014.

[22] J. Bertin, "Sémiologie graphique: Les diagrammes-les réseaux-les cartes," https://infoscience.epfl.ch/record/51376, 1973.

[23] M. Ghoniem, J.-D. Fekete, and P. Castagliola, "On the readability of graphs using node-link and matrix-based representations: A controlled experiment and statistical analysis," *Information Visualization*, vol. 4, no. 2, pp. 114–135, Jun. 2005.

[24] N. Henry, J. D. Fekete, and M. J. McGuffin, "NodeTrix: A hybrid visualization of social networks," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '07)*, vol. 13, no. 6, pp. 1302–1309, 2007.

[25] N. Henry and J.-D. Fekete, "Matlink: Enhanced matrix visualization for analyzing social networks," in *Human-Computer Interaction – INTERACT 2007*, ser. Lecture Notes in Computer Science, C. Baranauskas, P. Palanque, J. Abascal, and S. D. J. Barbosa, Eds. Springer Berlin Heidelberg, 2007, no. 4663, pp. 288–302.

[26] N. Elmqvist, T.-N. Do, H. Goodell, N. Henry, and J. Fekete, "ZAME: Interactive large-scale graph visualization," in *Visualization Symposium, 2008. PacificVIS '08. IEEE Pacific*, 2008, pp. 215–222.

[27] E. Kerzner, A. Lex, C. Sigulinsky, T. Urness, B. Jones, R. Marc, and M. Meyer, "Graffinity: Visualizing connectivity in large graphs," *Computer Graphics Forum*, vol. 36, no. 3, pp. 251–260, Jun. 2017.

[28] Y. Yang, T. Dwyer, S. Goodwin, and K. Marriott, "Many-to-many geographically-embedded flow visualisation: An evaluation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 411–420, 2017.

[29] M. Okoe, R. Jianu, and S. G. Kobourov, "Node-link or adjacency matrices: Old question, new insights," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2018.

[30] S. K. Card and D. Nation, "Degree-of-interest trees: A component of an attention-reactive user interface," in *Proceedings of the Working Conference on Advanced Visual Interfaces*, ser. AVI '02. New York, NY, USA: ACM, 2002, pp. 231–245.

[31] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proceedings of the National Academy of Sciences USA*, vol. 95, no. 25, pp. 14 863–14 868, 1998.

[32] J. Seo and B. Shneiderman, "Interactively exploring hierarchical clustering results," *Computer*, vol. 35, no. 7, pp. 80–86, 2002.

[33] A. Lex, M. Streit, C. Partl, K. Kashofer, and D. Schmalstieg, "Comparative analysis of multidimensional, quantitative data," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '10)*, vol. 16, no. 6, pp. 1027–1035, 2010.

[34] B. Lee, L. Nachmanson, G. Robertson, J. M. Carlson, and D. Heckerman, "PhyloDet: A scalable visualization tool for mapping multiple traits to large evolutionary trees," *Bioinformatics*, vol. 25, no. 19, pp. 2611–2612, 2009.

[35] Ł. Kreft, A. Botzki, F. Coppens, K. Vandepoele, and M. Van Bel, "PhyD3: A phylogenetic tree viewer with extended phyloXML support for functional genomics data visualization," *Bioinformatics*, vol. 33, no. 18, pp. 2946–2947, Sep. 2017.

[36] M. Burch, F. Beck, and S. Diehl, "Timeline trees: Visualizing sequences of transactions in information hierarchies," in *Proceedings of the Working Conference on Advanced Visual Interfaces*, ser. AVI '08. New York, NY, USA: ACM, 2008, pp. 75–82.

[37] B. Johnson and B. Shneiderman, "Tree-maps: A space-filling approach to the visualization of hierarchical information structures," in *Proceedings of the IEEE Conference on Visualization (Vis '91)*, 1991, pp. 284–291.

[38] J. Stasko and E. Zhang, "Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations," in *Proceedings of the IEEE Symposium on Information Vizualization (InfoVis '00)*. IEEE Computer Society Press, 2000, pp. 57–65.

[39] J. B. Kruskal and J. M. Landwehr, "Icicle plots: Better displays for hierarchical clustering," *The American Statistician*, vol. 37, no. 2, p. 162, 1983.

[40] M. Wattenberg, "Visualizing the stock market," in *CHI '99 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '99. Pittsburgh, Pennsylvania: Association for Computing Machinery, May 1999, pp. 188–189.

[41] K. Andrews and H. Heidegger, "Information slices: Visualising and exploring large hierarchies using cascading, semicircular discs." in *InfoVis'98: Proc. IEEE Information Visualization Symposium, Carolina, USA*, 1998, pp. 9–12.

[42] H.-J. Schulz, "Treevis.net: A tree visualization reference," *IEEE Computer Graphics and Applications*, vol. 31, no. 6, pp. 11–15, 2011.

[43] N. McCurdy, J. Lein, K. Coles, and M. Meyer, "Poemage: Visualizing the sonic topology of a poem," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 439–448, 2016.

[44] C. Nobre and A. Lex, "OceanPaths: Visualizing multivariate oceanography data," in *Eurographics Conference on Visualization (EuroVis '15) - Short Papers*. The Eurographics Association, 2015.

[45] A. J. Pretorius and J. J. van Wijk, "Visual inspection of multivariate graphs," *Computer Graphics Forum (EuroVis '08)*, vol. 27, no. 3, pp. 967–974, 2008.

[46] M. Sedlmair, P. Isenberg, D. Baur, M. Mauerer, C. Pigorsch, and A. Butz, "Cardiogram: Visual analytics for automotive engineers," in *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems - CHI '11*. Vancouver, BC, Canada: ACM Press, 2011, p. 1727.

[47] A. Kerren, H. C. Purchase, and M. Ward, Eds., *Multivariate Network Visualization*, ser. Lecture Notes in Computer Science. New York: Springer, 2014, no. 8380.

[48] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*. Cambridge, UK: Cambridge University Press, 1994.

[49] J. Scott, *Social Network Analysis*. SAGE, 2017.

[50] N. Henry and J.-D. Fekete, "MatrixExplorer: A dual-representation system to explore social networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 677–684, 2006.

[51] F. van Ham, H.-J. Schulz, and J. M. Dimicco, "Honeycomb: Visual analysis of large scale social networks," in *Human-Computer Interaction – INTERACT 2009*, vol. 5727. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 429–442.

[52] J. S. Yi, N. Elmqvist, and S. Lee, "TimeMatrix: Analyzing temporal social networks using interactive matrix-based visualizations," *International Journal of Human-Computer Interaction*, vol. 26, no. 11-12, pp. 1031–1051, 2010.

[53] F. B. Viégas and J. Donath, "Social network visualization: Can we go beyond the graph?" in *Workshop on Social Networks, CSCW*, vol. 4, 2004, pp. 6–10.

[54] A. Bezerianos, F. Chevalier, P. Dragicevic, N. Elmqvist, and J. D. Fekete, "Graphdice: A system for exploring multivariate social networks," *Computer Graphics Forum (EuroVis '10)*, vol. 29, no. 3, pp. 863–872, 2010.

[55] S. Ghani, B. C. Kwon, S. Lee, J. S. Yi, and N. Elmqvist, "Visual analytics for multimodal social network analysis: A design study with social scientists," *IEEE Transactions on Visualization and Computer Graphics (VAST '13)*, vol. 19, no. 12, pp. 2032–2041, 2013.

[56] S. Oeltze, W. Freiler, R. Hillert, H. Doleisch, B. Preim, and W. Schubert, "Interactive, graph-based visual analysis of high-dimensional, multi-parameter fluorescence microscopy data in toponomics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 1882–1891, 2011.

[57] A. Lex, C. Partl, D. Kalkofen, M. Streit, S. Gratzl, A. M. Wassermann, D. Schmalstieg, and H. Pfister, "Entourage: Visualizing relationships between biological pathways using contextual subsets," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '13)*, vol. 19, no. 12, pp. 2536–2545, 2013.

[58] C. Partl, A. Lex, M. Streit, D. Kalkofen, K. Kashofer, and D. Schmalstieg, "enRoute: Dynamic path extraction from biological pathway maps for in-depth experimental data analysis," in *Proceedings of the IEEE Symposium on Biological Data Visualization*, ser. BioVis '12, 2012, pp. 107–114.

[59] F. Schreiber, T. Dwyer, K. Marriott, and M. Wybrow, "A generic algorithm for layout of biological networks," *BMC Bioinformatics*, vol. 10, no. 1, p. 375, 2009.

[60] P. D. Karp and S. M. Paley, "Automated drawing of metabolic pathways," in *Proceedings of the 3rd International Conference on Bioinformatics and Genome Research*, 1994, pp. 225–238.

[61] M. Meyer, B. Wong, M. Styczynski, T. Munzner, and H. Pfister, "Pathline: A Tool For Comparative Functional Genomics," *Computer Graphics Forum (EuroVis '10)*, vol. 29, no. 3, pp. 1043–1052, 2010.

[62] C. Partl, S. Gratzl, M. Streit, A. M. Wassermann, H. Pfister, D. Schmalstieg, and A. Lex, "Pathfinder: Visual analysis of paths in graphs," *Computer Graphics Forum (EuroVis '16)*, vol. 35, no. 3, pp. 71–80, 2016.

[63] A. Barsky, T. Munzner, J. Gardy, and R. Kincaid, "Cerebral: Visualizing multiple experimental conditions on a graph with biological context," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '08)*, vol. 14, no. 6, pp. 1253–1260, 2008.

[64] J. New, W. Kendall, J. Huang, and E. Chesler, "Dynamic visualization of coexpression in systems genetics data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 5, pp. 1081–1095, 2008.

[65] S. Diehl, *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Berlin Heidelberg: Springer-Verlag, 2007.

[66] S. Diehl and A. C. Telea, "Multivariate networks in software engineering," in *Multivariate Network Visualization*, ser. Lecture Notes in Computer Science, A. Kerren, H. C.

Purchase, and M. O. Ward, Eds. New York: Springer International Publishing, 2014, no. 8380, pp. 13–36.

[67] I. Jusufi, Y. Dingjie, and A. Kerren, "The network lens: Interactive exploration of multivariate networks using visual filtering," in *Proceedings of the Conference on Information Visualisation*, 2010, pp. 35–42.

[68] D. Sun and K. Wong, "On evaluating the layout of UML class diagrams for program comprehension," in *13th International Workshop on Program Comprehension (IWPC'05)*, 2005, pp. 317–326.

[69] D. Reniers, L. Voinea, O. Ersoy, and A. Telea, "The Solid* toolset for software visual analytics of program structure and metrics comprehension: From research prototype to product," *Science of Computer Programming*, vol. 79, pp. 224–240, 2014.

[70] S. Eick, J. Steffen, and J. Sumner, E.E., "Seesoft-a tool for visualizing line oriented software statistics," *IEEE Transactions on Software Engineering*, vol. 18, no. 11, pp. 957–968, 1992.

[71] H. Carr, S. Czanner (editors, J. Trümper, A. Telea, and J. Döllner, "Viewfusion: Correlating structure and activity views for execution traces," in *Theory and Practice of Computer Graphics*, 2012.

[72] F. Beck and S. Diehl, "Visual comparison of software architectures," *Information Visualization*, vol. 12, no. 2, pp. 178–199, Apr. 2013.

[73] H. Byelas and A. Telea, "Visualizing multivariate attributes on software diagrams," in *2009 13th European Conference on Software Maintenance and Reengineering*, 2009, pp. 335–338.

[74] R. Wettel and M. Lanza, "Visualizing software systems as cities," in *4th Ieee International Workshop on Visualizing Software for Understanding and Analysis*. Banff, Ontario: Society Press, 2007, pp. 92–99.

[75] A. Telea and D. Auber, "Code flows: Visualizing structural evolution of source code," *Computer Graphics Forum (EuroVis '08)*, vol. 27, no. 3, pp. 831–838, 2008.

[76] C. Hurter, B. Tissoires, and S. Conversy, "Fromdady: Spreading aircraft trajectories across views to support iterative queries," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 15, no. 6, pp. 1017–1024, 2009.

[77] V. Peysakhovich, C. Hurter, and A. Telea, "Attribute-driven edge bundling for general graphs with applications in trail analysis," in *2015 IEEE Pacific Visualization Symposium (PacificVis)*. Hangzhou, China: IEEE, 2015, pp. 39–46.

[78] A. Wolff, "Drawing subway maps: A survey," *Informatik - Forschung und Entwicklung*, vol. 22, pp. 23–44, 2007.

[79] W. Zeng, C. Fu, S. M. Arisona, A. Erath, and H. Qu, "Visualizing mobility of public transportation system," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1833–1842, 2014.

[80] R. Ball, G. A. Fink, and C. North, "Home-centric visualization of network traffic for security administration," in *In VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization And*. ACM Press, 2004, pp. 55–64.

[81] J. Oberheide, M. Goff, and M. Karir, "Flamingo: Visualizing internet traffic," in *2006 IEEE/IFIP Network Operations and Management Symposium NOMS 2006*, 2006, pp. 150–161.

[82] R. A. Becker, S. G. Eick, and A. R. Wilks, "Visualizing network data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 1, pp. 16–28, 1995.

[83] S. T. Eick, "Aspects of network visualization," *IEEE Computer Graphics and Applications*, vol. 16, no. 2, pp. 69–72, 1996.

[84] H. Shiravi, A. Shiravi, and A. A. Ghorbani, "A survey of visualization systems for network security," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 8, pp. 1313–1329, 2012.

[85] J. Pearlman and P. Rheingans, "Visualizing network security events using compound glyphs from a service-oriented perspective," in *VizSEC*, 2007.

[86] F. Mansman, L. Meier, and D. A. Keim, "Visualization of host behavior for network security," in *VizSEC 2007*, J. R. Goodall, G. Conti, and K.-L. Ma, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 187–202.

[87] X. Yin, W. Yurcik, M. Treaster, Y. Li, and K. Lakkaraju, "VisFlowConnect: Netflow visualizations of link relationships for security situational awareness," in *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security - VizSEC/DMSEC '04*. Washington DC, USA: ACM Press, 2004, p. 26.

[88] H. Koike, K. Ohno, and K. Koizumi, "Visualizing cyber attacks using IP matrix," in *IEEE Workshop on Visualization for Computer Security, 2005. (VizSEC 05).*, 2005, pp. 91–98.

[89] B. Lee, C. Parr, B. Bederson, V. Veksler, W. Gray, C. Kotfila, and C. Kotfila, "Treeplus: Interactive exploration of networks with enhanced tree layouts," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 6, pp. 1414–1426, 2006.

[90] M. Jern, J. Rogstadius, and T. Åström, "Treemaps and choropleth maps applied to regional hierarchical statistical data," in *2009 13th International Conference Information Visualisation*. Barcelona, Spain: IEEE, 2009, pp. 403–410.

[91] B. Alper, N. Riche, G. Ramos, and M. Czerwinski, "Design study of linesets, a novel set visualization technique," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '11)*, vol. 17, no. 12, pp. 2259–2267, 2011.

[92] S. Schöffel, J. Schwank, and A. Ebert, "A user study on multivariate edge visualizations for graph-based visual analysis tasks," in *2016 20th International Conference Information Visualisation (IV)*, 2016, pp. 165–170.

[93] J. Bae and B. Watson, "Developing and evaluating quilts for the depiction of large layered graphs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2268–2275, 2011.

[94] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry, "Task taxonomy for graph visualization," in *Proceedings of the AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization (BELIV '06)*, 2006, pp. 1–5.

[95] N. A. Christakis and J. H. Fowler, "The spread of obesity in a large social network over 32 years," *New England Journal of Medicine*, vol. 357, no. 4, pp. 370–379, Jul. 2007.

[96] M. Brehmer and T. Munzner, "A multi-level typology of abstract visualization tasks," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '13)*, vol. 19, no. 12, pp. 2376–2385, 2013.

[97] H.-J. Schulz, T. Nocke, M. Heitzler, and H. Schumann, "A design space of visualization tasks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2366–2375, 2013.

[98] E. R. A. Valiati, M. S. Pimenta, and C. M. D. S. Freitas, "A taxonomy of tasks for guiding the evaluation of multidimensional visualizations," in *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors Novel Evaluation Methods for Information Visualization - BELIV '06*. Venice, Italy: ACM Press, 2006, p. 1.

[99] B. Shneiderman and A. Aris, "Network visualization by semantic substrates," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 733–740, 2006.

[100] A. J. Pretorius, H. C. Purchase, and J. T. Stasko, "Tasks for multivariate network analysis," in *Multivariate Network Visualization*, ser. Lecture Notes in Computer Science, A. Kerren, H. C. Purchase, and M. O. Ward, Eds. New York: Springer International Publishing, Jan. 2014, no. 8380, pp. 77–95.

[101] Y. Tu and H. W. Shen, "GraphCharter: Combining browsing with query to explore large semantic graphs," in *2013 IEEE Pacific Visualization Symposium (PacificVis)*, 2013, pp. 49–56.

[102] F. van Ham and A. Perer, ""Search, show context, expand on demand": Supporting large graph exploration with degree-of-interest," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 953–960, 2009.

[103] H.-J. Schulz, S. Hadlak, and H. Schumann, "The design space of implicit hierarchy visualization: A survey," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 4, pp. 393–411, 2011.

[104] C. Vehlow, F. Beck, and D. Weiskopf, "The state of the art in visualizing group structures in graphs," *Eurographics Conference on Visualization (EuroVis) - STARs*, 2015.

[105] F. Beck, M. Burch, S. Diehl, and D. Weiskopf, "The state of the art in visualizing dynamic graphs," in *Proceedings of the Eurographics Conference on Visualization (EuroVis '14) – State of The Art Reports*, 2014.

[106] A. Lhuillier, C. Hurter, and A. Telea, "State of the art in edge and trail bundling techniques," *Computer Graphics Forum*, vol. 36, no. 3, pp. 619–645, 2017.

[107] S. Hadlak, H. Schumann, and H.-J. Schulz, "A survey of multi-faceted graph visualization," in *Eurographics Conference on Visualization (EuroVis) - STARs*, R. Borgo, F. Ganovelli, and I. Viola, Eds. The Eurographics Association, 2015.

[108] VERBI Software, "MAXQDA Analytics Pro," Berlin: GERMANY, 2018.

[109] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale, "Seven guiding scenarios for information visualization evaluation," University of Calgary, Tech. Rep., 2011.

[110] J. S. Olson and W. A. Kellogg, *Ways of Knowing in HCI*. New York: Springer Publishing Company, Incorporated, May 2014.

[111] M. Wattenberg, "Visual exploration of multivariate graphs," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '06. New York, NY, USA: ACM, 2006, pp. 811–819.

[112] C. Viau, M. J. McGuffin, Y. Chiricota, and I. Jurisica, "The flowvizmenu and parallel scatterplot matrix: Hybrid multidimensional visualizations for network exploration," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '10)*, vol. 16, no. 6, pp. 1100–1108, 2010.

[113] M. Krzywinski, J. Schein, I. Birol, J. Connors, R. Gascoyne, D. Horsman, S. J. Jones, and M. A. Marra, "Circos: An information aesthetic for comparative genomics," *Genome Research*, vol. 19, no. 9, pp. 1639–1645, 2009.

[114] H. Schulz and H. Schumann, "Visualizing graphs - a generalized view," in *Tenth International Conference on Information Visualisation (IV'06)*, 2006, pp. 166–173.

[115] D. Auber, Y. Chiricota, F. Jourdan, and G. Melancon, "Multiscale visualization of small world networks," in *IEEE Symposium on Information Visualization 2003 (IEEE Cat. No.03TH8714)*. Seattle, WA, USA: IEEE, 2003, pp. 75–81.

[116] C. Dunne and B. Shneiderman, "Motif simplification: Improving network visualization readability with fan, connector, and clique glyphs," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*. Paris, France: ACM Press, 2013, p. 3247.

[117] S. Ghani and N. Elmqvist, "Improving revisitation in graphs through static spatial features," in *Proceedings of Graphics Interface 2011*. Canadian Human-Computer Communications Society, 2011, pp. 175–182.

[118] M. A. Westenberg, S. A. F. T. van Hijum, O. P. Kuipers, and J. B. T. M. Roerdink, "Visualizing genome expression and regulatory network dynamics in genomic and metabolic context," *Computer Graphics Forum*, vol. 27, no. 3, pp. 887–894, 2008.

[119] B. H. Junker, C. Klukas, and F. Schreiber, "VANTED: A system for advanced data analysis and visualization in the context of biological networks," *BMC Bioinformatics*, vol. 7, no. 1, p. 109, 2006.

[120] Y. Katz, E. T. Wang, J. Silterra, S. Schwartz, B. Wong, J. P. Mesirov, E. M. Airoldi, and C. B. Burge, "Sashimi plots: Quantitative visualization of RNA sequencing read alignments," *arXiv:1306.3466 [q-bio]*, 2013.

[121] D. Guo, "Flow mapping and multivariate visualization of large spatial interaction data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1041–1048, 2009.

[122] K. Xu, C. Rooney, P. Passmore, D. Ham, and P. H. Nguyen, "A user study on curved edges in graph visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2449–2456, 2012.

[123] J. Schwank, S. Schöffel, J. Stärz, and A. Ebert, "Visualizing uncertainty of edge attributes in node-link diagrams," in *2016 20th International Conference Information Visualisation (IV)*, Jul. 2016, pp. 45–50.

[124] C. B. Nielsen, S. D. Jackman, I. Birol, and S. J. M. Jones, "ABySS-Explorer: Visualizing genome sequence assemblies," *IEEE transactions on visualization and computer graphics*, vol. 15, no. 6, pp. 881–888, 2009.

[125] S. Ko, S. Afzal, S. Walton, Y. Yang, J. Chae, A. Malik, Y. Jang, M. Chen, and D. Ebert, "Analyzing high-dimensional multivaríate network links with integrated anomaly detection, highlighting and exploration," in *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2014, pp. 83–92.

[126] D. Holten, P. Isenberg, J. J. van Wijk, and J. Fekete, "An extended evaluation of the readability of tapered, animated, and textured directed-edge representations in node-link graphs," in *2011 IEEE Pacific Visualization Symposium (PacificVis)*, 2011, pp. 195–202.

[127] E. M. Rodrigues, N. Milic-Frayling, M. Smith, B. Shneiderman, and D. Hansen, "Group-in-a-box layout for multi-faceted analysis of communities," in *2011 IEEE Third Int'l Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third Int'l Conference on Social Computing*. Boston, MA, USA: IEEE, 2011, pp. 354–361.

[128] H.-J. Schulz, M. John, A. Unger, and H. Schumann, "Visual analysis of bipartite biological networks," in *Proceedings of the Eurographics Workshop on Visual Computing for Biomedicine (VCBM '08)*, 2008, pp. 135–142.

[129] C. Partl, A. Lex, M. Streit, H. Strobelt, A. M. Wassermann, H. Pfister, and D. Schmalstieg, "ConTour: Data-driven exploration of multi-relational datasets for drug discovery," *IEEE Transactions on Visualization and Computer Graphics (VAST '14)*, vol. 20, no. 12, pp. 1883–1892, 2014.

[130] A. Ahmed, V. Batagelj, X. Fu, S. Hong, D. Merrick, and A. Mrvar, "Visualisation and analysis of the internet movie database," in *2007 6th International Asia-Pacific Symposium on Visualization*. Sydney, NSW, Australia: IEEE, 2007, pp. 17–24.

[131] A. Aris and B. Shneiderman, "Designing semantic substrates for visual network exploration," *Information Visualization*, vol. 6, no. 4, pp. 281–300, 2007.

[132] M. Meyer, T. Munzner, and H. Pfister, "MizBee: A multiscale synteny browser," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '09)*, vol. 15, no. 6, pp. 897–904, 2009.

[133] M. Dörk, S. Carpendale, and C. Williamson, "Edgemaps: Visualizing explicit and implicit relations," in *IS&T/SPIE Electronic Imaging*, San Francisco Airport, California, USA, 2011, p. 78680G.

[134] E. Bonabeau, "Graph multidimensional scaling with self-organizing maps," *Information Sciences*, vol. 143, no. 1-4, pp. 159–180, 2002.

[135] D. Holten, "Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '06)*, vol. 12, no. 5, pp. 741–748, 2006.

[136] D. Holten and J. J. van Wijk, "Force-directed edge bundling for graph visualization," *Computer Graphics Forum (EuroVis '09)*, vol. 28, no. 3, pp. 983–990, 2009.

[137] J. F. Kruiger, P. E. Rauber, R. M. Martins, A. Kerren, S. Kobourov, and A. C. Telea, "Graph layouts by t-SNE," *Computer Graphics Forum*, vol. 36, no. 3, pp. 283–294, 2017.

[138] A. Bezerianos, P. Dragicevic, J. Fekete, J. Bae, and B. Watson, "Geneaquilts: A system for exploring large genealogies," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1073–1081, 2010.

[139] W. J. Longabaugh, "Combing the hairball with BioFabric: A new approach for visualization of large networks," *BMC Bioinformatics*, vol. 13, no. 1, p. 275, Oct. 2012.

[140] M. Okoe, R. Jianu, and S. Kobourov, "Revisited network representations," *25th Symposium on Graph Drawing*, p. 16, 2017.

[141] S. Rufiange, M. J. McGuffin, and C. P. Fuhrman, "TreeMatrix: A hybrid visualization of compound graphs," *Computer Graphics Forum*, vol. 31, no. 1, pp. 89–101, 2012.

[142] C. Dunne, N. Henry Riche, B. Lee, R. Metoyer, and G. Robertson, "GraphTrail: Analyzing large multivariate, heterogeneous networks while supporting exploration history," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, 2012, pp. 1663–1672.

[143] B. Alper, B. Bach, N. Henry Riche, T. Isenberg, and J.-D. Fekete, "Weighted graph comparison techniques for brain connectivity analysis," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*. Paris, France: ACM Press, 2013, p. 483.

[144] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola, "Fast optimal leaf ordering for hierarchical clustering," *Bioinformatics*, vol. 17, no. Suppl 1, pp. S22–S29, 2001.

[145] J. Díaz, J. Petit, and M. Serna, "A survey of graph layout problems," *ACM Comput. Surv.*, vol. 34, no. 3, pp. 313–356, 2002.

[146] I. Liiv, "Seriation and matrix reordering methods: An historical overview," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 3, no. 2, pp. 70–91, 2010.

[147] C. Mueller, B. Martin, and A. Lumsdaine, "A comparison of vertex ordering algorithms for large graph visualization," in *2007 6th International Asia-Pacific Symposium on Visualization*, 2007, pp. 141–148.

[148] J.-D. Fekete, "Reorder.js: A Javascript library to reorder tables and networks," in *IEEE Symposium on Information Visualization*, Chicago, United States, 2015, p. 3.

[149] M. Behrisch, B. Bach, N. Henry Riche, T. Schreck, and J.-D. Fekete, "Matrix reordering methods for table and network visualization," *Computer Graphics Forum*, vol. 35, no. 3, pp. 693–716, 2016.

[150] Z. Shen and K.-L. Maz, "Path visualization for adjacency matrices," in *Proceedings of the 9th Joint Eurographics / IEEE VGTC Conference on Visualization*, ser. EUROVIS'07. Eurographics Association, 2007, pp. 83–90.

[151] B. Watson, D. Brink, M. Stallman, R. Devajaran, T.-M. Rhyne, and H. Patel, "Visualizing very large layered graphs with quilts," North Carolina State University. Dept. of Computer Science., Tech. Rep., 2008.

[152] J. J. van Wijk and H. van de Wetering, "Cushion treemaps: Visualization of hierarchical information," in *In Proceedings of the IEEE Symposium on Information Visualization (InfoVis99)*, 1999, pp. 73–78.

[153] F. Beck, F.-J. Wiszniewsky, M. Burch, S. Diehl, and D. Weiskopf, "Asymmetric visual hierarchy comparison with nested icicle plots," *Proceedings of the Workshop on Graph Visualization in Practice (GraphVIP '14)*, 2014.

[154] A. Slingsby, J. Dykes, and J. Wood, "Configuring hierarchical layouts to address research questions," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 977–984, 2009.

[155] M. Balzer and O. Deussen, "Voronoi treemaps," in *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, 2005, pp. 49–56.

[156] M. Bruls, K. Huizing, and J. J. van Wijk, "Squarified treemaps," in *Data Visualization 2000*, ser. Eurographics, W. C. de Leeuw and R. van Liere, Eds. New York: Springer, 2000, pp. 33–42.

[157] Y. Tu and H. Shen, "Visualizing changes of hierarchical data using treemaps," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1286–1293, 2007.

[158] J. D. Fekete and C. Plaisant, "Interactive information visualization of a million items," in *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '02)*, 2002, pp. 117–124.

[159] J. C. Roberts, "Multiple-view and multiform visualization," in *Proceedings of the SPIE Conference on Visualization and Data Analysis (VDA '02)*, vol. 3960. SPIE, 2000, pp. 176–185.

[160] M. Q. Wang Baldonado, A. Woodruff, and A. Kuchinsky, "Guidelines for using multiple views in information visualization," in *Proceedings of the Working Conference on Advanced Visual Interfaces - AVI '00*. Palermo, Italy: ACM Press, 2000, pp. 110–119.

[161] R. Shannon, T. Holland, and A. Quigley, "Multivariate graph drawing using parallel coordinate visualisations," University of St Andrews, Tech. Rep., 2008.

[162] A. Lex, M. Streit, E. Kruijff, and D. Schmalstieg, "Caleydo: Design and evaluation of a visual analysis framework for gene expression data in its biological context," in *Proceedings of the IEEE Symposium on Pacific Visualization (PacificVis '10)*. IEEE, 2010, pp. 57–64.

[163] W. Javed and N. Elmqvist, "Exploring the design space of composite visualization," in *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis '12)*. IEEE, 28 2012-march 2, pp. 1–8.

[164] J. Abello and F. van Ham, "Matrix zoom: A visual interface to semi-external graphs," in *IEEE Symposium on Information Visualization*, 2004, pp. 183–190.

[165] J. Stasko, C. Görg, and Z. Liu, "Jigsaw: Supporting investigative analysis through interactive visualization," *Information Visualization*, vol. 7, no. 2, pp. 118–132, 2008.

[166] C. Plaisant, B. Shneiderman, and R. Mushlin, "An information architecture to support the visualization of personal histories," *Information Processing & Management*, vol. 34, no. 5, pp. 581–597, 1998.

[167] R. Pienta, F. Hohman, A. Endert, A. Tamersoy, K. Roundy, C. Gates, S. Navathe, and D. H. Chau, "VIGOR: Interactive visual exploration of graph query results," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 215–225, 2018.

[168] C. Partl, A. Lex, M. Streit, D. Kalkofen, K. Kashofer, and D. Schmalstieg, "enRoute: Dynamic path extraction from biological pathway maps for exploring heterogeneous experimental datasets," *BMC Bioinformatics*, vol. 14, no. Suppl 19, p. S3, 2013.

[169] B. Lee, L. Nachmanson, G. Robertson, J. Carlson, and D. Heckerman, "Det. (distance encoded tree): A scalable visualization tool for mapping multiple traits to large evolutionary trees," Microsoft Research, Tech. Rep., 2008.

[170] D. Naquin, Y. d'Aubenton-Carafa, C. Thermes, and M. Silvain, "CIRCUS: A package for Circos display of structural genome variations from paired-end and mate-pair sequencing data," *BMC Bioinformatics*, vol. 15, no. 1, 2014.

[171] C. Collins, G. Penn, and S. Carpendale, "Bubble sets: Revealing set relations with isocontours over existing visualizations," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '09)*, vol. 15, no. 6, pp. 1009–1016, 2009.

[172] E. R. Gansner, Y. Hu, and S. Kobourov, "GMap: Visualizing graphs and clusters as maps," in *2010 IEEE Pacific Visualization Symposium (PacificVis)*. Taipei, Taiwan: IEEE, 2010, pp. 201–208.

[173] B. Saket, P. Simonetto, and S. Kobourov, "Group-level graph visualization taxonomy," *arXiv:1403.7421 [cs]*, 2014.

[174] A. Lex, M. Streit, H.-J. Schulz, C. Partl, D. Schmalstieg, P. J. Park, and N. Gehlenborg, "StratomeX: Enabling visualization-driven cancer subtype analysis," in *Poster Compendium of the IEEE Symposium on Biological Data Visualization (BioVis '12)*, 2012.

[175] S. van den Elzen and J. J. van Wijk, "Small multiples, large singles: A new approach for visual data exploration," *Computer Graphics Forum (EuroVis '13)*, vol. 32, no. 3pt2, pp. 191–200, 2013.

[176] B. Bach, E. Pietriga, and J.-D. Fekete, "Visualizing dynamic networks with matrix cubes," in *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems - CHI '14*. Toronto, Ontario, Canada: ACM Press, 2014, pp. 877–886.

[177] J.-D. Fekete, D. Wang, N. Dang, A. Aris, and C. Plaisant, "Interactive poster: Overlaying graph links on treemaps," in *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '03)*. IEEE, 2003, pp. 82–83.

[178] D. Archambault, T. Munzner, and D. Auber, "Grouseflocks: Steerable exploration of graph hierarchy space," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 4, pp. 900–913, Jul. 2008.

[179] S. Zhao, M. McGuffin, and M. Chignell, "Elastic hierarchies: Combining treemaps and node-link diagrams," in *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '05)*. Washington (DC): IEEE Computer Society Press, 2005, pp. 57–64.

[180] N. Henry, A. Bezerianos, and J. Fekete, "Improving the readability of clustered social networks using node duplication," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1317–1324, 2008.

[181] A. Bigelow, C. Nobre, M. Meyer, and A. Lex, "Origraph: Interactive network wrangling," in *Proceedings of IEEE Conference on Visual Analytics Science and Technology (VAST '19)*. IEEE, 2019.

[182] P.-Y. Koenig, F. Zaidi, and D. Archambault, "Interactive searching and visualization of patterns in attributed graphs," in *Proceedings of Graphics Interface 2010*. Canadian Information Processing Society, 2010, pp. 113–120.

[183] G. W. Furnas, "Generalized fisheye views," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '86)*. ACM, 1986, pp. 16–23.

[184] J. Heer and S. K. Card, "DOITrees revisited: Scalable, space-constrained visualization of hierarchical data," in *Proceedings of the Working Conference on Advanced Visual Interfaces*, ser. AVI '04. New York, NY, USA: ACM, 2004, pp. 421–424.

[185] I. Jusufi, A. Kerren, and B. Zimmer, "Multivariate network exploration with JauntyNets," in *2013 17th International Conference on Information Visualisation*, 2013, pp. 19–27.

[186] E. S. Mills, "Genealogy in the information age: History's new frontier," *National Genealogical Society Quarterly*, vol. 91, no. 91, pp. 260–277, 2003.

[187] R. L. Bennett, K. S. French, R. G. Resta, and D. L. Doyle, "Standardized human pedigree nomenclature: Update and assessment of the recommendations of the national society of genetic counselors," *Journal of Genetic Counseling*, vol. 17, no. 5, pp. 424–433, 2008.

[188] J. Xu, K. Kochanek, S. Murphy, and B. Tejada-Vera, "National vital statistics reports. deaths: Final data for 2007," *Center for Disease Control and Prevention Division of Vital Statistics*, vol. 58, 2010.

[189] National Center for Health Statistics (US), *Health, United States, 2015: With Special Feature on Racial and Ethnic Health Disparities*, ser. Health, United States. National Center for Health Statistics (US), 2016.

[190] N. L. Pedersen and A. Fiske, "Genetic influences on suicide and nonfatal suicidal behavior: Twin study findings," *European Psychiatry*, vol. 25, no. 5, pp. 264–267, 2010.

[191] P. McGuffin, A. Marušič, and A. Farmer, "What can psychiatric genetics offer suicidology?" *Crisis: The Journal of Crisis Intervention and Suicide Prevention*, vol. 22, no. 2, pp. 61–65, 2001.

[192] M. Sokolowski, J. Wasserman, and D. Wasserman, "Polygenic associations of neurodevelopmental genes in suicide attempt," *Molecular Psychiatry*, 2015.

[193] W. Katon, "Asthma, suicide risk, and psychiatric comorbidity," *American Journal of Psychiatry*, vol. 167, no. 9, pp. 1020–1022, 2010.

[194] A. V. Bakian, R. S. Huber, H. Coon, D. Gray, P. Wilson, W. M. McMahon, and P. F. Renshaw, "Acute air pollution exposure and risk of suicide completion," *American Journal of Epidemiology*, vol. 181, no. 5, pp. 295–303, 2015.

[195] M. Sedlmair, M. Meyer, and T. Munzner, "Design study methodology: Reflections from the trenches and the stacks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2431–2440, Dec. 2012.

[196] S. Goodwin, J. Dykes, S. Jones, I. Dillingham, G. Dove, A. Duffy, A. Kachkaev, A. Slingsby, and J. Wood, "Creative user-centered visualization design for energy analysts and modelers," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2516–2525, 2013.

[197] Progeny Genetics LLC, "Progeny," 2016.

[198] R. L. Bennett, K. A. Steinhaus, S. B. Uhrich, C. K. O'Sullivan, R. G. Resta, D. Lochner-Doyle, D. S. Markel, V. Vincent, and J. Hamanishi, "Recommendations for standardized human pedigree nomenclature. pedigree standardization task force of the national society of genetic counselors." *American Journal of Human Genetics*, vol. 56, no. 3, pp. 745–752, 1995.

[199] C. Fuchsberger, S. Miksch, L. Forer, and C. Pattaro, "Analyzing populations with visual and analytical methods to identify family clustered diseases," *Proceedings of the 12th World Congress on Health (Medical) Informatics; Building Sustainable Health Systems*, p. 2243, 2007.

[200] V.-P. Mäkinen, M. Parkkonen, M. Wessman, P.-H. Groop, T. Kanninen, and K. Kaski, "High-throughput pedigree drawing," *European Journal of Human Genetics*, vol. 13, no. 8, pp. 987–989, May 2005.

[201] J. C. Barrett, B. Fry, J. Maller, and M. J. Daly, "Haploview: Analysis and visualization of LD and haplotype maps," *Bioinformatics*, vol. 21, no. 2, pp. 263–265, 2005.

[202] R. E. Voorrips, M. C. A. M. Bink, V. D. Weg, and W. Eric, "Pedimap: Software for the visualization of genetic and phenotypic data in pedigrees," *Journal of Heredity*, vol. 103, no. 6, pp. 903–907, 2012.

[203] H. Thiele and P. Nürnberg, "HaploPainter: A tool for drawing pedigrees with complex haplotypes," *Bioinformatics*, vol. 21, no. 8, pp. 1730–1732, 2005.

[204] M. J. McGuffin and R. Balakrishnan, "Interactive visualization of genealogical graphs," in *Proceedings of the IEEE Symposium on Information Visualization*, ser. InfoVis '05, 2005, pp. 16–23.

[205] G. M. Draper and R. F. Riesenfeld, "Interactive fan charts: A space-saving technique for genealogical graph exploration," in *Proceedings of the 8th Annual Workshop on Technology for Family History and Genealogical Research (FHTW 2008)*, 2008.

[206] A. Mazeika, J. Petersons, and M. H. Böhlen, "PPPA: Push and pull pedigree analyzer for large and complex pedigree databases," in *Advances in Databases and Information Systems*, 2006, pp. 339–352.

[207] C. Tuttle, L. G. Nonato, and C. Silva, "PedVis: A structured, space-efficient technique for pedigree visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1063–1072, 2010.

[208] R. Ball, "Visualizing genealogy through a family-centric perspective," *Information Visualization*, vol. 16, no. 1, pp. 74–89, 2017.

[209] N. W. Kim, S. K. Card, and J. Heer, "Tracing genealogical data with TimeNets," in *Proceedings of the International Conference on Advanced Visual Interfaces*, ser. AVI '10. ACM, 2010, pp. 241–248.

[210] S. Fu, H. Dong, W. Cui, J. Zhao, and H. Qu, "How do ancestral traits shape family trees over generations?" *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 205–214, 2018.

[211] Y. Liu, S. Dai, C. Wang, Z. Zhou, and H. Qu, "GenealogyVis: A system for visual analysis of multidimensional genealogical data," *IEEE Transactions on Human-Machine Systems*, vol. PP, no. 99, pp. 1–13, 2017.

[212] B. Cannon, M. Hiremath, C. Jorcyk, and A. Joshi, "Cove: A colony visualization system for animal pedigrees," in *Proceedings of the 7th International Symposium on Visual Information Communication and Interaction*, ser. VINCI '14. ACM, 2014, pp. 9:9–9:18.

[213] T. Paterson, M. Graham, J. Kennedy, and A. Law, "VIPER: A visualisation tool for exploring inheritance inconsistencies in genotyped pedigrees," *BMC Bioinformatics*, vol. 13, no. 8, p. S5, 2012.

[214] P. D. Shaw, M. Graham, J. Kennedy, I. Milne, and D. F. Marshall, "Helium: Visualization of large scale plant pedigrees," *BMC Bioinformatics*, vol. 15, no. 1, p. 259, Aug. 2014.

[215] M. Glueck, A. Gvozdik, F. Chevalier, A. Khan, M. Brudno, and D. Wigdor, "Phenostacks: Cross-sectional cohort phenotype comparison visualizations," *IEEE Transactions on Visualization and Computer Graphics (VAST '16)*, vol. 23, no. 1, pp. 191–200, 2017.

[216] R. Adrian, "Tree drawing algorithms," in *Handbook of Graph Drawing and Visualization*. CRC Press, 2013, pp. 155–192.

[217] G. W. Furnas, "A fisheye follow-up: Further reflections on focus + context," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '06. ACM, 2006, pp. 999–1008.

[218] C. Ware, H. Purchase, L. Colpoys, and M. McGill, "Cognitive measurements of graph aesthetics," *Information Visualization*, vol. 1, no. 2, pp. 103–110, 2002.

[219] S. Gratzl, A. Lex, N. Gehlenborg, H. Pfister, and M. Streit, "Lineup: Visual analysis of multi-attribute rankings," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '13)*, vol. 19, no. 12, pp. 2277–2286, 2013.

[220] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale, "Empirical studies in information visualization: Seven scenarios," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 9, pp. 1520–1536, Sep. 2012.

[221] R. A. Kerber, "Method for calculating risk associated with family history of a disease," *Genetic Epidemiology*, vol. 12, no. 3, pp. 291–301, 1995.

[222] H. Coon, T. M. Darlington, W. B. Callor, E. Ferris, A. Fraser, Z. Yu, N. William, S. C. Das, S. E. Crowell, M. Puzia, M. Klein, A. Docherty, L. Jerominski, D. S. Cannon, K. R. Smith, B. Keeshin, A. V. Bakian, E. Christensen, N. J. Camp, and D. Gray, "Identification of genome-wide significant shared genomic segments in large extended Utah families at high risk for completed suicide," *bioRxiv*, p. 195644, 2017.

[223] R. G. Waller, T. M. Darlington, X. Wei, M. Madsen, A. Thomas, K. Curtin, H. Coon, V. Rajamanickam, J. Musinsky, D. Jayabalan, D. Atanackovic, V. Rajkumar, S. Kumar, S. Slager, M. Middha, P. Galia, D. Demangel, M. Salama, V. Joseph, J. McKay, K. Offit, R. J. Klein, S. M. Lipkin, C. Dumontet, C. M. Vachon, and N. J. Camp, "Novel pedigree analysis implicates dna repair and chromatin remodeling in multiple myeloma risk," *bioRxiv*, p. 137000, Sep. 2017.

[224] S. Knight, R. P. Abo, H. J. Abel, D. W. Neklason, T. M. Tuohy, R. W. Burt, A. Thomas, and N. J. Camp, "Shared genomic segment analysis: The power to find rare disease variants," *Annals of Human Genetics*, vol. 76, no. 6, pp. 500–509, 2012.

[225] C. R. Marshall, W. Wu, D. S. Greer, D. Antaki, A. Shetty, P. A. Holmans, D. Pinto, M. Gujral, W. M. Brandler, D. Malhotra, Z. Wang, K. V. F. Fajarado, M. S. Maile, S. Ripke, I. Agartz, M. Albus, M. Alexander, F. Amin, J. Atkins, S. A. Bacanu, R. A. B. Jr, S. E. Bergen, M. Bertalan, E. Bevilacqua, T. B. Bigdeli, D. W. Black, R. Bruggeman, N. G. Buccola, R. L. Buckner, B. Bulik-Sullivan, W. Byerley, W. Cahn, G. Cai, M. J. Cairns, D. Campion, R. M. Cantor, V. J. Carr, N. Carrera, S. V. Catts, K. D. Chambert, W. Cheng, C. R. Cloninger, D. Cohen, P. Cormican, N. Craddock, B. Crespo-Facorro, J. J. Crowley, D. Curtis, M. Davidson, K. L. Davis, F. Degenhardt, J. D. Favero, L. E. DeLisi, D. Dikeos, T. Dinan, S. Djurovic, G. Donohoe, E. Drapeau, J. Duan, F. Dudbridge, P. Eichhammer, J. Eriksson, V. Escott-Price, L. Essioux, A. H. Fanous, K.-H. Farh, M. S. Farrell, J. Frank, L. Franke, R. Freedman, N. B. Freimer, J. I. Friedman, A. J. Forstner, M. Fromer, G. Genovese, L. Georgieva, E. S. Gershon, I. Giegling, P. Giusti-Rodríguez, S. Godard, J. I. Goldstein, J. Gratten, L. de Haan, M. L. Hamshere, M. Hansen, T. Hansen, V. Haroutunian, A. M. Hartmann, F. A. Henskens, S. Herms, J. N. Hirschhorn, P. Hoffmann, A. Hofman, H. Huang, M. Ikeda, I. Joa, A. K. Kähler, R. S. Kahn, L. Kalaydjieva, J. Karjalainen, D. Kavanagh, M. C. Keller, B. J. Kelly, J. L. Kennedy, Y. Kim, J. A. Knowles, B. Konte, C. Laurent, P. Lee, S. H. Lee, S. E. Legge, B. Lerer, D. L. Levy, K.-Y. Liang, J. Lieberman, J. Lönnqvist, C. M. Loughland,

P. K. E. Magnusson, B. S. Maher, W. Maier, J. Mallet, M. Mattheisen, M. Mattingsdal, R. W. McCarley, C. McDonald, A. M. McIntosh, S. Meier, C. J. Meijer, I. Melle, R. I. Mesholam-Gately, A. Metspalu, P. T. Michie, L. Milani, V. Milanova, Y. Mokrab, D. W. Morris, B. Müller-Myhsok, K. C. Murphy, R. M. Murray, I. Myin-Germeys, I. Nenadic, D. A. Nertney, G. Nestadt, K. K. Nicodemus, L. Nisenbaum, A. Nordin, E. O'Callaghan, C. O'Dushlaine, S.-Y. Oh, A. Olincy, L. Olsen, F. A. O'Neill, J. V. Os, C. Pantelis, G. N. Papadimitriou, E. Parkhomenko, M. T. Pato, T. Paunio, P. E. I. Consortium, D. O. Perkins, T. H. Pers, O. Pietiläinen, J. Pimm, A. J. Pocklington, J. Powell, A. Price, A. E. Pulver, S. M. Purcell, D. Quested, H. B. Rasmussen, A. Reichenberg, M. A. Reimers, A. L. Richards, J. L. Roffman, P. Roussos, D. M. Ruderfer, V. Salomaa, A. R. Sanders, A. Savitz, U. Schall, T. G. Schulze, S. G. Schwab, E. M. Scolnick, R. J. Scott, L. J. Seidman, J. Shi, J. M. Silverman, J. W. Smoller, E. Söderman, C. C. A. Spencer, E. A. Stahl, E. Strengman, J. Strohmaier, T. S. Stroup, J. Suvisaari, D. M. Svrakic, J. P. Szatkiewicz, S. Thirumalai, P. A. Tooney, J. Veijola, P. M. Visscher, J. Waddington, D. Walsh, B. T. Webb, M. Weiser, D. B. Wildenauer, N. M. Williams, S. Williams, S. H. Witt, A. R. Wolen, B. K. Wormley, N. R. Wray, J. Q. Wu, C. C. Zai, R. Adolfsson, O. A. Andreassen, D. H. R. Blackwood, E. Bramon, J. D. Buxbaum, S. Cichon, D. A. Collier, A. Corvin, M. J. Daly, A. Darvasi, E. Domenici, T. Esko, P. V. Gejman, M. Gill, H. Gurling, C. M. Hultman, N. Iwata, A. V. Jablensky, E. G. Jönsson, K. S. Kendler, G. Kirov, J. Knight, D. F. Levinson, Q. S. Li, S. A. McCarroll, A. McQuillin, J. L. Moran, B. J. Mowry, M. M. Nöthen, R. A. Ophoff, M. J. Owen, A. Palotie, C. N. Pato, T. L. Petryshen, D. Posthuma, M. Rietschel, B. P. Riley, D. Rujescu, P. Sklar, D. S. Clair, J. T. R. Walters, T. Werge, P. F. Sullivan, M. C. O'Donovan, S. W. Scherer, B. M. Neale, J. Sebat, and C. a. S. W. G. o. t. P. G. Consortium, "Contribution of copy number variants to schizophrenia from a genome-wide study of 41,321 subjects," *Nature Genetics*, vol. 49, no. 1, p. 27, 2017.

[226] E. T. Monson, K. de Klerk, S. C. Gaynor, A. H. Wagner, M. E. Breen, M. Parsons, T. L. Casavant, P. P. Zandi, J. B. Potash, and V. L. Willour, "Whole-gene sequencing investigation of SAT1 in attempted suicide," *American Journal of Medical Genetics Part B: Neuropsychiatric Genetics*, vol. 171, no. 6, pp. 888–895, 2016.

[227] R. A. Neher and T. Bedford, "Nextflu: Real-time tracking of seasonal influenza virus evolution in humans," *Bioinformatics*, vol. 31, no. 21, pp. 3546–3548, 2015.

[228] T. A. Manolio, F. S. Collins, N. J. Cox, D. B. Goldstein, L. A. Hindorff, D. J. Hunter, M. I. McCarthy, E. M. Ramos, L. R. Cardon, A. Chakravarti, J. H. Cho, A. E. Guttmacher, A. Kong, L. Kruglyak, E. Mardis, C. N. Rotimi, M. Slatkin, D. Valle, A. S. Whittemore, M. Boehnke, A. G. Clark, E. E. Eichler, G. Gibson, J. L. Haines, T. F. C. Mackay, S. A. McCarroll, and P. M. Visscher, "Finding the missing heritability of complex diseases," *Nature*, vol. 461, no. 7265, pp. 747–753, 2009.

[229] T. Munzner, "Drawing large graphs with H3Viewer and site manager," in *Graph Drawing*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Aug. 1998, pp. 384–393.

[230] M. C. Hao, M. Hsu, U. Dayal, and A. Krug, "Web-based visualization of large hierarchical graphs using invisible links in a hyperbolic space," in *Advances in Visual Information Management*, ser. IFIP — The International Federation for Information Processing. Springer, Boston, MA, 2000, pp. 83–94.

[231] P. Eklund, N. Roberts, and S. Green, "OntoRama: Browsing RDF ontologies using a hyperbolic-style browser," in *First International Symposium on Cyber Worlds, 2002. Proceedings.*, 2002, pp. 405–411.

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

[233] B. Lee, K. Brown, Y. Hathout, and J. Seo, "GOTreePlus: An interactive gene ontology browser," *Bioinformatics*, vol. 24, no. 7, pp. 1026–1028, Apr. 2008.

[234] L. Gou and X. Zhang, "Treenetviz: Revealing patterns of networks over tree structures," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '11)*, vol. 17, no. 12, pp. 2449–2458, 2011.

[235] S. Kairam, N. H. Riche, S. Drucker, R. Fernandez, and J. Heer, "Refinery: Visual exploration of large, heterogeneous networks through associative browsing," *Computer Graphics Forum*, vol. 34, no. 3, pp. 301–310, Jun. 2015.

[236] K. Furmanova, S. Gratzl, H. Stitz, T. Zichner, M. Jaresova, M. Ennemoser, A. Lex, and M. Streit, "Taggle: Scalable visualization of tabular data through aggregation," *Information Visualization*, 2019.

[237] P. Isenberg, F. Heimerl, S. Koch, T. Isenberg, P. Xu, C. D. Stolper, M. Sedlmair, J. Chen, T. Möller, and J. Stasko, "Vispubdata.org: A metadata collection about IEEE visualization (VIS) publications," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 9, pp. 2199–2206, Sep. 2017.

[238] S. Greenberg and B. Buxton, "Usability evaluation considered harmful (some of the time)," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '08. New York, NY, USA: ACM, 2008, pp. 111–120.

[239] S. van den Elzen, D. Holten, J. Blaas, and J. van Wijk, "Reducing snapshots to points: A visual analytics approach to dynamic network exploration," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 1–10, Jan. 2016.

[240] A. Lex, N. Gehlenborg, H. Strobelt, R. Vuillemot, and H. Pfister, "UpSet: Visualization of intersecting sets," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '14)*, vol. 20, no. 12, pp. 1983–1992, 2014.

[241] R. Kosara, "An empire built on sand: Reexamining what we think we know about visualization," in *Proceedings of the Sixth Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization*, ser. BELIV '16. Baltimore, MD, USA: ACM, 2016, pp. 162–168.

[242] P. Saraiya, C. North, and K. Duca, "An insight-based methodology for evaluating bioinformatics visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 4, pp. 443–456, 2005.

[243] C. Plaisant, J.-D. Fekete, and G. Grinstein, "Promoting insight-based evaluation of visualizations: From contest to benchmark repository," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 1, pp. 120–134, 2008.

[244] S. Carpendale, "Evaluating information visualizations," in *Information Visualization: Human-Centered Issues and Perspectives*, ser. Lecture Notes in Computer Science, A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 19–45.

[245] V. Yoghourdjian, D. Archambault, S. Diehl, T. Dwyer, K. Klein, H. C. Purchase, and H.-Y. Wu, "Exploring the limits of complexity: A survey of empirical studies on graph visualisation," *Visual Informatics*, vol. 2, no. 4, pp. 264–282, 2018.

[246] R. Keller, C. M. Eckert, and P. J. Clarkson, "Matrices or node-link diagrams: Which visual representation is better for visualising connectivity models?" *Information Visualization*, vol. 5, no. 1, pp. 62–76, 2006.

[247] C. Chang, B. Bach, T. Dwyer, and K. Marriott, "Evaluating perceptually complementary views for network exploration tasks," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM, 2017, pp. 1397–1407.

[248] D. Ren, L. R. Marusich, J. O'Donovan, J. Z. Bakdash, J. A. Schaffer, D. N. Cassenti, S. E. Kase, H. E. Roy, W.-y. S. Lin, and T. Höllerer, "Understanding node-link and matrix visualizations of networks: A large-scale online experiment," *Network Science*, vol. 7, no. 2, pp. 242–264, 2019.

[249] J. Christensen, J. H. Bae, B. Watson, and M. Rappa, "Understanding which graph depictions are best for viewers," in *Smart Graphics*, ser. Lecture Notes in Computer Science, M. Christie and T.-Y. Li, Eds. New York: Springer, 2014, pp. 174–177.

[250] A. Abuthawabeh, F. Beck, D. Zeckzer, and S. Diehl, "Finding structures in multi-type code couplings with node-link and matrix visualizations," in *Working Conference on Software Visualization (VISSOFT)*. IEEE, 2013, pp. 1–10.

[251] S. Schöffel, J. Schwank, J. Stärz, and A. Ebert, "Multivariate networks: A novel edge visualization approach for graph-based visual analysis tasks," in *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '16. ACM, 2016, pp. 2292–2298.

[252] U. Gadiraju, S. Möller, M. Nöllenburg, D. Saupe, S. Egger-Lampl, D. Archambault, and B. Fisher, "Crowdsourcing versus the laboratory: Towards human-centered experiments using the crowd," in *Evaluation in the Crowd. Crowdsourcing and Human-Centered Experiments*, D. Archambault, H. Purchase, and T. Hoßfeld, Eds. Cham: Springer International Publishing, 2017, vol. 10264, pp. 6–26.

[253] D. L. Alonso, A. Rose, C. Plaisant, and K. L. Norman, "Viewing personal history records: A comparison of tabular format and graphical presentation using LifeLines," *Behaviour & Information Technology*, vol. 17, no. 5, pp. 249–262, Jan. 1998.

[254] M. Tory, D. Sprague, F. Wu, W. Y. So, and T. Munzner, "Spatialization design: Comparing points and landscapes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1262–1269, Nov. 2007.

[255] J. Heer and G. G. Robertson, "Animated transitions in statistical data graphics," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '07)*, vol. 13, no. 6, pp. 1240–1247, 2007.

[256] J. Heer and M. Bostock, "Crowdsourcing graphical perception: Using mechanical turk to assess visualization design," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10. ACM, 2010, pp. 203–212.

[257] M. Feng, C. Deng, E. M. Peck, and L. Harrison, "The effects of adding search functionality to interactive visualizations on the web," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. Montreal QC, Canada: ACM Press, 2018, pp. 1–13.

[258] M. Okoe, R. Jianu, S. Kobourov, R. Jianu, and S. Kobourov, "Revisited experimental comparison of node-link and matrix representations," in *Graph Drawing and Network Visualization*. Springer, 2018, vol. 10692, pp. 287–302.

[259] E. Wall, M. Agnihotri, L. Matzen, K. Divis, M. Haass, A. Endert, and J. Stasko, "A heuristic approach to value-driven evaluation of visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 491–500, Jan. 2019.

[260] T. Munzner, *Visualization Analysis and Design*. Boca Raton: CRC Press, Taylor & Francis Group, 2014.

[261] W. Huang, P. Eades, S.-H. Hong, and H. B.-L. Duh, "Effects of curves on graph perception," in *Pacific Visualization Symposium (PacificVis)*. IEEE, 2016, pp. 199–203.

[262] R. Rao and S. K. Card, "The table lens: Merging graphical and symbolic representations in an interactive focus + context visualization for tabular information," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM, 1994, pp. 318–322.

[263] I. Safarli and A. Lex, "TaMax: Visualizing dense multivariate networks with adjacency matrices," in *Proceedings of the IEEE Information Visualization Conference (InfoVis Posters)*, 2019.

[264] P. Berger, H. Schumann, and C. Tominski, "Visually exploring relations between structure and attributes in multivariate graphs," in *Information Visualisation (IV)*. Paris, France: IEEE, 2019, pp. 261–268.

[265] P. Dragicevic, "Fair statistical communication in hci," in *Modern Statistical Methods for HCI*, ser. Human–Computer Interaction Series. New York: Springer, 2016, pp. 291–330.

[266] G. Cumming, *Understanding the New Statistics: Effect Sizes, Confidence Intervals, and Meta-Analysis*. Routledge, 2013.