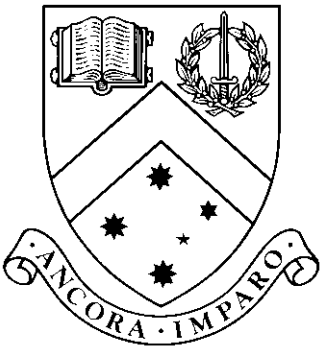
All Diagrams Neat and Beautiful, All Networks Great and Small

by

Vahan Yoghourdjian, BSc., MSc.

Վահան Եողուրդճեան



submitted in fulfillment of the requirements of the degree of

Doctor of Philosophy (0190)

Supervisor: Associate Prof. Tim Dwyer

Associate Supervisors: Prof. Kim Marriott, Dr. Michael Wybrow, and Dr. Karsten Klein

Caulfield School of Information Technology
Monash University

August, 2018

All Diagrams Neat and Beautiful, All Networks Great and Small

Vahan Yoghourdjian, BSc., MSc.
[redacted]
Monash University, 2018

Supervisor: Associate Prof. Tim Dwyer
Associate Supervisors: Prof. Kim Marriott, Dr. Michael Wybrow, and Dr. Karsten Klein

Abstract

Network diagrams are used to visually represent information concerning interconnected objects. These objects could be social (e.g. a group of friends), biological (e.g. a protein-protein interaction system), or belong to any other application area. Diagrams, in general, assist the understanding and analysis of some underlying data, however their visualisation is extremely challenging. The two major challenges of network diagram visualisation are scalability and quality. Most visualisations suffer from the problem of scalability; where the more information they show the harder it becomes to read or understand them. This is a key problem for network diagrams, since many of today's networks consist of thousands or even millions of objects. Quality is another key factor for designing visual representations. A lot of research has been conducted to formalise design principles for visualisations. Similarly, researchers have identified several layout features that enhance the quality of network diagrams. Nonetheless, it is difficult to design network layout models that support even some combinations of these features. Furthermore, these two challenges are intertwined and cannot be solved independently, as the requirements for the layout quality change with the size of the network diagram.

In this thesis we explore the two ends of the network layout problem and aim to achieve visualisations that are both scalable; to very large network data, but are in some sense optimal with respect to the quality of automatically computed layout. We propose an optimal layout model for small networks or smaller parts of large networks. To achieve this, we apply and explore the application of state-of-the-art combinatorial optimisation techniques to the network layout problem to obtain a high-quality detailed view of network data. Our evaluation shows that these techniques can support optimal layouts for networks with less than one hundred nodes in reasonable time. We then explore the threshold that separates small and large networks in terms of cognition capacities. We approach this by conducting a survey on empirical studies, which use node-link diagrams as part of their evaluation, and a user study where we evaluate cognitive load with respect to network size. We discover that most empirical studies use networks with less than a hundred nodes and that cognitive load significantly increases beyond this limit. We also propose a summary representation for large networks. We use decomposition techniques to show connectivity information but hide details of individual nodes and links. Our evaluation shows that the summary evaluation scales to very large networks and allows users to perform overview tasks efficiently.

All Diagrams Neat and Beautiful, All Networks Great and Small

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Vahan Yoghourdjian
August 20, 2018

The title *All Diagrams Neat and Beautiful, All Networks Great and Small* is an epigram inspired by the first two sentences of a hymn from the 19th century [21].

"All things bright and beautiful, All creatures great and small"

The epigram reflects on the work in this thesis, which proposes compact and aesthetically pleasing diagrams for both small and large networks, but is not literal. We are not claiming to have tackled representation challenges for all types of networks, nor cover every beautiful representation.

Contents

List of Tables

List of Figures

Acknowledgments

Առաջին, կ'ուզեմ շնորհակալութիւն յայտնել ընտանիքիս, որ ամենայն հանդուրժողութեամբ ինձի սորվեցուցին, թէ ուսումը կարեւորագոյն սնունդն է մարդ կերտելու: Ծնողքս՝ Թոմարզացի Յարութ Եողուրտճեան եւ Մուսա Լեռնցի Սոնա Չափարեան, ընդառաջեցին ուսումս շարունակելու որոշումիս, Ալեպրալիոյ հեռաւոր ափերուն: Կ'ուզեմ նաեւ շնորհակալութիւն յայտնել մեծ մօրս՝ Մարի Չափարեանին, որ իր անուշ ձայնով յաջողութիւն մաղթեց եւ քաջալերեց ամէն առիթի: Նաեւ երախտագիտութիւնս կ'ուզեմ յայտնել մեր Տէր Ասպուծոյ. Գիտութիւնը ընծայ է մարդկութեան: Յատուկ շնորհակալութիւն՝ եղբօրս՝ Հրակին, քոյրերուս՝ Լալէին եւ Հուրիին եւ մեր անուշիկ իշխանուհիին՝ Սէրլիին:

I would like to thank my supervisors: Tim Dwyer, Karsten Klein, Kim Marriott, and Michael Wybrow for being patient with me and providing a wealth of feedback that assisted me in finishing this thesis. I would also like to thank Helen Purchase for all the work she has brought to the field of network visualisation, and personally, for her feedback and kind words when working together. I am grateful to the administration team at the Caulfield campus. Special thanks to Sue McKemmish, Allison Mitchell, Julie Holden, Sidalavy Chaing, and Denyse Cove. The efforts of Sue McKemmish made my PhD journey pleasant and fulfilling, both academically and socially. Lastly, I would like to thank all the PhD students and Post-Docs at Monash University at the time of my studies. I made great friends and will cherish all our social gatherings.

Vahan Yoghourdjian

Monash University
August 2018

Chapter 1

Introduction

"All things bright and beautiful,
All creatures great and small,
He gave us eyes to see them,
And lips that we might tell,"

Cecil Frances Alexander,
All Things Bright and Beautiful

The first two sentences of the first verse and the first two sentences of the last verse of a song published in *Hymns for Little Children*.

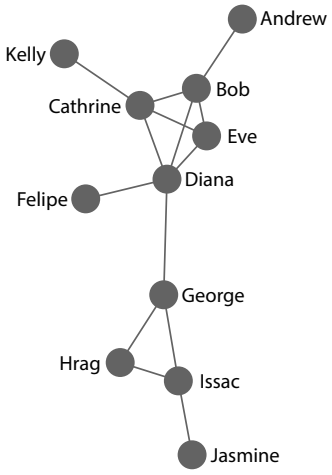
The title of this thesis is inspired by the hymn and changed to reflect the work in this thesis with respect to network visualisation.

With the advances in technology and its accessibility, we are faced with large amounts of data. Visualisation is a powerful tool that can be used to understand and communicate complex and large data. Network visualisation deals with data that is relational. Algorithms that create network diagrams are torn between scalability and quality. We aim to identify a threshold for data complexity, below which we should use high quality layout techniques and beyond which we should use summary representations.

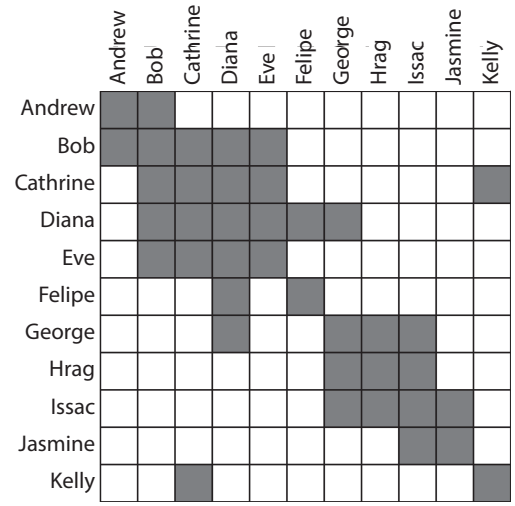
Networks are groups or systems of interconnected things. These interconnected things are often referred to as nodes, and the connections between them are called links.[1] Interesting networks arise in different application areas, such as Biology, Sociology, and Communications. The basic definition of a network is independent of application, but further refinements might occur based on the needs of specific applications. For example, the flow of water in a water distribution network, where the water tanks are nodes and the pipes connecting them are links, necessitates a certain direction on the links. Such networks, in which the nodes are sources or targets, are known as directed networks.

A group of friends is an example of a social network, where the nodes are people and the links between them represent their friendships. Nodes and links can have additional

[1] nodes–vertices and links–edges can be used interchangeably



(a) A node-link representation of a social network. The nodes represent the people and the lines (links) represent their friendship. The names are shown using text labels, which are placed in proximity to their respective nodes.



(b) A social network represented using an adjacency matrix. The people are represented by positions on the horizontal and vertical axes. The intersecting cells between each column and row represents an existing friendship between the person on that column and the person on that row. In this case, if a cell is marked grey then a friendship exists, otherwise it does not.

Figure 1.1: The same small social network of friends is represented with a node-link diagram on the left and a matrix on the right. Both representations show all the people in the network and their relationships.

attributes associated with them. The names of the people in a social network are examples of 'node attributes', while the age of the relationship is an example of a 'link attribute'.

The increase in number of connected devices and the rapid advances in technology, have allowed the gathering of large amounts of data. This data is often relational, which means that it can be represented as a network. For example, the 14 social networks in the Stanford *SNAP* collection have 515,362 nodes and 8,575,643 links on average (median: 79,333 nodes and 677,876 links) [213]. The largest network in their collection represents the members of an on-line community called LiveJournal [8]. The network has almost five million members and around 69 million connections. Understanding and communicating these large networks by text is very hard. Hence, diagrams are used to compress these lengthy chunks of text into neatly informative representations. *Network Visualisation* or *Graph Drawing*, deals with creating diagrams for networks.

Node-link diagrams are one of the most common types of visualisation used to represent networks. Naturally, and as the name suggests, marks of different shapes are used to represent the nodes. These are often circles, squares, or dots, while links are represented by lines, which are drawn between the nodes. Node-link diagrams are one of the oldest ways to visualise networks and, to date, remain the most commonly used. Figure 1.1(a) presents an example of a node-link diagram. This diagram represents a fabricated social network, where the nodes represent people and a link exists between two people if they are friends.

Various visual channels can be used to encode information as properties of marks [233]. Different channels are used to represent different types of information. For example, with colour channels, hue can be used to represent categorical information of nodes (e.g. name of a person), while luminance can be used to represent ordinal information (e.g. age). Further visual channels, such as position, shape, tilt, or size can be used on the marks to show different attributes of nodes and links. For example, nodes of a network may be arranged left to right or top to bottom to show hierarchy in a directed network. It could also be used to show a certain flow, similar to the example of Figure 1.2. In this case, the links have arrowheads, which are shape channels used in order to show direction.

An alternative way to visualise networks is with a tabular, or adjacency matrix, representation. Nodes are represented by the rows and the columns of the matrix. A link exists between the two nodes if the cell at the intersection of the respective row and column is marked. Figure 1.1(b) shows an example of a matrix diagram, which represents the same network as Figure 1.1(a). In this case, the people are represented by the rows and columns. The intersecting cell between a row and a column represents the relationship between the people represented by the respective row and column. In this example, Kelly and Cathrine are friends, since the row representing Cathrine and the column representing Kelly intersect at a grey cell. The presence of a link can be shown using different marks and channels within a cell. For example, the cell may be shaded or marked with glyphs of various shapes. Similarly, link attributes may be indicated by different marks or channels, such as colour-shading or labels. The precise potential benefits of different visual marks and channels is still an area of ongoing research (e.g. [65] [77]).

There are also other techniques to visualise networks, but these usually tend to be variations, or hybrids, of the above two methods. One of the biggest advantages of using adjacency matrices over node-link diagrams is that they do not share the challenges associated with drawing the links, such as crossings and occlusion. Adjacency matrices associate a cell for each connection, thus eliminating the possibility of overlap. Nonetheless, this leads to non-compact visualisations, since a cell is needed for each pair of nodes, whether or not that pair of nodes is actually connected by a link. A matrix is a square with an area of $n \times n$, where n is the number of nodes, thus the size of a matrix grows quadratically with the number of nodes. Node-link diagrams remain the most popular way to visualise network data [233]. The strongest weakness of adjacency matrices is their inability to reflect the topological structure of the underlying network. The direct representation of links in node-link diagrams prove to be more suitable for tasks that require a detailed understanding of the connections in a network (e.g. finding paths), than the indirect representation in adjacency matrices [141].

1.1 Challenges of Network Diagrams

Above, we describe at a very high-level, the most commonly used approaches to network visualisation. We started to hint at the problem of scale also, in that matrices grow in area very quickly as the number of nodes increases, while node-link diagrams become very cluttered and difficult to read when there are many links between nodes.

In this section, we identify four significant challenges (labelled **CH-1–4**) for making clear and insightful network visualisations for graphs of all sizes, link-densities and complexities. We outline them below, with detailed discussion to follow:

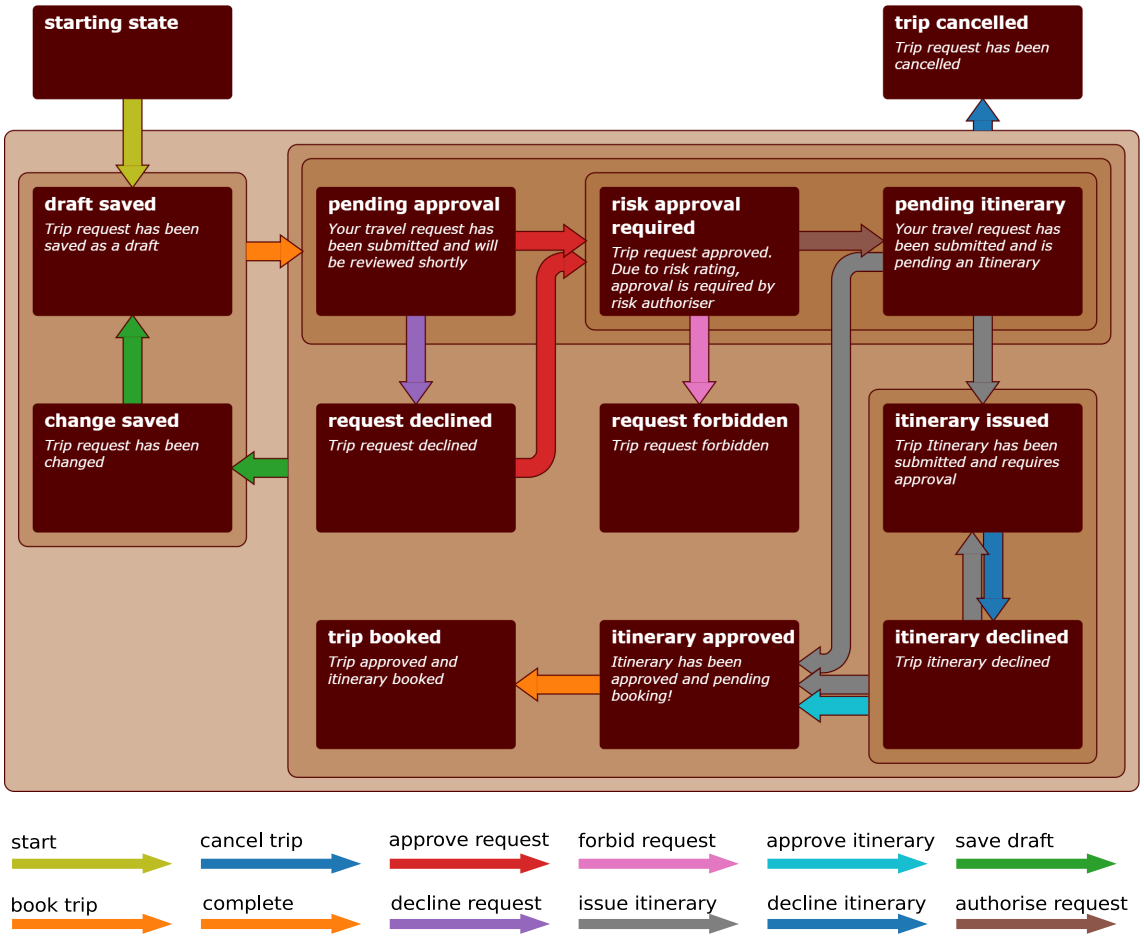


Figure 1.2: A network that represents the different steps of a trip booking process. The nodes of the network represent the current state of the booking, while the links represent a transition from one state to the other. This diagram is drawn using our *Ultra-Compact Grid Layout* method, which is discussed in Chapter 3. The nodes are represented using rectangles arranged on different spatial positions. In this example, the nodes have additional attributes (state label and state description). These attributes are shown using labels on the rectangles. The nodes are also grouped into a hierarchy. The grouping is based on an edge compression technique called *power graph compression* [109], which is discussed further in Section 2.5. The links of this network also have additional attributes. The type of link is represented by a different colour hue, while the direction is denoted by the arrowhead.

- **CH-1**: Methods that produce quality layouts are difficult to adapt to different layout requirements.

- **CH-2**: Methods for quality layout do not scale computationally for large networks, while fast methods that scale for large networks produce poorer quality layouts.

- **CH-3**: Cognitive and perceptual limits mean that networks beyond a certain size are hard to understand, regardless of the quality of layout.

- **CH-4**: Tasks that require an understanding of high-level structure of large, dense and complex networks are not well supported by existing network visualisation techniques.

The quality and readability of network diagrams has been a hot topic in the field of information visualisation. Munzner explains that the effectiveness of visual channels, in general, are calculated through accuracy, discriminability, separability, and the ability of providing visual pop-out and perceptual groupings [233]. More specifically, many researchers have conducted empirical studies in order to understand what makes network diagrams more readable and memorable. We discuss these studies further in Section 2.4.

For node-link diagrams, studies have found several layout features that affect the quality. For example, studies have shown that reducing the number of crossings between the lines representing the edges will positively affect readability. Compactness is another example; there is a general belief that diagrams become less readable and memorable, the more white space is included.

The major challenge is deciding on a placement for nodes and routing the links in a way to maximise desired features, while minimising the undesired ones. However, attempting to create node-link diagrams with some of these features is hard (**CH-2**). For example, finding a layout that minimises edge crossings is NP-complete [136]. Optimising these features also necessitates trade-offs, which further complicate the design process. Attempting to minimise the number of edge crossings might require non-symmetric layouts. The challenge of drawing networks, with as many desirable features as possible, is amplified when the size of the represented networks are large.

Existing methods for network layout focus on several layout features that have been identified to improve readability. These methods are often based on complex multi-stage pipelines, which in turn, adds additional dependencies (**CH-1**). As an example, so-called "orthogonal" approaches seek to break the layout problem down into a sequence of combinatorial optimisation sub-problems, such that the network can ultimately be rendered with only horizontal and vertical line segments. These approaches are appealing since they seek to minimise link crossings, however, they achieve this at the cost of introducing bends into links. Attempting to minimise the number of bends could lead to long links and non-compact diagrams, which are also aesthetics that affect readability. Figure 1.3 shows an example of an orthogonal drawing of a small network with 13 nodes. Such complex, multi-stage approaches to layout introduce dependencies between the order in which the phases occur; changing this order would result in different layouts.

Due to the existence of very large networks in some domains, state-of-the-art methods for network layout mainly focus on heuristic techniques that ease the computational complexity. One of the most commonly used approaches for network visualisation is the force-directed approach (also known as force-based). Layouts created using this approach are regarded as ‘faithful’ to the structure of the network, since connected nodes are kept

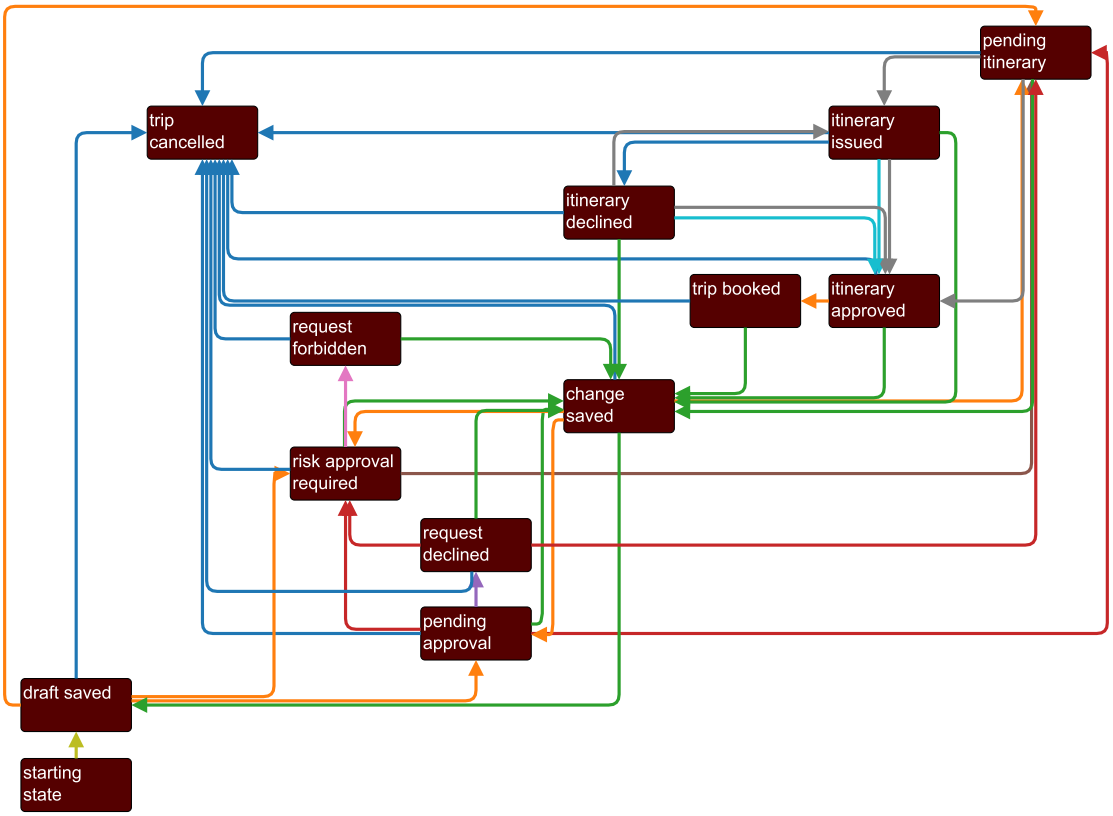


Figure 1.3: The same network of Figure 1.2, which represents the different steps of a trip booking process. The nodes represent the state of the booking, while the links represent a transition from one state to another. In this example, we do not use the edge compression technique. This diagram was created using yEd graph editor [14] and is based on an orthogonal layout.

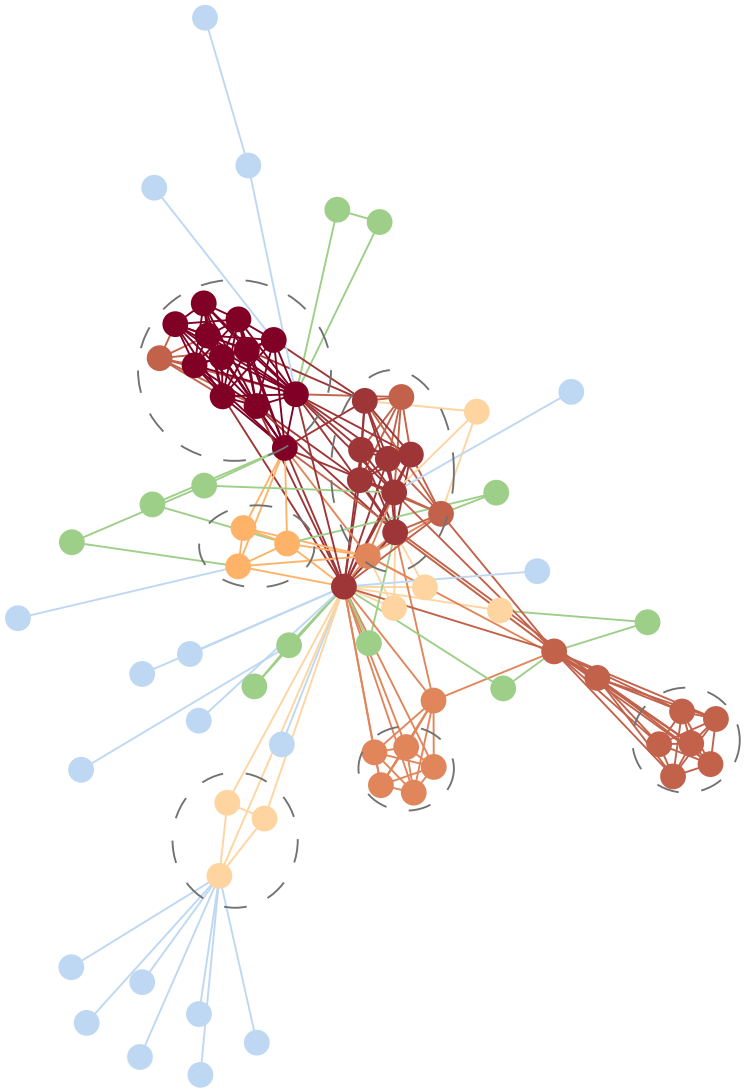


Figure 1.4: This network represents the co-occurrences of the characters in the *Les Miserables* [177] novel. The nodes represent the characters of the novel, and two nodes are linked if the characters appear in the same chapter. The names of the characters are not shown to avoid clutter. This layout was created using the force-directed method of D3 [3]. We can clearly see that the nodes that are closely connected are placed in proximity, resulting in visually identifiable clusters encircled using grey dotted lines. However, many of the marks are drawn on top of each other, which makes it difficult to see the details.

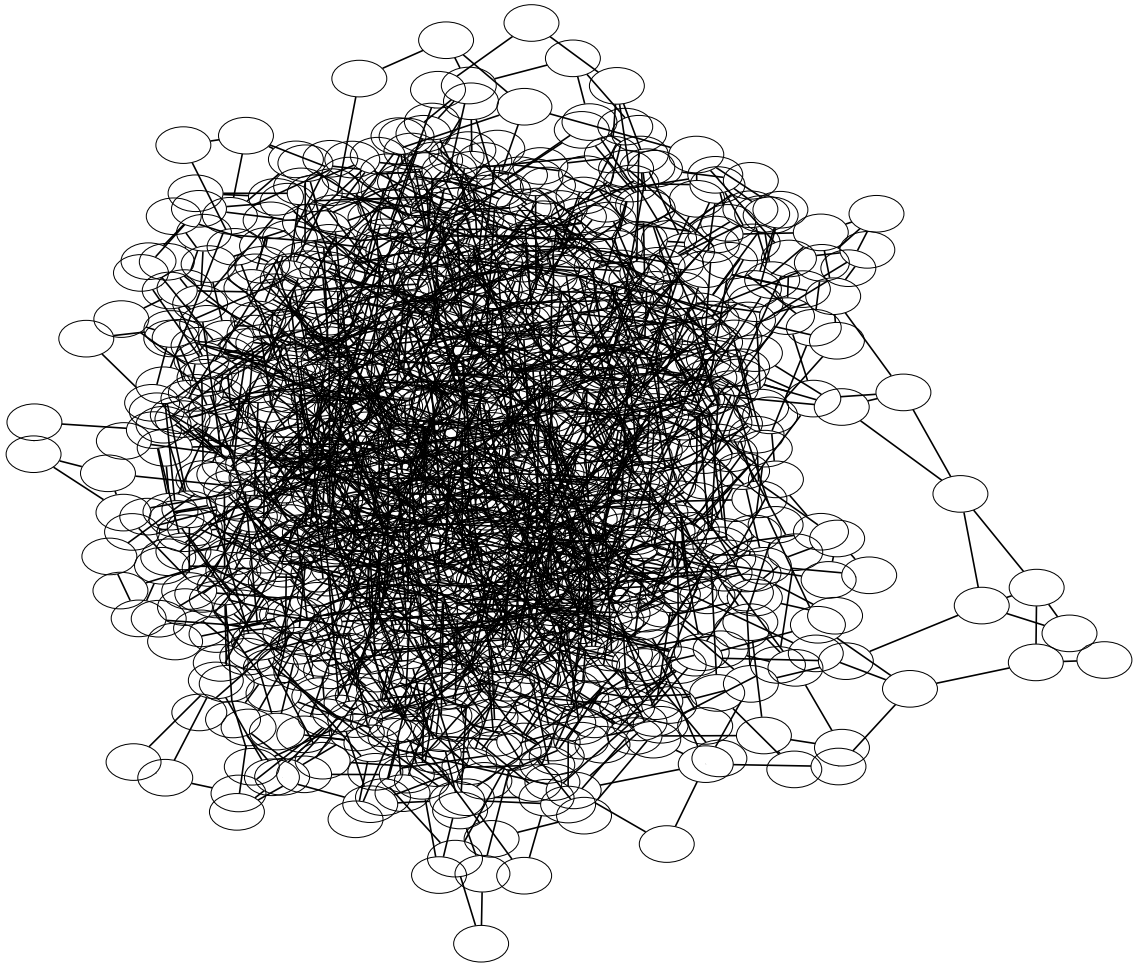
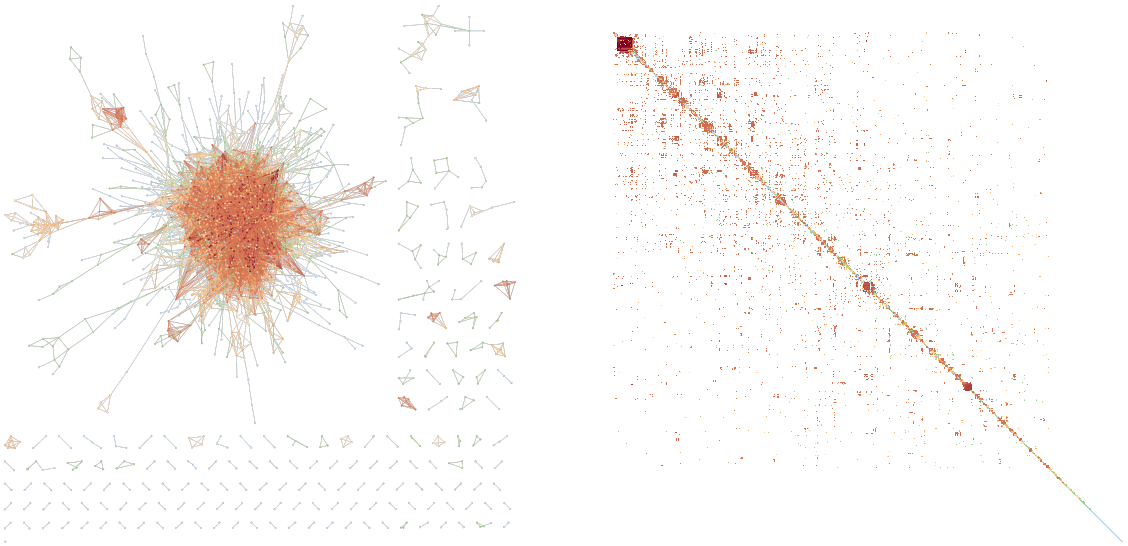


Figure 1.5: A random network with 1,000 nodes and 2,000 edges created using a networkx [154] graph generator, which is based on the *Watts-Strogatz* model [321]. The diagram was created in 869 milliseconds, using the *sfdp* drawing engine of GraphViz [118]. The *sfdp* engine is based on a multi-level force-based algorithm [164] (see Section 2.2.1 for a discussion of force-based graph drawing algorithms).



(a) Node-link diagrams become hard to read when the network is large, and resemble hairballs when the network is dense. This diagram was created using the organic layout of the yEd graph editor [14].

(b) The square matrix grows in area with the increase in number of nodes. The larger it grows the less number of pixels on a screen are devoted to one cell. Beyond a certain value, cells might become invisible, since they will have less than one pixel allocated to them. Furthermore, adjacency matrices resemble out-of-tune TVs for large networks. This diagram was created using D3 [3] and the rows and columns have been ordered using the optimal leaf ordering of Reorder.js [128]. All the cells adjacent to a filled cell in the adjacency matrix were also filled to compensate for the small size of each cell, thus making connections more visible.

Figure 1.6: A protein-protein interaction network, with 3,273 nodes and 15,631 edges, represented with a node-link diagram (left) and an adjacency matrix (right).

close to each other. Figure 1.4 shows an example arranged using a D3 [3] force-based layout method.

Even though some methods that follow this approach are computationally fast and can arrange layouts for large networks, the resulting diagrams often resemble 'hair-balls' (**CH-2**). Figure 1.5 shows a diagram of a network with 1,000 nodes and 2,000 edges drawn using *sfdp* [164], which is one of the most scalable force-based methods.

Apart from the technical challenges of visualising large networks, there is a limit of cognition that naturally arises. This limitation persists no matter how well the network visualisation is designed and represented, or what advanced technologies are used to exhibit them. Displaying an overwhelming amount of information, regardless of how well it is shown, leads to visual clutter (**CH-3**).

According to Munzner, there are five major ways to overcome this limitation: derive new data, change the view over time, partition into multiple views, reduce the amount of data shown within a view, and embed focus and context information together within a single view [233]. We discuss techniques that resort to these measures in Section 2.5. We also propose a summary representation that shows a reduced amount of data in Chapter 6.

While there are numerous interactive tools to visualise large networks, the cognitive limitations of working with such large networks is understudied. Identifying these limitations would help inform the design of more effective tools for network visualisation.

Recent research has also proposed ways to visualise representative summaries of very large networks. However most summary representations are tailored to allow detailed analysis and require complex computation (**CH-4**). *GraphPrism* [185] is an example of a summary representation for large networks. It uses tabular views with colour-coded cells providing a number of node-specific and edge-specific metrics. This is a novel approach that provides a summary of a great amount of information about a network, but will be unfamiliar to most users, and likely hard for them to interpret. Moreover, many of the metrics used require expensive computation (**CH-2**). We discuss summary representations in more detail in Section 2.5.

1.2 Research Aims and Questions

In this section we present our research questions in relation to the challenges discussed in the previous section.

The motivation for our research is to produce a visual representation with the highest possible quality for networks. We aim to explore the limits of cognitive scalability of graph visualisation, beyond which an overview representation with fewer elements would be more suitable than a detailed representation. We also aim to produce a summary visualisation of large networks that is designed to allow general tasks to be performed efficiently, thus tackling both challenges of quality and scalability of network visualisation.

This guides us to our overarching research question;

- **How can we design visualisations that will allow people to efficiently understand and work with network data of all sizes?**

Researchers have identified layout features that enhance the quality of node-link diagrams. Nonetheless, employing these features to create layouts is computationally hard and often leads to complex pipeline methods (**CH-1** and **CH-2**). We want our method to avoid such rigid steps. Rather than the 'HOW', we want to focus more on the 'WHAT', in terms of a final layout. Declarative programming would allow us to formalise a model of the network layout problem with an objective that is independent of the exact steps leading to it.

- **RQ-1: How can we model the network layout problem in a way that general-purpose optimisation solvers can be used to compute extremely high-quality layout with respect to aesthetic and readability criteria?**

We want to explore the limits of cognition from a network visualisation perspective, since the cognitive scalability of node-link diagrams is understudied. Methods for network layout that can handle large networks often sacrifice quality for scalability (**CH-3**). We can make better design choices by understanding how different factors in network visualisation affect its comprehension.

- **RQ-2: What are the factors that limit human cognition of network representations of various sizes?**

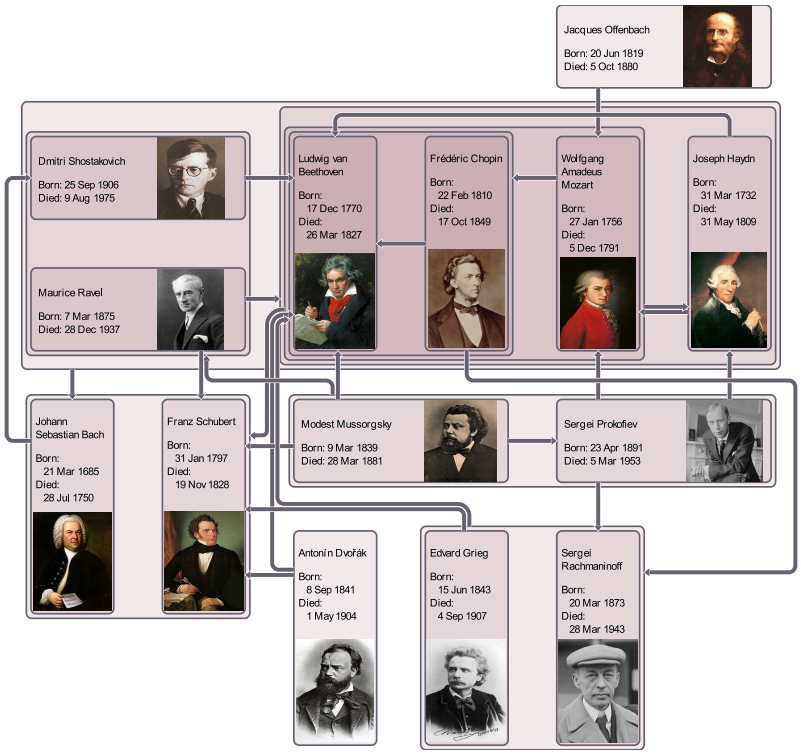


Figure 1.7: A representation of a small network, which shows the relationships between visited Wikipedia pages of about 14 music composers. The layout is optimal, in terms of compactness and proximity of connected nodes. The layout is based on our optimal model for network layout and was solved using SAT. This layout has an additional feature, allowing the nodes to have a 2×1 or 1×2 size. Computing the placement and orientation of the nodes took 37.42 seconds on a regular laptop.

Recent techniques for visualising very large networks aim to handle the issue of cognitive scalability (**CH-3**) by using interactivity. However there is a need for summary representations that show enough structural information to allow viewers to perform overview tasks efficiently (**CH-4**).

- **RQ-3: What visualisation designs better support overview tasks for very large networks?**

Techniques for summary representations should also be computationally fast in order to scale to large networks and large sets of networks (**CH-4**).

1.3 Contributions

The work presented in this thesis has several contributions. We group them into three main themes, which are discussed further in this section.

High-Quality Layout for Small Networks

In response to **RQ-1** (Section 1.2), we propose a novel model for network layout: *Ultra-Compact Grid Layout*, which aims to maximise compactness and minimise distances between connected nodes (Chapter 3). It also supports hierarchical groupings and is easily

adapted to support other layout features. We show the benefits of this layout model via real world examples. We also create variations of the model, which support other desired layout features, thus showing the flexibility of our proposed method. Figures 1.2 and 1.7 show examples that were created based on our layout model. For the example shown in Figure 1.7, we allow the nodes to be positioned either as portrait or landscape, but still achieve an optimal layout in terms of compactness and short links.

We devised a generic optimisation problem that represents this model and developed four implementations. Two of the implementations were developed using Mixed Integer Programming (MIP) – *OPL* and *MiniZinc* [236] and solved using *CPLEX* [2]. One was developed using Constraint Programming (CP) –*MiniZinc* [236] and solved using *G12/CPX* [129]. The implementation found to be most practical for small networks was developed using the Boolean Satisfiability Problem (SAT) and solved using *BumbleBEE* [226]. The experimental evaluation of these implementations showed that the SAT solver was able to handle larger networks than the *G12/CPX* solver and *CPLEX*.

However, even the best solver could only offer an optimal solution in reasonable time for small graphs (e.g. under 100 nodes). To try to scale the general approach to larger networks, we also developed a modified version of the MIP implementation using a *Large Neighbourhood Search* meta-heuristic. This version starts with an initial layout created using *WebCola* [13] with constraints to place the nodes on grid points. Parts of this layout (neighbourhoods) are iteratively reconfigured to achieve an optimal placement. Dividing the whole problem into sub-problems and solving them iteratively in this way, reduced the strain on the solver and achieved faster running times. The experimental evaluation showed that this approach scales to larger networks.

This work has been published in:
Yoghourdjian, V., Dwyer, T., Gange, G., Kieffer, S., Klein, K., & Marriott, K. (2016). High-quality ultra-compact grid layout of grouped networks. IEEE Transactions on Visualization and Computer Graphics, 22(1), 339-348.

What is Small, What is Large?

We have identified a size threshold for node-link diagrams, in terms of number of nodes and density, beyond which tasks that require a detailed understanding of the network structure become very difficult (**RQ-2**).

We wanted to understand the limits of complexity in terms of network size with which people could effectively visualise using node-link representations. That is, while the problem of computational scalability of network layout is well studied, we wanted to know more about what the human limitations of understanding visualisations of graphs was, and whether we could get some insights into these human-limitations from a meta-study of the literature reporting human trials of various network visualisation techniques. Thus, we conducted a survey, which covers 152 empirical studies that evaluate node-link diagrams. These studies were gathered from 124 articles, which were published in reputable journals and conference proceedings. The survey showed that most of the studies used sparse networks with up to a few hundred nodes. Beyond these limits most visualisations use interaction to focus only on parts of the networks at given times. We present the survey and our findings in Chapter 4.

We were unable to find any existing study that directly explores the limits of cognitive scalability with respect to the size of networks. Thus we conducted a controlled study to

explore the effects of network size on cognitive load. In our study, we showed node-link representations of networks with different sizes and asked the participants of the study to find a shortest path between two given nodes. In addition to the common measures used to analyse efficiency (such as accuracy, response time, and subjective feedback), we used physiological measures, such as eye-tracking data, heart-rate, and electrical activity of the brain.

Our results showed that cognitive load increases with the increase in network size and becomes overwhelming for graphs with many nodes, or dense graphs. We discuss this in more detail in Chapter 5. The results also showed strong correlations between different layout features and efficiency measures. This validates the usefulness of our optimal layout, which does not scale to networks with more than a hundred nodes, beyond which cognitive load becomes challenging.

Based on our findings from the survey (Chapter 4) and user study (Chapter 5), throughout this thesis we use *small* to refer to networks with 20 nodes or less, *medium* for networks with more than 20 nodes but 50 nodes or less, *large* for networks with > 50 and ≤ 200 nodes, and *very large* for networks with > 200 nodes. We use *sparse* for networks with densities of more than 1 and densities of 2 or less, *dense* for networks with > 2 and ≤ 4 densities, and *very dense* for networks with densities of more than 4.

The survey has been submitted to the *Information Visualisation Journal*, while the user study has been submitted to *IEEE Conference on Information Visualization*.

Summary Representation of Large Networks

For large and very large networks, and in response to **RQ-3**, we propose a summary representation. The studies covered in our survey that considered large or very large networks tended to involve tasks that did not require a detailed level of information (we discuss task types further in Chapter 4). Similarly, the results of our user study, which is discussed in Chapter 5, showed that detailed tasks become too difficult on large networks.

We explored different graph decomposition techniques, in order to reduce large network data to smaller representative networks. We wanted to use a decomposition technique that is extremely fast, hides most of the low-level nodes and links of the network, and would result in an intuitive summary that could be easily related to the original network, unlike existing summary representations that are non-intuitive and computationally expensive to create.

We partitioned the network into components based on the denseness of their connections. This can be done in linear time of the number of edges. Moreover, the resulting summary highlights the densely connected parts of the network and identifies parts that are not closely related.

We represented this summary using different visualisation idioms that work well with hierarchical structures: treemaps, icicle plots, and packed circles.

We conducted an experimental evaluation to validate the theoretical running time of creating our proposed summary using circle packing, which we call *Graph Thumbnails*, by comparing it to the actual running time. Our results show that a *Graph Thumbnail* representation can be created for a network in linear time with respect to the number of edges.

We also conducted two user studies to evaluate the effectiveness of *Graph Thumbnails*.

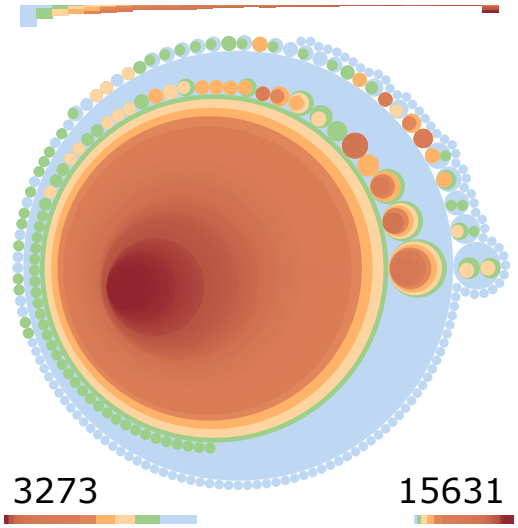


Figure 1.8: The same protein-protein interaction network of Figure 1.6, represented using our technique for visualising structural summaries of large networks–*Graph Thumbnails*. The number on the lower left indicates the number of nodes in the graph, while the number on the lower right indicates the number of edges. The stacked bar underneath the numbers shows the distribution of nodes across the different coloured components of the thumbnail. The histogram on top, shows the degree distribution.

The first study aimed to compare *Graph Thumbnails* to node-link diagrams and adjacency matrices in terms of the ability to show similarities and differences between pairs of networks. The results showed that *Graph Thumbnails* and adjacency matrices outperform node-link diagrams significantly. They also showed that participants were better at identifying similarities when using our method over the two other representations. According to the post-survey questionnaire, the qualitative results for preference were in favour of *Graph Thumbnails*.

In the second user study, we asked the participants to perform three overview tasks given the three visual representations. The results show that *Graph Thumbnails* outperform the other two on all tasks.

Our evaluation confirmed that our proposed informative summary suits overview tasks better than the detailed node-link and matrix views. We also show through a usage case how to use *Graph Thumbnails* in a real-world example.

This work has been published in:

Yoghourdjian, V., Dwyer, T., Klein, K., Marriott, K., & Wybrow, M. (2018). Graph Thumbnails: Identifying and Comparing Multiple Graphs at a Glance. IEEE Transactions on Visualization and Computer Graphics.

1.4 Thesis Outline

This thesis is structured in the following way. In Chapter 2 we provide the necessary notations and characteristics of different networks. We present popular graph drawing approaches and algorithms in Section 2.2. In Section 2.3 we discuss common tasks that are performed on network visualisation. In Section 2.4, we discuss aesthetics and layout features which affect readability. We also discuss methods for community detection and

common summary representations in Section 2.5. Further background is presented as necessary in each chapter.

In Chapter 3 we present a novel approach to solve the network layout problem using combinatorial optimisation techniques. The resulting layout, which is based on our *Ultra-Compact Grid Layout* model, is optimally compact and enforces grouping and non-overlap constraints. We also discuss the advantages of this approach and provide three implementations, using Mixed Integer Programming, Constraint Programming and the Boolean Satisfiability Problem. As an extension to the optimal approach, we propose a method that uses *Large Neighbourhood Search* meta-heuristics. Lastly, we evaluate the running time of all the methods.

Chapter 4 presents an exhaustive survey of empirical studies involving node-link diagrams. In Chapter 5 we present a user study that explores the effects of graph size and other visual factors on the cognitive load of participants when finding the shortest path between two nodes in node-link diagrams.

In Chapter 6 we present a technique to decompose large networks into meaningful summaries (*Graph Thumbnails*). We discuss the design process and mention the complexity of the algorithm. We also provide an experimental evaluation.We evaluate the usefulness of our technique by conducting two user studies that compare *Graph Thumbnails* to node-link diagrams and adjacency matrices. We also present a usage case that demonstrates a useful application of *Graph Thumbnails*.

Chapter 7 presents overall conclusions and ideas for future work that lead on logically from the topics explored, and the findings revealed in this thesis.

Chapter 2

Background and Literature Review

«Ճանաչել զիմաստութիւն եւ
զխրատ, իմանալ զբանս հանճարոյ:»

The first sentence written using the Armenian alphabet translated from the Bible (*Proverbs 1-2*):
"for gaining wisdom and instruction;
for understanding words of insight;"

In this chapter, we present definitions and terminology that are necessary to understand the work described in this thesis. We also present related work with respect to the challenges and research questions outlined in Section 1.1 and Section 1.2, respectively. Many of the algorithms for network layout were developed by the *graph drawing* research community and have strong theoretical bases, thus, in Section 2.1 we briefly cover some necessary definitions of *graph theory*. We also discuss the foundation of *graph drawing* as a pre-requisite for network visualisation in Section 2.2.

In Section 2.3, we discuss tasks that are commonly performed when using networks. Network visualisations are often evaluated based on their efficiency in supporting certain tasks. Since the quality of network visualisation is one of the major topics in this thesis, we present studies that quantitatively and qualitatively evaluate layout aesthetics and features (Section 2.4). One of the contributions of this thesis is a novel summary representation for large networks that provides insight into the structure of the network, while hiding details that would otherwise create clutter. In Section 2.5, we present community detection and some popular summary representations.

2.1 Graph Theory

Graph Theory is a branch of discrete mathematics that deals with graphs. A graph $G = (V,E)$ consists of a set V of vertices and a set E of edges. An edge $e \in E$ represents a relationship between a pair $(u,v) : u,v \in V$ of vertices. In the case where $u = v$, then the edge is called a 'loop'.

An important characteristic of vertices in a graph is their degree or valency. The degree of a vertex represents the number of edges connected to it. A vertex $v \in V$ can exist in G without belonging to any edge $e \in E$. Vertices that do not have any edge connecting to them are also known as *0-degree*, or isolated vertices. A graph is known to

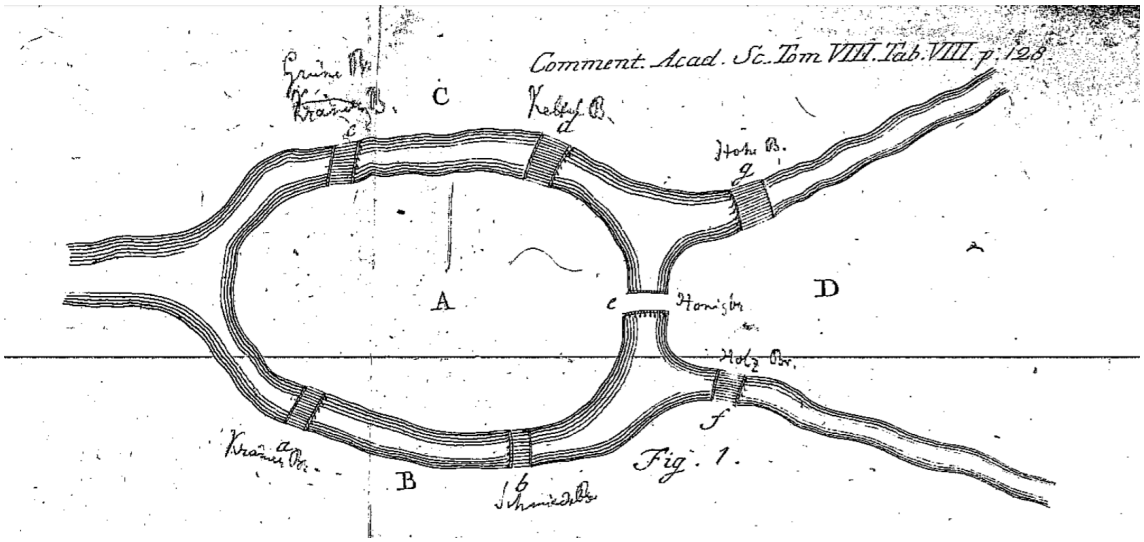


Figure 2.1: A simplified diagram created by Euler, which shows the seven bridges of Königsberg [14].

be disconnected if there exists any pair $(u,v) : u,v \in V$ of vertices that cannot be reached from one another. A graph with only one vertex is known as a 'trivial graph'.

The number of vertices and the number of edges in a graph are finite. In theory, the number of nodes is referred to as the order of the graph, and the number of edges is known as the size of the graph (although in practice, size also has different meanings). Terms such as 'small graphs' and 'large graphs' are often used in practice to refer to the number of vertices, while the terms 'sparse' and 'dense' are used to refer to the number of edges. We revisit these terms in Chapters 4 and 5 and explore graph size with respect to human cognition.

An undirected graph is one whose edges are not directed. In contrast, an edge e in a directed graph has a source u and a target v. In other words, the pair (u,v) for a directed edge e is ordered. A weighted graph is one that has a quantitative attribute associated with each edge. A complete graph has an edge between all pairs of vertices. Trees and regular graphs are two of the simplest types of graphs. A connected undirected graph with no cycles is a tree. A tree may also be defined over a directed graph — no node has more than one incoming edge. While a graph whose vertices have equal degrees is called a regular graph. Graphs are classified into categories based on their structure. For example, in *scale-free* graphs, the degree distribution of the vertices follows a power law. In other words, *scale-free* graphs have few nodes with high valency and many nodes with a few connections. In addition to *scale-free* graphs, we use graphs from different classes in our user studies of Chapter 6.

The earliest known use of a graph by a mathematician arose from a notable mathematical problem known as the "Seven Bridges of Königsberg". The challenge was to find a route through the city that would cross each of the seven bridges only once. In an article to prove that this problem has no solution, Leonhard Euler considered the seven bridges of the city of Königsberg to be seven edges between four vertices, as shown in Figure 2.1 [122]. This transformation is an example of how simplified abstractions help solve a problem. Euler's theory suggests two possible cases where a route can be found that would cross each bridge only once (also known as an *Euler walk*). One possible case is when the representative graph has exactly two vertices with odd degrees. In this case,

the two odd-degree vertices are the start and the end points. Another possible case exists when there are no vertices with odd degrees. In this case, however, the starting and ending points are the same. All four vertices in the *Königsberg* example have odd degrees, thus, an *Euler walk* does not exist.

Graph theory commonly deals with topics of identifying, classifying, and counting graphs and subgraphs with specific characteristics. Topics related to connectivity and traversability are also common. Similar to the "Seven Bridges of Königsberg" problem, there are other problems that deal with path finding and reachability, such as the shortest path problem. We explore the limits of the ability of node-link diagrams to assist in path finding problems in terms of network size in Chapter 6.

Another major branch of problems in *graph theory* involve decomposition, which aims to break a graph into meaningful sub-graphs. Community detection is another example, where vertices of a graph are grouped into sets that share common characteristics. We decompose large graphs into much smaller trees based on the connectivity structure of the graphs in Chapter 6.

Graphs naturally arise in different application areas. Molecular structure in Chemistry can be represented as graphs. Other examples of graphs arise in Biology (we present a usage case of an optimal layout for a biological pathway in Figure 3.7 of Chapter 3. Another example of a biological network is presented through a usage case of our proposed summary representation; to visualise a set of protein-protein interaction networks in Chapter 6 (Figure 6.14). Networks are also common in circuitry and software engineering. We present a usage case of an optimal layout for a software-dependency network in Chapter 3 (Figure 3.2).

Graph theory is rich with other definitions and characteristics of graphs. The *Handbook of Graph Theory* [148] provides more details and additional definitions.

2.2 Graph Drawing

Graph Drawing is a field with roots in Mathematics and Computer Science that deals with visually representing or embedding graphs in a plane or space. Different approaches and standards have been applied to represent graphs in the two-dimensional planes, and three-dimensional space. Dots are often used to represent vertices and an arc is drawn between two dots to represent an edge if the two vertices are connected. However, the terms 'dots' and 'arcs' were more often used before the age of computers and are thus, replaced by 'nodes' and 'links' respectively.

One of the earliest problems in *graph drawing* is the embedding of graphs on the Euclidean plane. In a planar embedding, the dots representing the vertices are assigned points on a two-dimensional plane, while each edge is represented by an arc connecting pairs of points, such that no arc intersects any other arc, including itself. Nonetheless, some graphs cannot have a planar embedding. The graphs that can be embedded on the plane are called 'planar graphs'.

Other problems in *graph drawing* involve maintaining symmetry and minimising number of crossings as an optimisation problem. We refer to the *Handbook of Graph Drawing and Visualization* [297] for more details.

Graph drawing algorithms take a graph as input and return a drawing of that graph. One of the earliest *graph drawing* algorithms was proposed by Tutte, which decides on the position of a node based on the average position of its connections [303].

2.2.1 Graph Drawing Models and Methods

The field of *graph drawing* evolved further, with the need for algorithms to draw general graphs rather than specific classes of graphs, such as planar or sparse (few edges), and tree-like graphs. That is, early methods were applicable only to very limited classes of graphs. For example, Tutte's method was only applicable to planar, bi-connected graphs (each node is reachable from all other nodes by two distinct paths). A single graph can be drawn in several different ways. Thus, a major challenge in graph drawing is to decide how to achieve a good arrangement, in terms of readability.

In this section we present some of the commonly used methods and algorithms for drawing graphs.

Force-Based Approaches

The earliest network layout algorithms are mostly force-based. The force-directed approach was independently discovered by Fisk and Isett [130], and Eades [111]. Fisk and Isett assumed components of a circuit to be connected by elastic leads, "which are in a state of tension until they contract to some arbitrarily short length". Similarly, Eades [111] likened vertices to metal balls connected to each other by springs. When suspended and released, the nodes end up in positions that reflect the network's structure.

The earliest variations of the force-directed approach focus on modifying the force model. For example, Kamada and Kawai [186] present a new force-directed model, which takes into consideration a desired distance between connected nodes. The computational complexity of their proposed algorithm is generally $O(|V|^3)$ or $O(|V|^2 log|V| + |E||V|)$ for sparse graphs [297]. In contrast, Fruchterman and Reingold [132] argue that the model does not have to be physically realistic and ignore the physical reality of the repulsion and attraction forces. In their model, they make sure that nodes are evenly distributed, in addition to having uniform edge lengths and inherent symmetries. They also propose an iterative improvement of the layout using a linear measure called 'temperature'. Each iteration of their algorithm has a complexity of $O(|V|^2 + |E|)$ [297].

More than thirty years after the first force-based algorithm, Hadany and Harel [152] introduced a 'multi-scale' technique that divides the layout problem into coarse-scale and fine-scale node position adjustments. They show examples of diagrams for networks with more than 1,000 nodes, achieved by their technique.

The force-directed approach remains the most commonly used in *graph drawing* methods, due to its computational speed. The most scalable examples, developed so far, are the multi-level algorithms of Walshaw [315], Hachul and Jünger [151], and Hu [164]. All three algorithms have a worst case running time of $O(|V|log|V| + |E|)$ per iteration.

The aesthetic benefits brought forth with the force-directed approach degrade as the graphs get larger in size or become denser. The overlap of marks, especially edge crossings in dense graphs, makes it hard to see the underlying structure (e.g. Figure 1.5). Nonetheless, it is still one of the most widely used approaches, mainly due to its simplicity to implement and its speed.

We make use of the force-directed approach to create an initial layout for our iterative method, discussed in Chapter 3.

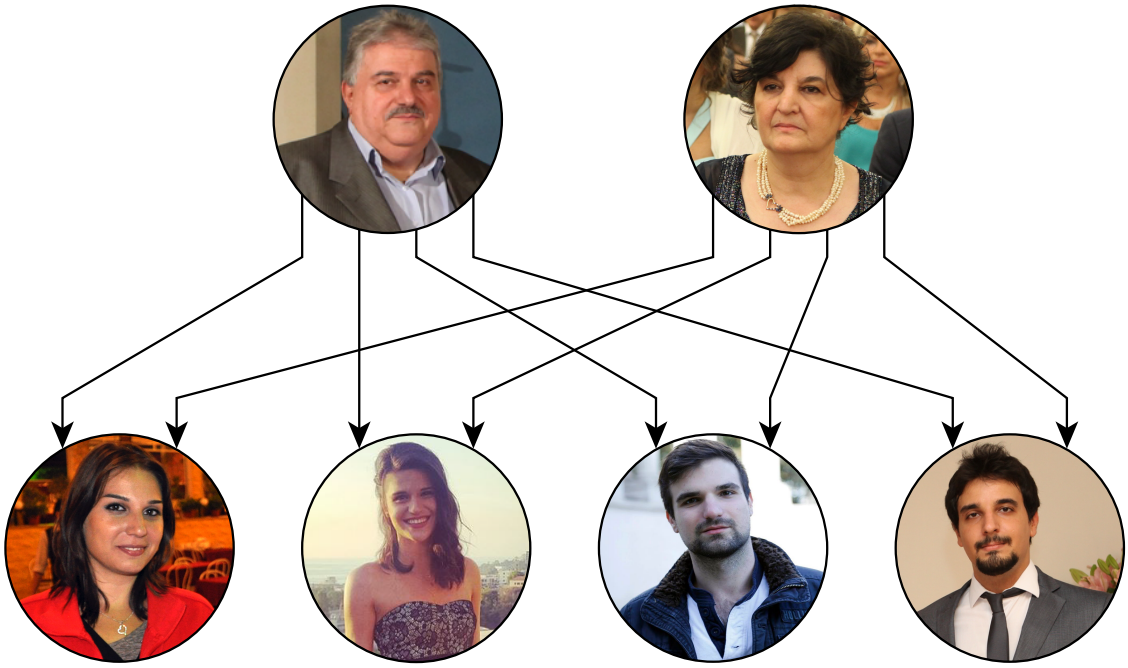


Figure 2.2: My family represented as a graph using a hierarchical layout. The hierarchy is represented using levels where nodes are placed on the same row if they belong to the same level. For example, my parents are placed on the first row and my siblings and I are placed on the second row. Links pointing downwards are used to connect my parents to each of my siblings and myself. This diagram was created using yEd [14]

Layered Approaches

In many cases, the relationships between the vertices are directed, and imply a hierarchy. Arrowheads are often used to indicate the direction of the edges. An arrowhead is placed at the side of the destination or target node. The hierarchy is shown by placing nodes of different hierarchies on separate horizontal or vertical layers. Figure 2.2 shows an example of a network arranged using a layered approach.

One of the key additions to the layered approach, compared to the force-based approach, is the requirement that links are to be pointing in the same direction. Sugiyama *et al.* [293] propose a multi-stage approach to arrange graphs using a layered layout, which respects hierarchies. During the first stage of the algorithm, a physical hierarchy is formed. The second stage deals with 'breaking' cycles. An edge pointing to the opposite direction is reversed to 'break' a cycle. The third stage gets rid of long edges by adding dummy vertices where edges cross layers. The fourth stage deals with replacing vertices in each level of the hierarchy to achieve a minimum number of crossings. Vertices are moved again in the fifth stage, in order to achieve secondary characteristics, such as straight lines and proximity. In the final stage, dummy nodes are removed and long edges are reinstated. Sugiyama's algorithm has a computational complexity of $O(|V||E|log|E|)$

Recent research has proposed alternative ways to achieve the desired results of the various stages of the original algorithm. For example, Eiglsperger *et al.* [117] propose a faster approach ($O(|V|+|E|log|E|)$) that avoids the insertion of unnecessary dummy nodes.

The biggest challenge of the layered approach is the rigidity imposed by the multi-stage structure. The inherent dependencies between the stages make it difficult to change

the layout at later stages. We aim to overcome this challenge by using declarative programming techniques, discussed in Chapter 3.

Orthogonal Approaches

Another popular approach for network layout algorithms is 'orthogonal' layout, which uses horizontal and vertical line segments to represent the edges. The main motivation, for this approach, is to increase the angular resolution of the layout. Angular resolution increases when the edges that stem from the same vertex are further apart. Minimal bends and compactness are also aesthetics that are often considered in algorithms that are based on the orthogonal approach. These two aesthetics are often contradictory and have trade-offs.

Batini *et al.* [56] presented a way to arrange networks in an orthogonal layout style. Their method is widely used and is known as *Topology-Shape-Metrics* (TSM). TSM is incremental and consists of three main steps: *Planarisation*; the first step, minimises edge crossings and the external boundaries. *Orthogonalisation*; the second step, minimises the number of bends. *Grid embedding*, which is the third and final step, minimises the global length of edges and the area of the smallest rectangle covering the diagram (compactness). The three steps deal with three sets of characteristics, namely topology, shape and metrics. The complexity of their proposed algorithm is $O(|V|^2 + |E|)$.

Most orthogonal algorithms arrange the nodes on a grid (e.g. [191, 296]). Grid arrangements make the diagrams memorable and easy to follow [222] [232]. They are used by designers in typographical layouts, where a viewing space is divided into regular cells. Also, people who draw graphs manually prefer to use a grid-layout [263].

We use an optimally compact orthogonal grid layout called *CompactGrid* in Chapter 3. Similar to the layered approach, orthogonal-based methods suffer from dependencies between the stages of their multi-stage structure. Our declarative approach does not require such dependencies and, instead, uses weighted objectives which are easily modified per the requirements of the application.

2.2.2 Network Layout as a Combinatorial Optimisation Problem

Combinatorial Optimisation deals with problems, where an optimal combination of decisions must be made on a set of variables, while satisfying a set of constraints.

An optimisation problem is often described as a model with input parameters, decision variables, constraints that must be satisfied, and an objective function that is either minimised or maximised. The knapsack problem is a common example of a *combinatorial optimisation* problem. In this problem, a knapsack exists with a limited weight capacity. It has to be filled with items from a set, where each item has a weight and a value. The objective is to fill the knapsack based on the highest value, but not exceed the weight limit of the knapsack.

In the case of network layout problems, multiple layout features can be expressed either as constraints, which need to be satisfied, or as part of the objective function that is optimised.

There are several programming paradigms that can be used to describe an optimisation problem. *Integer Linear Programming* (ILP), *Mixed Integer Programming* (MIP) [95], *Constraint Programming* (CP) [209], and *Boolean Satisfiability* (SAT) [84] are examples of generic methods for solving combinatorial optimisation problems. For example, the

difference between ILP and MIP is that, in ILP, all the variables must be of type integer, while in MIP, the variables do not have to be discrete.

Sometimes the problems associated with graph layout are too hard to be solved to optimality in a reasonable amount of time. In such cases, heuristics can be used to speed up the optimisation. Heuristics can dramatically enhance the solving time with relatively small effort, but do not guarantee optimal solutions and lead to approximate solutions that are often good enough. Meta-heuristics are top-level strategies that guide the use of underlying heuristics to solve a given problem. Simulated annealing, Neighbourhood Search, Greedy Randomised Adaptive Search, The Pilot Method, Tabu Search, and others are examples of meta-heuristics [313].

For example, *Simulated Annealing* is named after the chemical transformation of crystals into their solid forms and involves a gradual cooling of temperature in order to yield the best shapes [137]. Davidson and Harel [91] use it in graph drawing. They define an objective function that aims to achieve even distribution of nodes, compactness, uniform edge length, and minimum number of crossings. The computational complexity of their proposed algorithm, which terminates after a constant number of stages, is $O(|V|^2|E|)$.

Many of the methods and algorithms presented in Section 2.2 use *combinatorial optimisation* partially in their approach. Batini *et al.* [56] used it to minimise the number of crossings. Klau and Mutzel [195] used ILP to enhance compaction. Sugiyama *et al.* [293] used it to minimise bends. The main limitation for not using generic *combinatorial optimisation* methods has mainly been due to performance limitations. Gange *et al.* [133] use SAT and MIP solvers to minimise crossings and minimise edge deletion, for planarisation to optimality, in layered network layout.

Combinatorial Optimisation techniques have advanced in the recent years. For example, Lima *et al.* [216] evaluate the effect of enhancements in MIP on two classic problems and report massive improvements in terms of time.

Computing power has also advanced tremendously in the last two decades. The average memory of personal computers in 1995 was 4–8MBs, whereas nowadays it's 4–8GBs. Processing power has had a jump as well, with 200 MHz[1] processors in 1996 to 4.2 GHz[2] processors with multiple cores in 2015.

In addition to improvements in solver techniques, enhancements have occurred at the level of designing the model. Previously, the models were encoded using low level clauses, but in time new higher-level modelling languages have been developed [61] (e.g. Minizinc [236]).

Many decision problems in diverse fields, where optimising characteristics of the final results are desired, benefit from the advances in *combinatorial optimisation* techniques. Network visualisation fits perfectly into this category. In Chapter 3 we discuss the use of generic optimisation solvers to create network layouts with optimal quality with respect to different layout features. We also evaluate different generic optimisation solvers with respect to the network layout problem.

[1] http://www.intel.com/pressroom/kits/quickrefyr.htm#1996

[2] http://ark.intel.com/products/family/88392/6th-Generation-Intel-Core-i7-Processors#@Desktop

2.3 Tasks for Network Visualisation

In this section, we discuss different types of tasks that are performed on network visualisations. In the literature, tasks are sometimes perceived as those performed by the visualisation; we consider those as interactions and consider tasks to be queries or activities performed by the analysts or the users.

Shneiderman and Aris [288] present 14 basic tasks that are often performed on networks. They explain that these form a starting point and that "There are an unlimited number of tasks that could be defined.". Lee *et al.* [212] present compound tasks that represent detailed tasks. They also categorise these compound tasks into four groups: *Topology-based*, *Attribute-based*, *Browsing*, and *Overview*. We revisit this taxonomy in Chapter 4 and provide a visual description in Figure 4.7(a).

Tasks in the categories: *Topology-based*, *Attribute-based* and *Browsing*, require a detailed understanding of the network, while *Overview* tasks, according to Lee *et al.* [212], are exploratory and are used to get estimates quickly. Lee *et al.* [212] also discuss 'high-level tasks', which are excluded from the four categories. They consider tasks related to changes in a graph over time to be 'high-level tasks'. Nonetheless, dynamic graphs that change in time have many additional tasks that need to be considered. A separate taxonomy by Ahn *et al.* [18] includes tasks for dynamic graphs.

Throughout this thesis we differentiate between tasks that require a detailed understanding of the network data, such as finding the shortest path between two nodes in a network, and those that can be performed with a high-level understanding of the network structure (e.g. estimating which network is the most dense in a set of networks); except in Chapter 4, where we survey the literature with respect to empirical studies that include node-link diagrams and use the taxonomy of Lee *et al.* [212].

2.4 Evaluating Network Diagrams - Cognition and Perception

The algorithms and methods described in Section 2.2 are designed to achieve certain layout features and aesthetics that improve readability. However, these algorithms were not evaluated until the 1990s, 30 years after their conception. Most of the early evaluations compared existing algorithms to each other rather than focusing on domain application and requirements. In this section, we present a wealth of research that has empirically evaluated these algorithms and aesthetics.

One of the earliest works by Himsolt [158] focused on evaluating layout quality, based on algorithms rather than features. Even so, layout features were indirectly explored, since most algorithms are associated with specific layout features. He evaluated layout algorithms based on the layout features they handle, the restrictions on the characteristics of the input graphs, and their time performance. He also considered a visual preference/ rating by assigning a higher score to layouts he preferred. He concluded that traditionally used features for layout and their rankings must change, and existing algorithms should be revised in order to be more flexible. Similarly, Purchase [257] evaluated the efficiency of layouts produced by eight existing layout algorithms. The eight algorithms were a mix of force-based, orthogonal, and layered approaches. The results showed that even though each of the eight algorithms were designed according to different aesthetics, the performance of the participants did not differ significantly.

In an earlier work, Purchase *et al.* [260] conducted a task-based user study in order to assess the influence of three layout features, namely crossings, bends and symmetry. The participants were asked to perform the following tasks; identify the shortest path between two given vertices, the minimum number of vertices that need to be removed to eliminate paths between two given vertices, and the minimum number of edges that need to be removed to eliminate paths between two given vertices. The results showed that the number of crossings and bends have a large influence on quality, but the results did not show any conclusive relation between symmetry and task performance. In another study, Purchase [256] discovered that the number of edge crossings is a more important aesthetic than the number of bends and symmetry. Furthermore, the angle between edges on a node and an orthogonal drawing of edges and nodes did not reveal any significant advantage. In a later work, Ware *et al.* [320] evaluated six layout features, namely continuity, number of crossings, angle of crossings, number of branches, shortest path length, and geometric line length. They asked the participants to find the shortest path between two marked vertices. They concluded that the 'bendiness' of the shortest paths had a stronger effect than the number of crossings.

In order to evaluate diagrams according to layout features, Purchase formulated ways to measure their presence in network diagrams [258]. She devised formulae to measure seven layout features: number of crossings, number of bends, symmetry, minimum angle between edges connected to a node, edge orthogonality, node orthogonality, and consistent flow direction. These metrics can be used to either evaluate already drawn graphs, or to formulate a cost function for algorithms that perform optimisation. Purchase argues that the decision should be about the weight of each layout feature, rather than simply excluding or including features. Our layout model, which is described in Chapter 3, allows the assignment of weights to layout features defined as objectives, according to layout requirements.

Other than assessing existing layout algorithms and predefined features, more recent work has attempted to identify features that are not found in layouts produced by existing methods, but are preferred by humans and widely used in human-generated diagrams. Ham and Rogowitz [305] asked users to rearrange the vertices of given network diagrams to achieve graphs that best represented the data structure. They evaluated the results by measuring cluster separability, extraction, distance, and delineation features in the rearranged diagrams. They concluded that the layouts produced by the participants where highly structured and had less crossings. The results also showed that uniform edge lengths were not particularly sought after. Dwyer *et al.* [106] conducted a study similar to the former; they discovered that the participants created layouts with less crossings. They conducted a second study where they evaluated the user-generated layouts and compared them to force-based, orthogonal and circular[3] layouts, which were achieved using automated algorithms. Similar studies by Purchase *et al.* [263] and Kieffer *et al.* [191] discovered that humans prefer grid-based layouts. Based on their findings, Kieffer *et al.* [191] developed an algorithm that takes into consideration layout features discovered through user studies done on human-generated orthogonal layouts.

There is a lot of work done in investigating the correlation between human-preferred and algorithm-based layout features, as well as the quality of network diagrams on one hand, and the trade-off between these features on the other. However, some of these works are hard to find and are distributed across a variety of publications. In order to have a better understanding, avoid repeating the same experiments, and to advance the field, it

[3] when the nodes are arranged on the circumference of a circle and the edges are chords

is important to conduct an extensive review of the literature. In Chapter 4 we present our exhaustive survey on empirical studies performed on node-link diagrams.

2.5 Visual Representations of Large Networks

In addition to the computational complexity of layout algorithms, beyond a certain number of nodes and links, visual marks representing these nodes and links start overlapping and covering each other. Layout algorithms that can produce node-link diagrams of large networks often aim to arrange the nodes in a way to reflect the structure of the network. However, even the layouts with optimal arrangements will reach a limit and resemble 'hair-balls'. Adjacency matrices and hybrid visualisations are also used to visualise large networks. The rows and columns of adjacency matrices can be reordered to show different structural information (e.g. [128]). The results of a study by Ghoniem *et al.* [140] show that adjacency matrices are more effective than node-link diagrams for most tasks except path finding. More recent studies show that node-link diagrams are as useful, to perform tasks that require an understanding of network structure, as adjacency matrices [17,247]. The results of a study by Okoe *et al.* [247] confirms that adjacency matrices performed poorly for path finding tasks, but also outperformed node-link diagrams in finding common-neighbours and counting clusters. The results of our study, which is discussed in Chapter 6, also show that adjacency matrices are better at identifying high-level differences in the structure of networks than node-link diagrams. However, even adjacency matrices have scalability limits and resemble 'out-of-tune-TVs' when used to visualise networks beyond a certain size.

Interactive tools are commonly used to overcome the scalability issues of visualising large and complex data. Yi *et al.* [334] propose seven categories for interaction techniques that are used in visualisation. We discuss these categories further in Chapter 4 and Figure 4.7(b). In the context of network visualisation, arrangements of nodes in node-link diagrams and the reordering of rows and columns in adjacency matrices (*Reconfigure*) aim at placing communities of nodes close to each other to highlight the structure of the network (*Connect*). This degrades smoothly with larger networks to show 'heat map' like representations for adjacency matrices and clustered masses for node-link diagrams.

Other common techniques are to filter out nodes or links (*Filter*), or aggregate them into fewer representative elements, which could be interactively compressed or expanded (*Abstract/Elaborate*). Zooming (*Abstract/Elaborate*) can also be used to show further detail, while keeping parts of the network out of view. Navigation techniques (*Explore*) would then be used to move around the visualisation and change the view as necessary.

Some techniques rely on the user to interact with the visualisation (e.g. [107]), while others rely on automatic highlighting, filtering, or aggregating parts of the network into communities. Generally, communities, also known as clusters or modules, are discovered based on the notion that there should be more connections or shared characteristics between the nodes or links within a community, than between them and external ones in the rest of the network.

Community detection algorithms deal with detecting and partitioning graphs into communities. Algorithms differ in what they consider to be good partitioning criteria. Thus, one of the main challenges for network visualisation is to choose a good clustering to portray important characteristics of the network. Another challenge is the difficulty of finding optimal partitions, thus heuristics and approximations are needed to be able to partition large networks.

Modularity [238] is one of the most popular measures for community detection. It is measured by comparing the number of edges within a community to the expected number of edges in a random graph with a similar degree distribution. The idea is based on the assumption that random graphs are not expected to possess inherent community structures. Clauset *et al.* [81] propose a fast and greedy algorithm that has a running time of $O(|E|d\log|V|)$.

The *Minimum Description Length* (MDL) [265] is another interesting clustering principle, which compresses nodes and edges into fewer elements based on the similar features that they share. Some approaches are based on finding bridges that connect dense communities to the rest of the network; without these bridges the communities would be isolated. *Edge Betweenness* [146] is a measure used to detect bridges. It is measured for each edge by calculating the number of shortest paths between pairs of vertices that traverse it. Kolda and Bader [200] propose an algorithm for community detection using the edge betweenness measure with a time complexity of $O(|E||V|)$. Kirchoff's equation is used to assign voltages to nodes. Nodes with similar voltages would form a community. This can be done in $O(|V|+|E|)$ time [327]. Similarly, label propagation and node colouring techniques can be used to detect communities. Random walks can be used to traverse the network in a stochastic way and determine a *closeness* measure for nodes based on the number of hops needed to reach one from the other. *Walktrap* [254] and *infomap* [269] are popular community detection algorithms that are based on random walks.

Some community detection algorithms are based on the structural characteristics of the communities. Stars, cliques and leaf nodes are the most common structures considered. A clique is a group of nodes in a graph that are fully connected to each other. A star is a tree with one internal node that is connected to all the other leaf nodes. For example, Palla *et al.* [250] suggest using k-cliques as communities. *ASK-GraphView* [16] uses community detection to compress isolated tree-like structures. Koutra and Kang [202] propose a tool called *VoG* that uses MDL to detect cliques and stars. Dunne and Shneiderman [103] propose using compact glyphs to represent cliques and other structural communities.

The k-core decomposition is achieved by recursively removing nodes that have a degree less than k, starting with the lowest degree, and until all nodes have a degree $\geq k$ [54]. Alvarez *et al.* [26] propose using the k-core decomposition to arrange the nodes on the circumference of nested circles. We also use the k-core decomposition in achieving our novel summary representation, which is discussed in Chapter 6.

Link Clustering approaches deal with detecting communities of links rather than nodes (e.g. [193]). *Edge bundling* techniques are used to group several links with similar features (e.g. [121] [100] [43] [324]). There are also approaches with loose definitions of communities [89].

Some approaches are adjustable to use multiple community detection methods. *Hive plots* [205] arrange the nodes on multiple axes so that the nodes within a community are placed on the same axis. *GraphPrism* [185] shows a summary of the properties of the nodes and the links using matrices. The properties in the summary representation can be used to interactively highlight parts of the node-link (detailed) representation of the network.

Many tools are interactive and allow the users to navigate through communities. Eades and Huang [113] propose an interactive visualisation that expands and compresses groups as necessary, while van Ham and Wijk [306] propose a fish eye scheme to interactively

look into groups. *TopoLayout* [29] uses different clustering techniques to compress nodes and edges into meta-nodes and meta-edges.

The nature of the graphs at hand can also play a decisive role in selecting one community detection approach over the other. Many of the approaches mentioned above work well on sparse graphs, but do not detect meaningful communities in dense graphs. Lancichinetti and Forunato [208] conduct a study to compare some of the popular community detection algorithms. The results show that the *Infomap* [269] method performed the best across a wide range of graphs with different characteristics. Lee and Archambault [210] conduct a study, which shows that *Infomap* detects communities similar to what users would identify. Wu *et al.* [328] conduct studies to explore community detection approaches with respect to their ability to show network structure.

We further discuss existing visual representations for large networks in Chapter 6.

2.6 Conclusion

Most algorithms for network layout focus on improving computational complexity. Evaluation methods also focus on aesthetic criteria and layout features that enhance readability.

It is widely believed that node-link diagrams for larger networks are not suited for tasks that require a detailed understanding of the network. This means that for small networks, quality and readability should be the main focus, while for large networks a summary representation should sufficiently show structural information, while hiding the details of the underlying network. An important question is, what is ‘small’ and what is ‘large’, and beyond what size of networks should one move from the detailed representation to the summary representation? We explore these in this thesis.

Chapter 3

High-Quality Ultra-Compact Grid Layout

"and though she be but little,
she is fierce."

William Shakespeare
A Midsummer Night's Dream

As discussed in Chapter 2, there has been a lot of research into what constitutes a good network layout. Most existing tools to create network diagrams of high quality use complex multi-stage pipelines. Attempting to modify these pipeline techniques is very tedious. Furthermore, the layouts produced by them have significant issues. For example, in attempts to create layouts with minimal overlap, the outcome ends up having a lot of wasted space.

As contributions of this chapter, we present a novel ultra-compact and grid-like network layout aesthetic. We also reassess the use of combinatorial optimisation to achieve an optimal drawing of a network according to different layout features. The abundant use of grid arrangements by designers, almost universally in typographical layout, has inspired the former, while the latter is motivated by the advances in combinatorial optimisation.

The second contribution of this chapter is to reassess whether these techniques can be used for high-quality layout of small graphs. While they are fast enough for graphs of up to around 50 nodes, we found these methods are too slow for larger graphs. Our third contribution is a *Large Neighbourhood Search* meta-heuristic approach, that makes these techniques scalable to larger networks.

The work discussed in this chapter was done in collaboration with my supervisors: Tim Dwyer, Karsten Klein, Kim Marriott and Michael Wybrow, and collaborators Steve Kieffer and Graeme Gange. The results were published in *Transactions on Computer Graphics and Visualization* as proceedings from the 2015 *IEEE Conference on Information Visualization* (*InfoVis*), with the exception of Section 3.8.

(a) Grid systems in typographic layout.

(b) With graphic designers playing an increasing role in the design of user interfaces for phone, tablet and desktop operating systems, this traditional grid-based design aesthetic is becoming more popular in these media. A case in point is Microsoft's 'Modern' interface, which seeks to unify app-design across devices.

Figure 3.1: The resurgence of the grid-design aesthetic in new media leads us to re-examine some of the aesthetic assumptions that have been made in designing layout methods for network diagrams.

3.1 Introduction

Computer science researchers (and others) have been exploring different ways to automatically lay out and draw diagrams that represent graphs or networks for many decades. Because of the difficulty of the network layout problem and limited computational power of early computers, the primary focus was on developing fast heuristic techniques with low time complexity. As computer power rapidly increased through the '90s and 2000s, many researchers continued to focus on fast heuristic techniques in a race to see who could untangle the biggest graphs.

Other research led to the development of complex multi-stage layout frameworks for higher quality layout of smaller networks. For example, a seminal paper by Batini *et al.* [56] proposed a multi-stage layout framework called *Topology-Shape-Metrics* (TSM), which led to the development of orthogonal *graph drawing* techniques, designed to produce drawings with orthogonal connectors. Another family of multi-stage approaches arose following Sugiyama *et al.* [293], specifically for layered-layout of directed graphs.

One feature common to the layouts produced by all of these different algorithms for network layout is the use of white space to clearly separate nodes and an implicit visual emphasis on edges rather than nodes. This leads to relatively sparse layouts in which most of the display space is empty. The main contribution of this chapter is to investigate a new network layout aesthetic based on *Ultra-Compact Grid Layout*.

This new aesthetic is motivated by the grid arrangements that are used almost universally by designers in typographical layout, and are increasingly common in other media such as computer interfaces (Figure 3.1). Layout in this tradition is built upon a grid that divides the viewing space into regular cells. Individual elements may span multiple grid-cells but—as much as possible—the grid subdivisions are respected. This approach provides a regularity to the layout that leads the eye in a familiar and comfortable way [232]. Recent studies of network layout have shown that grid arrangements are memorable [222], when people arrange small diagrams themselves they prefer to place nodes

at grid-points, [263] and such placements are also preferred to TSM-based orthogonal layout [191].

As we have mentioned, TSM-based orthogonal layout techniques were also influenced by a grid-aesthetic, though probably more due to circuit layout traditions than typography. As Figure 3.5 shows, our new aesthetic leads to a very different visual style. It uses ultra-compact arrangement on a grid with group membership shown by containment in nested rectangular regions. This containment then enables the use of the 'powergraph' approach [270] to collapse edges. In such an *edge-compressed* view [104] an edge from a group to another group implies all nodes in the first group are connected to all nodes in the second. Unlike virtually all existing approaches to 'powergraphs', or network diagrams with clustering, we do not require that groups are hierarchical, making our approach applicable to other types of diagrams such as representations of overlapping set-membership (Figure 3.6).

After exploring grid-design, this chapter's second contribution is to investigate practical methods for producing *Ultra-Compact Grid Layout* of the *highest-possible quality*. Rather than developing a specialised algorithm, we decided to explore more general purpose optimisation-based approaches. One reason for this is that current frameworks for high-quality network layout separate the layout into a pipeline of different steps and, as a consequence, the resulting layouts are often compromised because of the fixed trade-off between aesthetic criteria imposed by the pipeline. Furthermore, the implementations of these multi-stage pipeline methods are complex and brittle, as discussed before in Chapters 1 and 2 and, later in Section 3.3.

Since the time when pipeline-based graph-layout methods such as TSM were conceived, generic technologies for solving combinatorial and mixed-integer optimisation problems have improved by several orders of magnitude. Simultaneously, the computing power available on average desktop machines has increased exponentially. Optimisation problems that once took weeks to solve with home-sized computers can now be solved in seconds on current home computers.

We feel then, that the time is right to reassess network layout and see whether these general-purpose optimisation techniques can be usefully applied to solve simple mathematical models encoding such layout problems. One advantage of using such a generic approach is that it allows us to readily explore this new aesthetic, by allowing us to rapidly create examples from different applications and with different aesthetic trade-offs (Section 3.2).

Thus, after developing a model for *Ultra-Compact Grid Layout* (Section 3.4), we compare the applicability of several generic optimisation techniques (MIP, CP and SAT) to this problem (Section 3.5). While useful for exploring the design of the layout model, we found that even the best of these solving technique was only practical (in terms of running time) for graphs of up to around 50 nodes.

A common fall-back for solving difficult combinatorial optimisation problems is to use generic meta-heuristic techniques like tabu search, simulated annealing or genetic programming. Dozens of different techniques have been proposed. While not guaranteed to find an optimal solution, they are routinely used to find 'good' solutions to problems that are too hard to solve optimally using MIP, CP or SAT. We therefore developed a meta-heuristic to solve our layout problem.

We decided to use *Large Neighbourhood Search* (LNS) [19, 253](Section 3.7). This class of meta-heuristic is currently *de rigueur* for solving various transportation and scheduling problems. While we are not the first to try generic meta-heuristic approaches

for network layout, we are the first to consider LNS. Though not guaranteed to find an optimally compact solution, our evaluation shows that our LNS heuristic found reasonably good layouts (when compared to the optimal layout) and scaled to graphs with 100 nodes.

In summary, the technical contributions of this chapter are to:
- Introduce a new *Ultra-Compact Grid Layout* for networks and explore the design-space (Section 3.2);
- Present a declarative model of layout goals and constraints that allows us to rapidly evaluate different refinements and applications of these aesthetic criteria, solvable using generic constrained optimisation techniques and without the need for specialised algorithm development (Section 3.4);
- Compare the efficiency of different generic optimisation techniques (MIP, CP, SAT) for solving our declarative model (Section 3.5);
- Explore the use of a *Large Neighbourhood Search* based meta-heuristic to solve this declarative model. This would allow us to obtain compact grid layouts for graphs of up to 100 nodes in less than five minutes (Section 3.7).

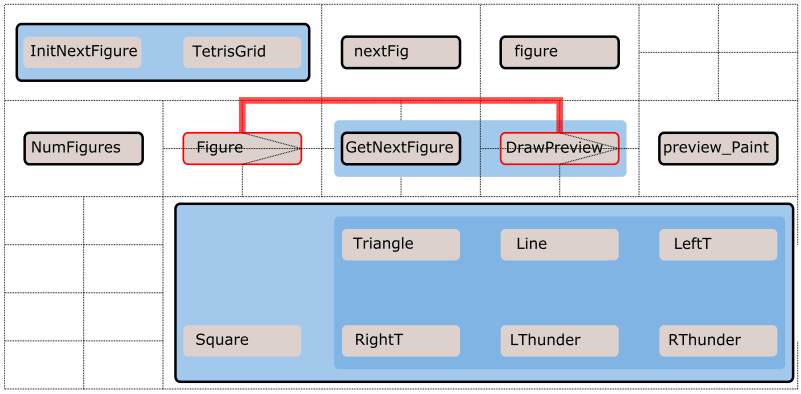
This chapter is structured in the following way. We present a new grid-based layout aesthetic for network diagrams in Section 3.2. We also provide some usage examples of our novel aesthetic. In Section 3.3, we present some related work. In Section 3.4 we present a declarative model of the network layout problem to achieve our proposed aesthetic. Section 3.5 provides details of how this model was implemented using three generic optimisation techniques: *Mixed Integer Programming* (*MIP*), *Constraint Programming* (*CP*), and *Boolean Satisfiability Problem* (*SAT*). We proceed by evaluating and comparing three generic solvers that can solve our three implementations, namely *CPLEX* for *MIP*, *Gurobi* for *CP*, and *BumbleBEE* for *SAT*, in Section 3.6. We evaluate the use of LNS to achieve better running time in Section 3.7. Finally, in Section 3.8, we present additional exploration of ways we can enhance the solving time of our techniques.

3.2 Design

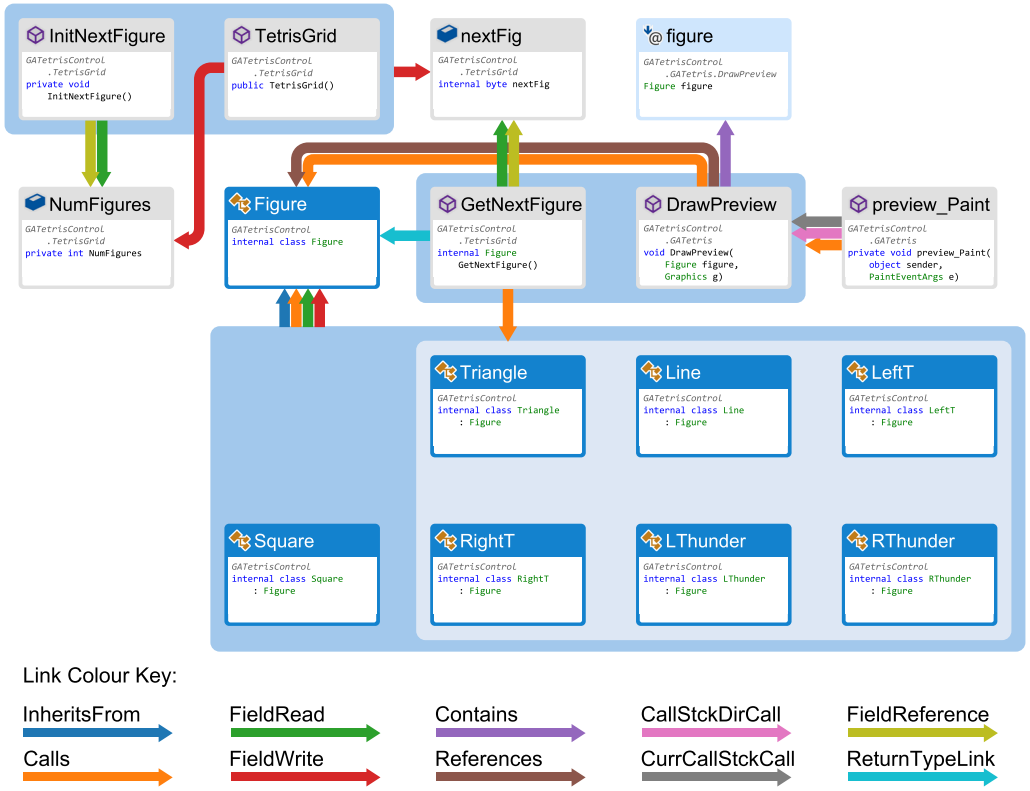
In this section we present a new layout aesthetic for network diagrams that is based on grid layout in typography, and we provide a number of motivating examples. The aesthetic incorporates the following layout requirements.

R1 - Node Content Emphasis

Many applications have more than just simple labels associated with nodes, such as rich graphics or text in paragraph or tabular form. In typography, grid-cells are packed quite densely in order to maximise the area devoted to this content. By contrast, orthogonal network diagram layouts are typically very sparse, devoting more space for edge paths, which—in order to minimise bends and crossings—may be very long. Networks featured in Figures 3.1, 3.3, 3.7(b) and 3.2 all contain significant text and graphic content associated with each of the nodes. For this detail to remain readable at reasonable scales, without resorting to interactive focus-and-context techniques (e.g. [180]), *compact node-placement is essential*. A strong correlation between human preference and layout compactness is also observed in a recent study by Kieffer *et al.* [191].



(a) The details of the routing process. This example shows an edge being routed from the 'DrawPreview' node to the 'Figure' node. The path of the edge needs to avoid crossing the nodes with thick borders. The segments of the edge that intersect with these obstacles are removed from the routing graph. 0-cost port connections are visible on the source and target nodes.



(b) After the routing is complete, the chosen edge paths are bundled and separated within the available channel space.

Figure 3.2: A software-dependency network with the routing detail and the final result. This example was solved in 0.732 seconds with a SAT solver (*BumbleBEE*). This network shows dependencies between types, methods and properties in C♯ code and was obtained in a debugging scenario using the Visual Studio *Code Map* tool. This layout neatly illustrates the cause of the bug: that *Square* is the only sub-class of *Figure* not created by the *GetNextFigure* method. Code snippets and icons on each of the nodes give added context, again illustrating the need for node content emphasis.

R2 - Proximity Implies Connectivity

If we are to devote less space to edge paths in our grid arrangements then we must rely more on the proximity of nodes to indicate connectivity. Recent studies have shown that layout that achieves such proximity is strongly preferred by readers of small diagrams [106]. In addition, this objective *indirectly* addresses crossings simply because shorter edges are less likely to cross. Minimising edge-length can sometimes be a more successful strategy for minimising crossings than heuristics, which directly address crossings, e.g. [105].

R3 - Variable Node Dimensions

Some nodes may have significantly more content than others. Following typographical layout conventions, these nodes can be expanded to fit their content, but they must always fully fill a rectangular set of grid-cells. Furthermore, where different orientations of the node are possible (e.g. picture beside text or picture below text), the layout should choose the orientation to best suit the layout. Figure 3.3 demonstrates layout with variable node orientations.

R4 - Containment

The semantics of many applications involve representing group membership over sets of nodes. In typography, such relationships are shown through nested rectangular enclosing regions.

R5 - Flow

In applications where the directionality is important, we would like flow to be shown in multiple directions, for example, left-to-right *and* top-to-bottom, as in document layout.

Part of the motivation for this work was the search for an effective layout method for edge-compressed dense, directed networks [104]. Without compression, graphs that have only a few nodes but many edges are already very difficult to read. For example, Figure 3.4 shows the state-chart for a travel booking system with 13 nodes and 44 edges arranged using the commercial layout software, yFiles [15].

Figures 3.5(a) and 3.5(b) show the edge-compressed version of the network with only 17 edges. In the compressed representation, an edge between two groups implies a *bi-clique*. That is, every node contained in the source group of the edge is the source of an edge to every node in the target group. Thus, precisely the same connectivity structure is conveyed but in a less cluttered way. Further, the grouping inferred by the edge-compression reveals structure; it is obvious from the single edge adjacent to the largest group, and the *trip cancelled* state that every state other than *start* is cancellable.

Figures 3.5(a) and 3.5(b) compare layouts obtained by a standard TSM approach (yFiles) and by our model. Our layout model here keeps connected nodes close together (**R2**) while preserving group containment within rectangular regions (**R4**). Furthermore, the layout is as compact as possible while respecting node and group containments, thus

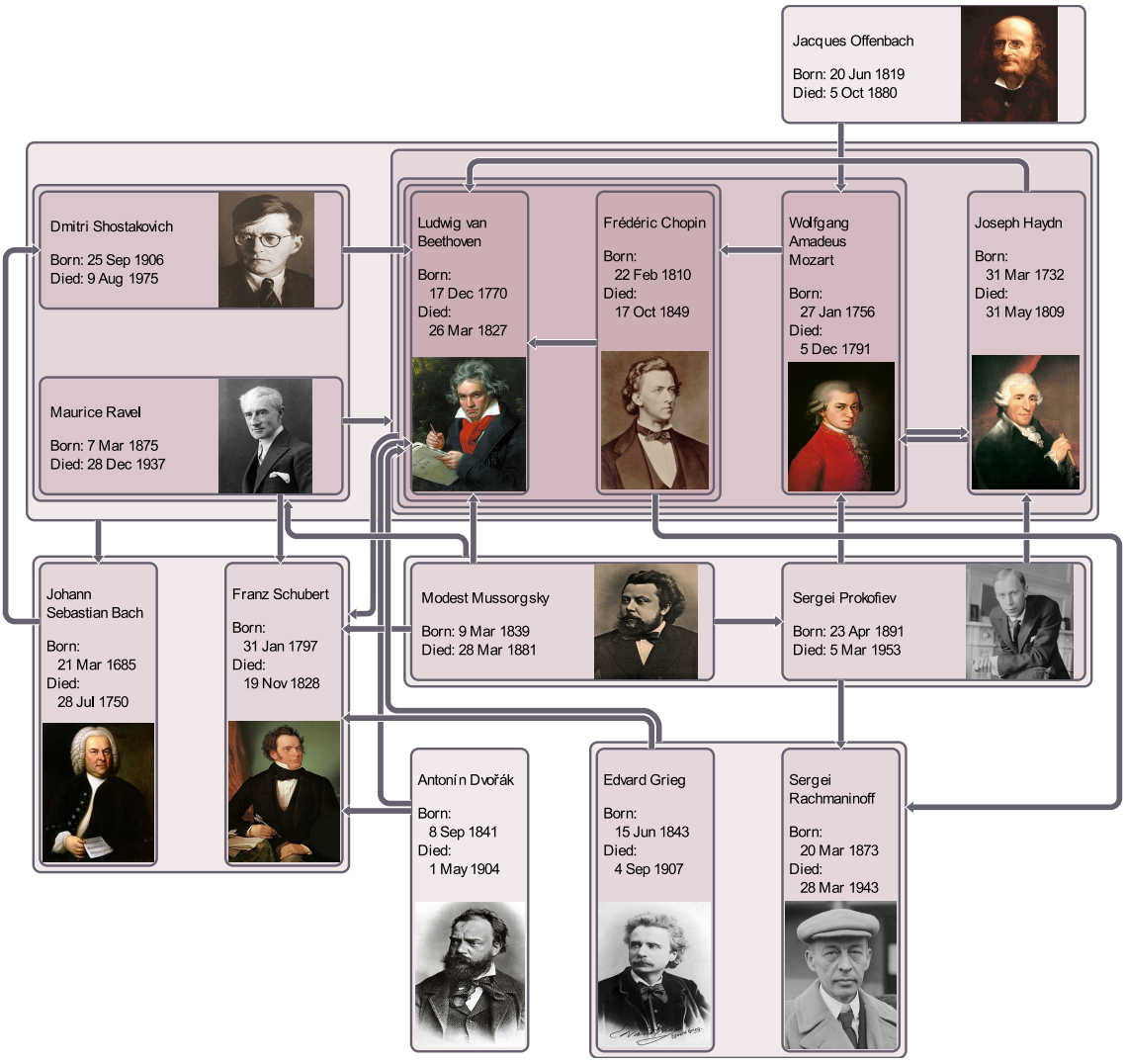


Figure 3.3: Links between major composers arranged with our model with the solver choosing the best orientations for nodes. Layout took 37.422 seconds using the SAT solver — disjunctions due to variable node orientations expand the search space.

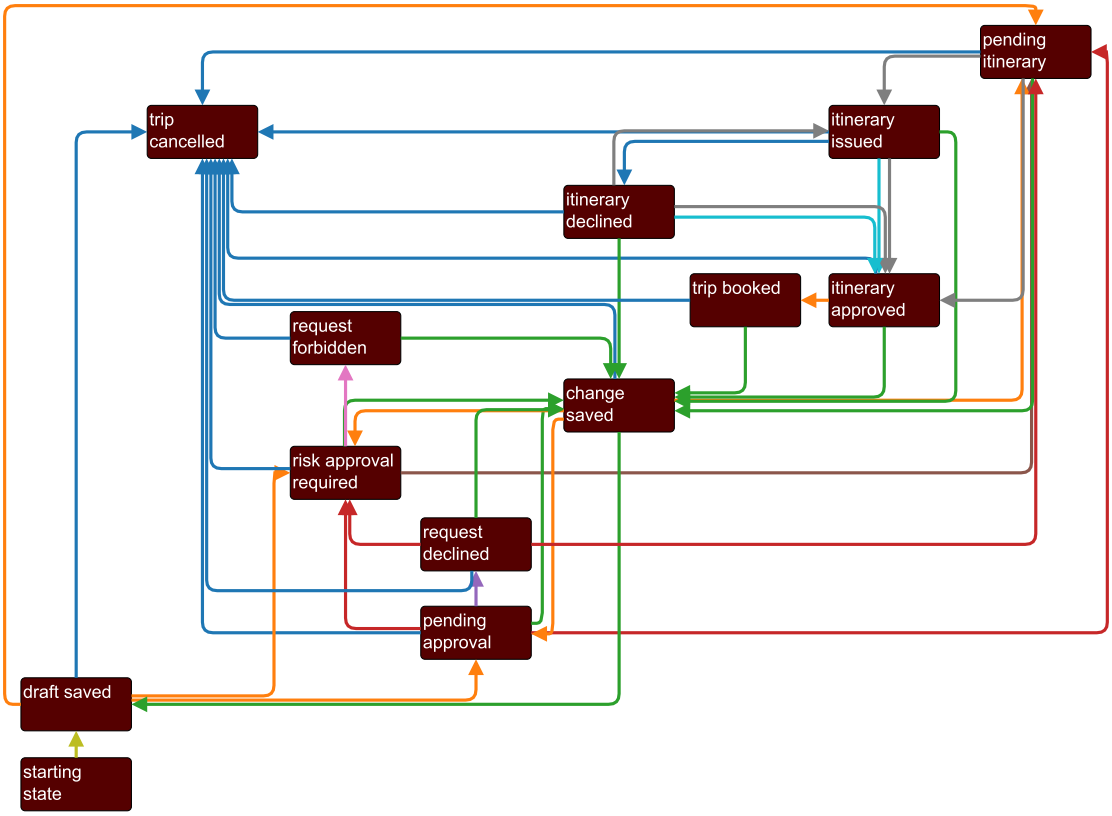
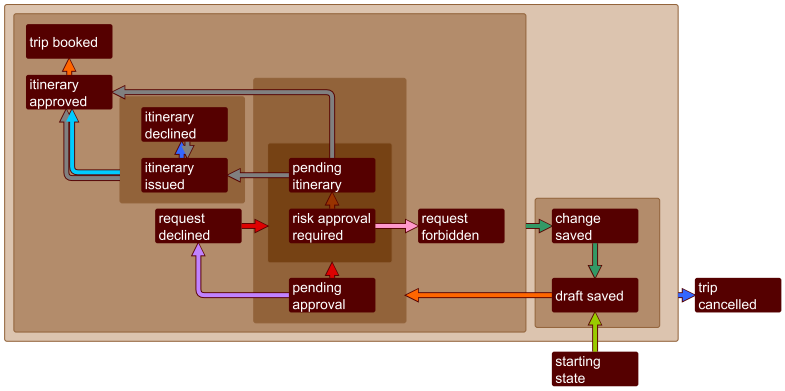
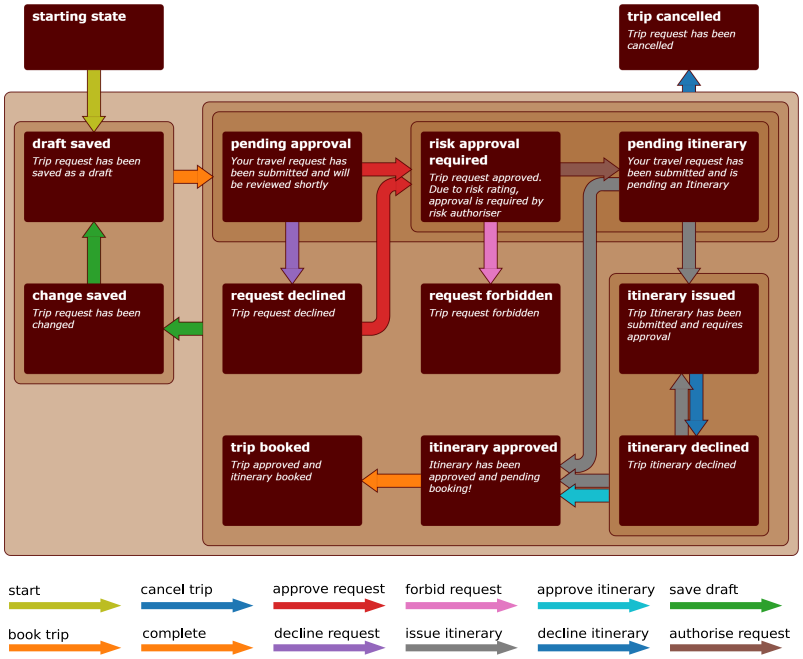


Figure 3.4: A traditional TSM-based orthogonal layout of the flat network. This diagram was produced using the commercial layout software yFiles [14]. The network represents the different steps of a trip booking process. The nodes represent the state of the booking, while the links represent a transition for one state to another.



(a) The *powergraph* resulting from the edge compression of the original state-machine of the booking system drawn using a TSM-based orthogonal layout. This diagram still has a lot of wasted space. If it was coerced to a grid, then the dimensions would be 6×7, leaving 29 empty grid-cells.



(b) The same *powergraph* of the state-machine drawn using our *Ultra-Compact Grid Layout*. The diagram has 4×4 grid dimensions, with only three empty grid-cells. As seen from the figures, the nodes in this diagram are much larger, allowing us to include more detailed descriptions of each state, due to the optimal use of space. This optimally compact solution was found in 0.464 seconds using the SAT solver *BumbleBEE*. Although we do not explicitly consider acquiring a minimal number of edge bends and crossings, our layout has an equal quality in these respects to the TSM-based orthogonal layout shown above. Moreover, our layout significantly reduces the overall area and total edge-length.

Figure 3.5: Drawings of the state machine for a travel-booking system. An edge compression technique is used to compress the 44 original edges into 14 edges. The resulting graph is called a *powergraph*. Both drawings of the *powergraph* show a clearer structure than the drawing of the original network (without the edge compression), which is shown in Figure 3.4. For example, the outermost compartment makes it obvious that all states apart from 'starting state' are cancellable, i.e. have a link to 'trip cancelled'.

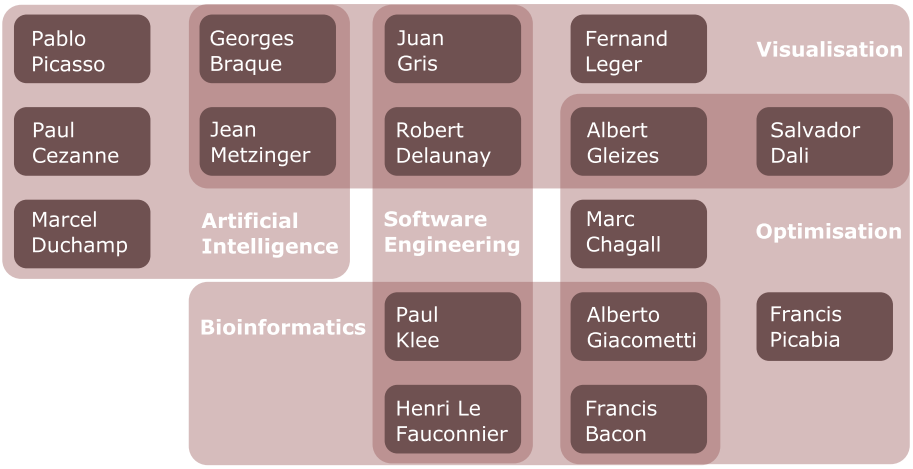


Figure 3.6: An example where group members can belong to more than one group. This Euler diagram shows researchers of a lab (anonymised). The groupings are based on their research interests. For example, 'Salvador Dali' is interested in both 'Visualisation' and 'Optimisation'. The problem of arranging this network into an optimal diagram, based on our described requirements, was solved in 0.047 seconds using the SAT solver *BumbleBEE*.

maximising space for, and hence readability of, node labels (**R1**). When node area is maximised in this way, we are able to include additional explanatory content for each node and the diagram becomes a more complete, stand-alone description of the state-machine.

Figure 3.3 demonstrates the possibility to provide a very compact layout for a graph with nodes that require more than a single grid-cell to fit their content (**R3**). The network is a section of the 'Composers Graph' that was one of the challenges for the 2014 Graph Drawing Conference contest [1]. Each node is a composer for whom we want to show both biographical details and a portrait. We allow the textual biographical details to fill one grid-cell, while the portrait can go in an adjacent cell, either beside or below the text. The solver automatically chooses the orientation of each node that permits layout that is optimal with respect to the other layout requirements.

Figure 3.6 is an Euler diagram representing the research interests of members of our lab (anonymised). Set labels are also treated as nodes and laid out within the same grid system. Note that the containment (**R4**) is no longer a strict hierarchy, yet our general layout model is still applicable. Drawing Euler diagrams under certain constraints, such as convexity of the regions, is not always possible. Actually, realising the drawing in an aesthetic way is a further challenge. Both of these problems have seen a lot of interest from computer-scientists and mathematicians, and sophisticated algorithms have been developed [267, 278]. Here, we have defined the layout for rectangular boxes with a relatively simple declarative model, and left both the problems of determining feasibility and (if possible) placement to the solver.

In Figure 3.7 we compare two different arrangements of a biological pathway network. In such, pathways the direction of the edges is often very important, for example, indicating the direction of a reaction. It is therefore a common convention to show flow in such diagrams from top-to-bottom or from left-to-right. Figure 3.7(a) uses a standard 'Sugiyama style' [293] layout obtained, again, with yFiles. This method assigns nodes to layers, such that edges exclusively span layers. By contrast, Figure 3.7(b) introduces a disjunction constraint allowing edges to flow *either* left-to-right or top-to-bottom.

Figure 3.2(b) shows a software-dependency graph. This network shows dependencies between types, methods and properties in C♯ code and was obtained in a debugging scenario using the Visual Studio *Code Map* tool. This layout neatly illustrates the cause of the bug: that *Square* is the only sub-class of *Figure* not created by the *GetNextFigure* method. Code snippets and icons on each of the nodes give added context, again illustrating the need for node content emphasis (**R1**).

3.3 Related work

The most widely-used family of automatic layout methods for undirected graphs are based on *force-directed* layout [197]. These methods iteratively place nodes such that edge-lengths become relatively uniform, while disconnected nodes are spaced further apart. This approach is attractive because the basic variants are easy to implement. In addition, force-based representations reveal the structure of small graphs, and cluster nodes in a certain way, so that proximity implies connectivity (**R2**). However, force-based representations are very organic — the antithesis of grid layout.

Rohrschneider *et al.* [268] tried to overcome that problem for biological networks by first computing a stress-based node-placement on a grid, followed by an edge routing heuristic. This approach does not allow group information to be taken into account, and the graph structure is hard to discern from the layouts. Recent work from Kieffer *et al.* [190] explored augmenting the objective function of a constrained force-directed technique to prefer nodes placed at grid-points, thereby creating a compromise between a grid-aesthetic and **R2**. This method (extended to grouped graphs) provides the starting point for our LNS approach.

A layout exploration in the specific domain of Metro-map layout from Nöllenburg and Wolff [242] was similar to ours in spirit in its attempt to obtain high-quality layout through the use of optimal (MIP) techniques. However, the metro-map layout problem is significantly constrained in that the topology is already given by the geographical positions of the stations. As a layout adjustment problem, rather than completely free arrangement of nodes, it is therefore more similar in terms of tractability to the LNS approach explored in Section 3.7.

Orthogonal layout approaches seek to represent edges with axis-parallel segments, preferably with only a small number of right-angle bends. From the approaches that were proposed, the planarisation-based *Topology-Shape-Metrics* (*TSM*) framework [56] has proven to be by far the most successful in practice. TSM first fixes an embedding for the planarised input graph, then solves bend minimisation for this embedding to achieve an orthogonal shape, and in the final compaction step, computes node positions for this shape.

However, TSM has limitations that inspire the work presented in this chapter: Primacy is given to minimising edge crossings, prohibiting a good compromise between aesthetic criteria, and even optimal solutions for a single phase (e.g. calculated using ILP, MIP or SAT strategies [62, 133, 195]). This will generally not lead to a solution close to the optimum, with respect to all optimisation criteria (see Figure 3.4). Orthogonal methods also typically do a poor job of handling nodes of widely varying dimensions, and they are difficult to extend for main constraints in applications, in particular grouping of nodes and flow direction. An exploration of applying optimal methods (ILP and SAT) to particular graph-theoretical problems related to orthogonal-drawings was considered by Biedl *et al.* [66] but their results are not readily applicable to practical layout. Betz *et al.* [64]

integrate upward crossing minimisation into a TSM approach to support non-uniform node heights for layouts of directed graphs.

A number of other researchers have investigated the use of meta-heuristic approaches to graph layout. Davidson and Harel [91] investigated the use of simulated annealing (SA) for undirected graphs. Harel and Sardas [155] used SA to beautify a layout drawn with a planarisation-based approach. Barsky *et al.* [52] and Kojima *et al.* [199] propose methods based on SA and local-search (respectively) for biological networks. The work of Barsky *et al.* is the most similar because they lay out the nodes on a grid, however, the layout is not particularly compact and does not use rectangular compartments. To our knowledge, ours is the first use of LNS for graph layout.

3.4 Layout Model

In this section we present a high-level declarative model for placing grouped nodes in a grid layout that formulates the problem as a constrained optimisation problem.[1]

3.4.1 Node-Placement Model

The high-level model for node-placement takes the following as input:

1. The set of leaf or *base* nodes $B = \{1, ..., n_B\}$ and the set of *container* nodes $C = \{n_{B+1}, .., n_C\}$ which contain groups of other nodes.

2. A fixed width w_u and height h_u for every base node $u \in B$. These are positive integers.

3. Every container node has a set of nodes that are contained inside it. These can be container or base nodes. This is specified by the Boolean matrix $con[u,v]$ which is true iff $v \in B \cup C$ is inside $u \in C$. The containment relationship need not be hierarchical.

4. The containment relationship gives rise to a non-overlap relationship between nodes. For convenience, this is pre-computed and passed into the model. It is given by the symmetric Boolean matrix $disj[u,v]$ which is true if $u, v \in B \cup C$ should not overlap.

5. For each pair of nodes $u, v \in B \cup C$ there is a non-negative desired distance $dd[u,v]$ between them with a non-negative weight $ddw[u,v]$. The weight $ddw[u,v]$ is 0 if u is contained in v or vice versa.

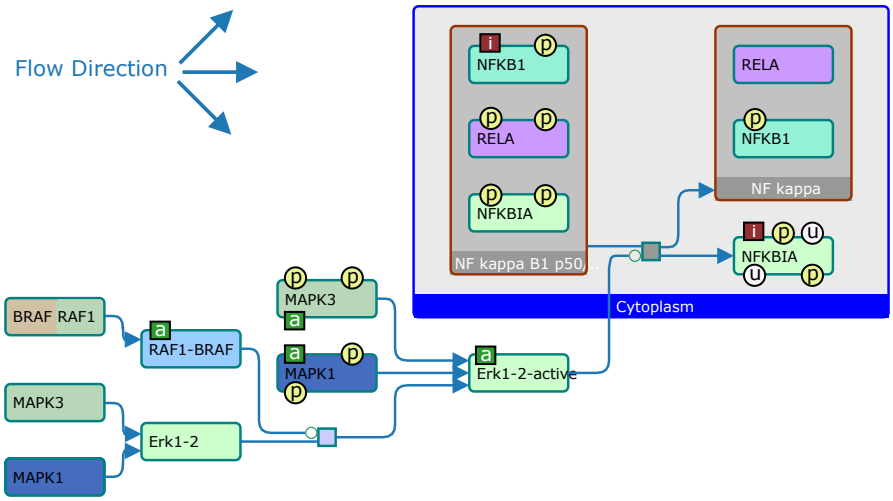
6. A maximum grid size, g_x and g_y, both of which are positive integers big enough to ensure that they contain the optimal layout.

Neither *con* nor *disj* need to contain redundant constraints: for efficiency they should be minimal.

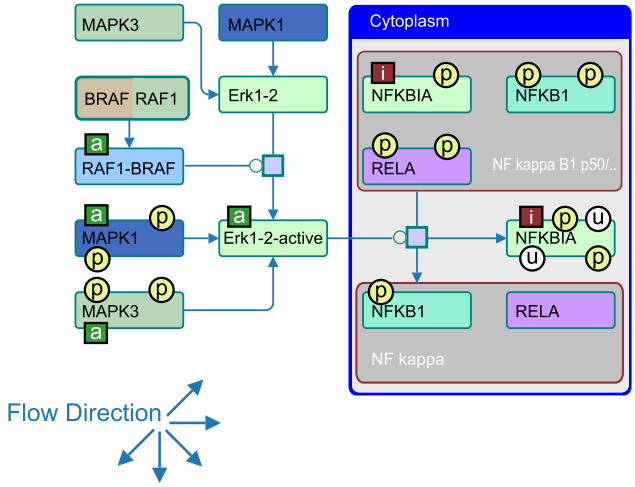
We experimented with different desired distances and weights. Following *stress*-based methods [134], we tried setting the desired distance between two nodes to the graph-theoretic-distance, taking into account containment[2]. We also tried simply setting the

[1] The full model in MiniZinc is available under an open-source license [236]

[2] This is the length of the shortest path between the nodes in an extended graph where there is an edge between two nodes x, y, in this extended graph if there is an edge in the original graph, or if $con[x,y]$ or $con[y,x]$ holds.



(a) Sugiyama style layered-layout with six layers and flow direction strictly left-to-right.



(b) Compact grid layout obtained with relaxed flow direction; solved in 2.035 seconds using the SAT solver.

Figure 3.7: A directed biological pathway from `http://www.pathwaycommons.org`.

desired distance and weight to the edge adjacency matrix. Observing similar results for both approaches, we opted for the latter.

Variables and constraints:

1. The core decision variable in our model is the position $(xs[u], ys[u])$ of the top-left corner of each base node $u \in B$. This must be a point on the grid: $xs[u] \in \{1, \ldots, g_x\}$ and $ys[u] \in \{1, \ldots, g_y\}$ where g_x and g_y give the size of the grid.

2. The position of the bottom-right corner of each base node is functionally dependent upon this: $\forall u \in B$, $xf[u] = xs[u] + w[u]$ and $yf[u] = ys[u] + h[u]$.

3. We require that the whole node fits on the grid: $\forall u \in B$, $xf[u] \leq g_x$ and $yf[u] \leq g_y$.

4. The position, width, and height of the container nodes are also functionally dependent on the position of the base nodes; as the containers are just the bounding box of their constituents, so $\forall u \in C, v \in B \cup C$:

$$\begin{aligned} xs[u] &= \min\{xs[v] \mid v \in B \cup C \wedge con[u,v]\} \\ xf[u] &= \max\{xf[v] \mid v \in B \cup C \wedge con[u,v]\} \\ w[u] &= xf[u] - xs[u] \end{aligned}$$

   and similar in the y-dimension.

5. The following disjunction ensures that nodes do not overlap: $\forall u < v \in C$ s.t. $disj[u,v]$,

$$xf[u] \leq xs[v] \vee xf[v] \leq xs[u] \vee yf[u] \leq ys[v] \vee yf[v] \leq ys[u].$$

Objective function to be minimised:

$$stress + \alpha cc + \beta oc$$

α and β are fixed weights and the functions *stress*, *cc* and *oc* measure different aesthetic criteria, as follows.

The *stress* term is the difference between the desired and actual distance between the nodes. Because we are using orthogonal connectors and grid layout, we use Manhattan distance. We measure the distance between the closest points on the perimeter of the nodes rather than between the centre of the nodes, as this leads to considerably better layout in the case that the nodes are not squares. To compute this we use the functionally dependent variables:

$$dx[u,v] = \begin{cases} xs[v] - xf[u] + 1, & \text{if } xf[u] \leq xs[v] \\ xs[u] - xf[v] + 1, & \text{if } xf[v] \leq xs[u] \\ 0, & \text{otherwise.} \end{cases}$$

We define $dy[u,v]$ symmetrically. Now,

$$stress = \sum_{u,v \in B \cup C} ddw[u,v] \cdot |dx[u,v] + dy[u,v] - dd[u,v]|\,.$$

The other components of the objective function are designed to ensure $\forall u \in B \cup C$ that compartments are compact $cc = \sum_{u \in C} w[u] + h[u]$, that the entire layout is compact $oc \geq xf[u]$, and that it fits inside a rectangle with a given aspect ratio ar: $yf[u] \leq ar \cdot oc$.

A great advantage of using a constrained optimisation approach is that it is straightforward to add additional constraint encodings based on additional aesthetic criteria. Thus, for the example presented in Figure 3.3, we add a constraint to dictate a fixed perimeter of size 2×1 for base nodes; placed either horizontally or vertically. In the flow layout in Figure 3.7 each source node should be either above or to the left of the destination node — another disjunction. Note that the optimisation problem that we tackle with our model is NP-hard, as can be shown by reduction from the rectangle packing problem [201].

Initially we tried to use a single constrained optimisation model for both node-placement and edge routing. This modelled each edge using a fixed number of horizontal and vertical segments (some of which could be 0 length) and including a penalty term for each

pair of segments to penalise possible edge crossings. However, this proved too slow for all but very small networks, and so is currently still not practical. We therefore developed a separate heuristic algorithm for edge routing given the positions of the nodes on a grid.

3.4.2 Routing

The grid-aesthetic naturally suggests routing connectors between base and container nodes in an orthogonal-style, i.e. with straight-line segments aligned to the grid. Obviously, intersections between these orthogonal edge paths and node boundaries (other than the source/target of the edge) should be avoided. Edge paths should only intersect container boundary rectangles if the container is the source or target, or is an ancestor of the source or target.

Currently, the standard for orthogonal connector routing is to route over an orthogonal visibility graph, followed by a 'nudging' phase to centre edge segments in channels between nodes [329]. Strict grid placement of nodes simplifies this problem considerably, as we can route over the graph formed by the grid itself, intersected with the centre lines between each column and row of nodes (see Figure 3.2). When we route an individual edge, we remove any edge segments which intersect nodes other than the start and end nodes, or segments intersecting containers which are not ancestors of the start and end nodes.

We also connect ports on the start and end nodes to each other. These port connections have zero cost in the subsequent shortest path finding problem between the start and end nodes. Thus, the route will always run through the ports providing the shortest path between source and target. Otherwise, the cost of traversing each segment in the shortest path traversal (Dijkstra) is simply the length of that segment, plus an additional penalty if traversing the edge would add a bend to the current path.

Finally, bundles of co-linear edge segments are constructed and an ordering within bundles that avoids unnecessary crossings is found, as suggested by Nöllenburg [241]. This ordering is used to generate constraints for a simple quadratic program subject to separation constraints (with solution as per [108]), to neatly space the edge segments in the available channels.

3.5 Optimal Node-Placement

The declarative model is a complete and precise mathematical formulation of the node-placement problem. In this section we evaluate three of the most widely used, generic techniques for solving such discrete constrained optimisation problems: *Mixed-Integer Programming* (MIP), *Boolean Satisfiability* (SAT), and *Constraint Programming* (CP). These are all guaranteed to find an optimal solution to the problem. In this section we detail the exact encodings used, as well as the experimental evaluation.

3.5.1 Constraint Programming

Subject to minor syntactic changes, the model given in Section 3.4 is a MiniZinc [236] model. The actual MiniZinc is shown in Appendix A. Thus, it can be directly executed and solved using any of the underlying solvers supported by MiniZinc.

For our evaluation we used a state-of-the-art constraint programming solver, G12/CPX [129] which utilises lazy clause generation. CPX, like most constraint programming

solvers, provides global constraints for finding the minimum and maximum elements in a list or array, and a predicate or function to compute the absolute value of a function and disjunctions of constraints. The calculation of distance between two nodes u, v in a given dimension was encoded as the expression

$$dx[u,v] = \max([0, xs[v] - xf[u] + 1, xs[u] - xf[v] + 1])$$

3.5.2 SAT

SAT solvers are designed to find values for Boolean variables that satisfy conjunctions of clauses, that is, disjunctions of Boolean literals. When encoding an integer problem into SAT, each integer variable $x \in [1,\ldots,n]$ is usually encoded as a set of Boolean variables $[x^1,\ldots,x^n]$. There are two standard encodings for integer variables with small domains.

The *sparse* encoding requires that exactly one of the n variables is true; this gives the semantics

$$x^i \equiv [\![x = i]\!]$$

where this is read as the Boolean variable x^i in the encoded SAT model is true iff $x = i$ holds in the original integer programming model. The direct encoding of this semantic requires $O(n^2)$ clauses, since in addition to the set of disjunctions that ensure that at least one is true, it also requires an encoding that ensures that at most, one value holds. Each pair of distinct values (i, j) requires a disjunction. There are $n(n-1)/2$ such pairs, resulting in $1 + n(n-1)/2$ clauses.

The alternative *unary* encoding of integer variables instead ensures that the x^i are ordered; that is, $x^i \Rightarrow x^{i-1}$. These literals then have the semantics

$$x^i \equiv [\![x \geq i]\!].$$

In addition to requiring only $O(n)$ clauses, the unary encoding is convenient for encoding a range of arithmetic constraints.

Example 1 *Using the unary encoding, $x \leq y$ can be encoded as*

$$\bigwedge_i [\![x \geq i]\!] \rightarrow [\![y \geq i]\!] \equiv \bigwedge_i \neg x^i \vee y^i.$$

Example 2 *Using the unary encoding, $x = |y|$ can be encoded as*

$$\bigwedge_{i \geq 0} [\![x \geq i]\!] \leftrightarrow ([\![y \geq i]\!] \vee [\![y \leq -i]\!]) \equiv \bigwedge_{i \geq 0} x^i \leftrightarrow (y^i \vee \neg y^{1-i}).$$

Example 3 *$x = \max(y_1,\ldots,y_n)$ can be encoded as:*

$$\bigwedge_i ([\![x \geq i]\!] \leftrightarrow \bigvee_j [\![y_j \geq i]\!]) \equiv \bigwedge_i (x^i \leftrightarrow \bigvee_j y_j^i).$$

Because of these advantages we use the unary encoding. Linear arithmetic constraints, such as $x = \sum c_i y_i$, can be implemented using a range of encodings, such as BDDs, adders or cardinality networks [40, 85, 115].

Reified versions of these constraints can also be easily constructed. A *reified constraint* is of form $b \leftrightarrow C$ and constrains the Boolean b to be true iff the constraint C

holds in the model. Reification is a standard technique used to encode disjunctions of constraints: the disjunction $C_1 \vee C_2$ is encoded as

$$(b_1 \leftrightarrow C_1) \wedge (b_2 \leftrightarrow C_2) \wedge (b_1 \vee b_2).$$

Given these primitives, we can straightforwardly encode the model into SAT. For example, the non-overlap of nodes u and v becomes

$$\left(\begin{array}{ll} & (b_{left} \vee b_{right} \vee b_{above} \vee b_{below}) \\ \wedge & b_{left} \leftrightarrow xf[u] \leq xs[v] \\ \wedge & b_{right} \leftrightarrow xf[v] \leq xs[u] \\ \wedge & b_{above} \leftrightarrow yf[u] \leq ys[v] \\ \wedge & b_{below} \leftrightarrow yf[v] \leq ys[u] \end{array}\right)$$

The encoding of the problem into SAT was performed using the *Ben-Gurion University Equi-Propagation Encoder* (BEE) [226], which compiles a declarative specification to SAT. Primitive arithmetic constraints are encoded using direct unary adders; whereas, larger sums, such as the objective value, are encoded with *odd-even sorting networks*. The optimisation is handled by solving a sequence of SAT instances. The solver initially solves the problem P and returns a solution $o = k$. Then it adds the constraint $(o < k)$ to the model and solves again. This is repeated until the resulting problem is found to be unsatisfiable. Hence, the last solution found is optimal.

3.5.3 MIP

The MIP encoding is the most complex among the approaches we compare. We use the standard MIP encoding of minimum and maximum and absolute value [323]. We used six matrices of binary variables to keep track of the relative position of each pair of vertices u, v. The arrays $left[u,v]$, $xoverlap[u,v]$, $right[u,v]$ encode that u must be to the left, horizontally overlap, or must be to the right of v; and analogously in the y direction we have $below[u,v]$, $yoverlap[u,v]$, $above[u,v]$. The following constraint enforces the desired relationships in the x-direction, a similar constraint is used for the y-direction: $\forall u < v \in B \cup C$

$$\begin{array}{l} lt(xf[u], xs[v], left[u,v]) \wedge lt(xf[v], xs[u], right[u,v]) \wedge \\ lt(xs[u], xf[v], xoverlap[u,v]) \wedge lt(xs[v], xf[u], xoverlap[u,v]) \end{array}$$

where $lt(x_1, x_2, b)$ enforces that $b \rightarrow x_1 \leq x_2$ and has the standard MIP encoding $-M * (1-b) + x_1 \leq x_2$ where M is a sufficiently large constant.

Using these it is simple to encode non-overlap and compute the distance between nodes:

1. The relative positions are mutually exclusive in each direction: $\forall u < v \in B \cup C$,

$$\begin{array}{l} (left[u,v] + xoverlap[u,v] + right[u,v] = 1) \wedge \\ (above[u,v] + yoverlap[u,v] + below[u,v] = 1) \end{array}$$

2. If there is a containment relationship between two nodes then they overlap in both directions: $\forall u < v \in B \cup C$ s.t. $con[u,v] \vee con[v,u]$ then $xoverlap[u,v] = 1 \wedge yoverlap[u,v] = 1$.

3. Enforce non-overlap: $\forall u < v \in B \cup C$ s.t. $disj[u,v]$,

$$left[u,v] + right[u,v] + above[u,v] + below[u,v] \geq 1$$

4. Compute x-distance: $\forall u < v \in B \cup C$ we have:

$$(dx[u,v] \geq 0.0) \wedge (dx[u,v] \geq xs[v] - xf[u] + 1) \wedge$$
$$(dx[u,v] \geq xs[u] - xf[v] + 1)$$

and:

$$lt(dx[u,v], 0.0, xoverlap[u,v]) \wedge$$
$$lt(dx[u,v], (xs[v] - xf[u] + 1), left[u,v]) \wedge$$
$$lt(dx[u,v], (xs[u] - xf[v] + 1), right[u,v])$$

and similarly for the y-direction.

We used the C++ implementation of IBM ILOG Concert Technology [7] and MiniZinc to encode our MIP model. We solved them using the state-of-the-art CPlex MIP solver [2], and for the MiniZinc model we used the flatzinc compatible CPlex interface. Both encodings yielded similar results. We show the results of the first.

3.6 Experimental Evaluation

In order to study the performance of our model encodings for CP, MIP and SAT on different graph characteristics, we ran experiments on different types of input graphs. Experiments were run on a standard desktop machine with an Intel Core i7-4771 3.50GHz processor and 32GB RAM. Solvers were restricted to run on a single thread with a time-out of 300 seconds. The results were drawn using HTML5, JavaScript, D3.js [3] and Cola.js [13].

Our graph corpus consists of graphs from two sources; a set of randomly generated scale-free graphs, and a set of graphs derived from real-world instances. For the latter set, we use a selection of 100 graphs from the well-established Rome graph set [10], as already used in [188]. This sample contains 10 graphs from each group of graphs with sizes $|nodes| = 10, 20, \ldots, 100$ and covers the Rome set well [188]. Density ($|edges|/|nodes|$) ranges from tree-like (~ 1), to quite dense (1.61). We generated the flat scale-free graphs based on the model proposed by Bollobás *et al.* [104], with 10 graphs for each graph size from 7 to 100 nodes. In these generated graphs we controlled for edge density such that $|edges|/|nodes|$ is up to 1.22. Our decision to use scale-free graphs is motivated by the fact that scale-freeness is often observed in graphs stemming from important application areas like biology and the social sciences. To obtain a grouping for each of these graphs, an edge-compression heuristic [109] was applied. Thus, our full corpus consists of 940 flat-graphs, 100 Rome graphs, and the corresponding 1040 grouped graphs, called 'powergraphs' in the sequel. We then attempted to obtain a layout for each graph using our MIP, CP and SAT.

For our experiments, the components of the objective function were weighed in the following order. The stress component was given a weight of four, the compartment compactness was given a weight of two ($\alpha = 1/2$), and the entire layout compactness was given a weight of three ($\beta = 3/4$). These values yielded the most desirable results, and can be modified to fit the needs and expectations of the users.

Running the solvers on graphs of different sizes showed that the SAT solver was able to solve graphs of larger sizes, while CP and MIP solvers were slower. Figure 3.8 shows the median running time for all graphs up to the largest solved instance, with 57 nodes. The results of Figure 3.8 are presented in Table 3.1. While these results tend to favour SAT over MIP, we cannot say definitively under what conditions such layout models are better suited to one solver over the other. Our intuition here is that SAT is typically well suited to problems with small variable domains and highly disjunctive constraints, whereas MIP can handle large domains gracefully only when a linear relaxation gives a good approximation to disjunctions.

	powergraph			**flatgraph**		
	<3s	*<1m*	*<5m*	*<3s*	*<1m*	*<5m*
MIP	7	9	11	7	9	11
CP	15	16	25	13	16	25
SAT	25	38	57	18	24	36

Table 3.1: The highest node count of graphs solved by MIP, CP, and SAT; categorised into three time frames. It is clear that SAT performed best, followed by CP, with MIP having the worst performance for both 'powergraphs' and 'flatgraphs'.

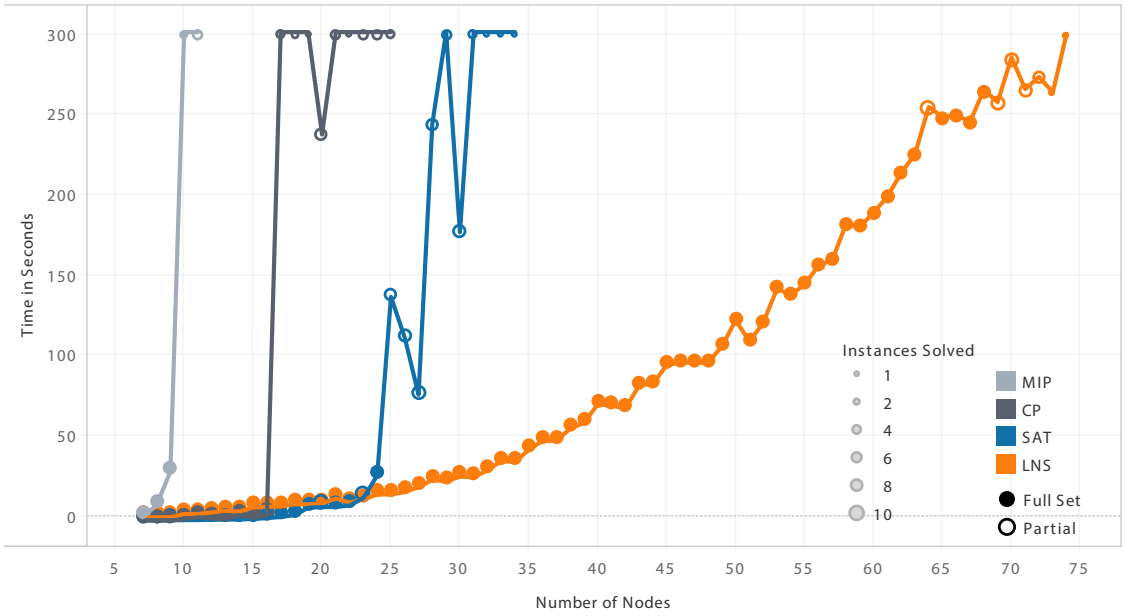
3.7 Large Neighbourhood Search Meta-Heuristic

The results presented in Section 3.6 indicate that SAT may be used to find optimal *Ultra-Compact Grid Layout* for small grouped graphs reliably, in around a second, which may be suitable for a web-service. It can be argued that interactive visualisations of small neighbourhoods are useful in exploring very large graphs, through filtering or semantic zooming that restricts the view to a sub-graph or aggregated overview. Still, being able to reliably visualise networks with hundreds of nodes gives a lot more flexibility.

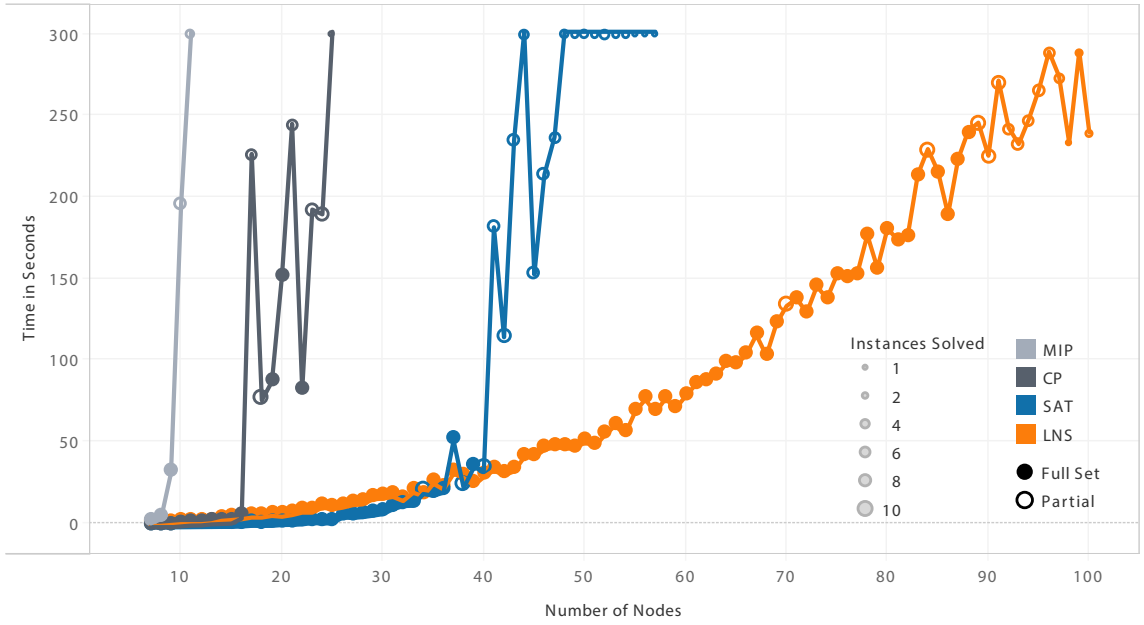
A common fall-back when solving hard combinatorial optimisation problems is to use generic meta-heuristic techniques like tabu search, simulated annealing or genetic programming. While not guaranteed to find an optimal solution, they are routinely used to find 'good' solutions to problems that are too hard to solve optimally.

We use *Large Neighbourhood Search* (LNS), which is currently the method of choice for solving various transportation and scheduling problems. The basic approach is to explore a large neighbourhood around the current solution and iteratively move to a high-quality neighbouring solution until no improvement is possible. One way in which the search can be done is to use a generic constrained optimisation technique to search the neighbourhood for the best solution. The advantage of this is that the search is then guaranteed to only find solutions that respect the problem constraints. This is why we decided to use LNS; with other meta-heuristic techniques, like simulated annealing, it would have been difficult to ensure that the containment and non-overlap constraints were satisfied during the search.

The LNS heuristic is intuitively simple: find an initial solution and then iteratively improve it by choosing a set of nodes for improvement. The space of their possible positions forms the neighbourhood for the search. Their new position is found by using MIP to solve the model given in Section 3.4, with some additional constraints fixing the relative position of the other nodes. The reason that we used MIP is that while MIP was



(a) *flatgraphs*



(b) *powergraphs*

Figure 3.8: Median solve times for ‘flatgraphs’ and ‘powergraphs’. Filled marks represent instances for which optimal results were found in under five minutes. Hollow marks indicate not all instances were solved in that time. Size of the marks indicate number of instances solved.

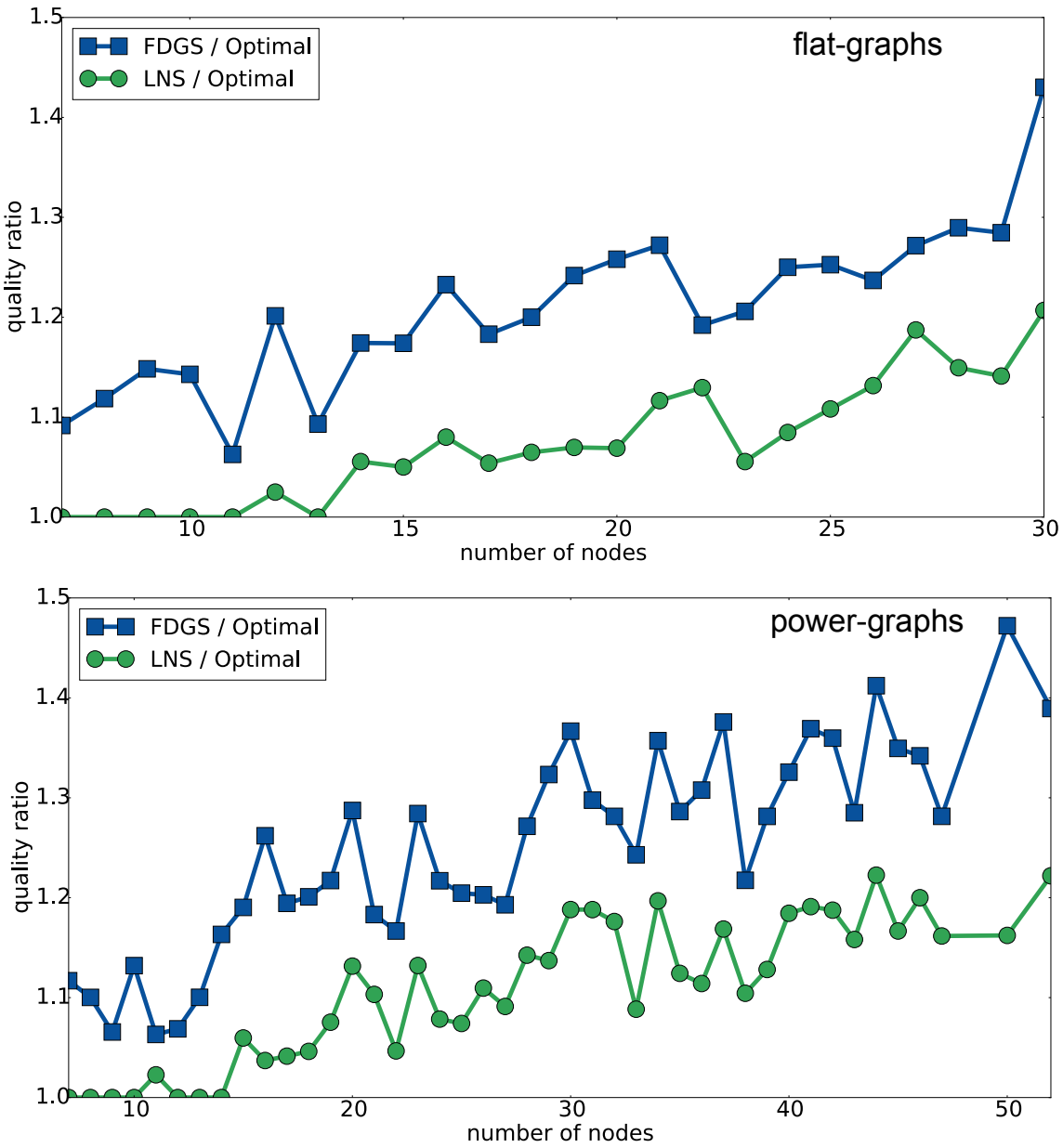


Figure 3.9: Average quality of objective obtained with the *Large Neighbourhood Search* (LNS) heuristic and the starting layout computed using *Force-Directed Grid-Snap* (FDGS), compared to the optimal objective for 'flatgraphs' (top) and 'powergraphs' (bottom). This is shown with respect to graph size.

slower than both CP and SAT for solving the original model, we found that it was the fastest method for solving this more constrained sub-problem. We believe this is for two reasons. The first is that the addition of relative position constraints removes most of the disjunctions from the model, so that the linear relaxation gives a good approximation to the underlying problem. The second is that MIP solvers such as *CPLEX* support *warm start* when solving similar problems. Algorithm 1 gives an overview of our heuristic.

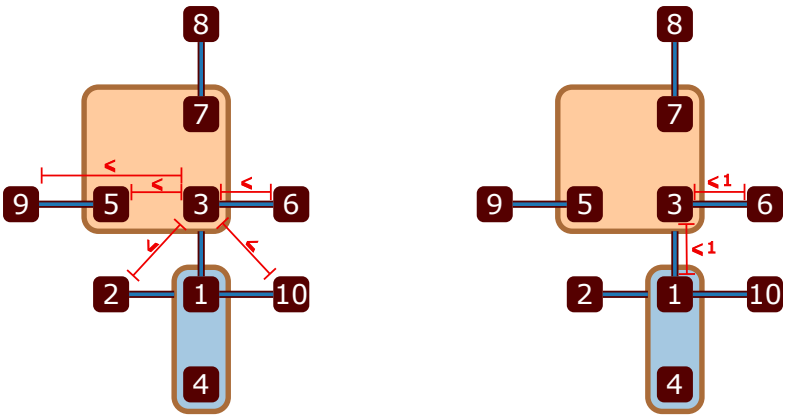
Algorithm 1 Grid Layout Heuristic

```
1: procedure GRID LAYOUT ( Graph g, Model m )
2:     l ← getForceDirectedLayoutWithGridSnap(g)
3:     d ← getDataWithTightConstraints(l)
4:     SolverLoad(m,d)
5:     b ← 10, c ← nil, t ← |B|/5 seconds
6:     for each c ← getNextNodeOrContainer(g,c) do
7:         relaxOrderingConstraints(c)
8:         δ ← δ ∪ getContainedNodes(c)
9:         lc1 ← getFreeLeavesDirectlyConnected(c)
10:        δ ← δ ∪ lc1
11:        δ ← δ ∪ getFreeLeavesDirectlyConnected(lc1)
12:        δ ← δ ∪ getContainedLeavesDirectlyConnected(c)
13:        relaxUpTo_b_nodes(δ,b)
14:        val ← runCPLEXsolver(timeout=t)
15:        setWarmStart(val)
16:        updateConstraints(val)
17:    end for
18:    for each leaves ← getUpTo(FreeLeaves(g), b) do
19:        relax(leaves)
20:        repeat 12-15
21:    end for
22:    return val
23: end procedure
```

The first step is to find an initial layout for the grouped network with a constraint-based force-directed approach. We implemented the 'grid-snap' technique described by Kieffer *et al.* [190] in the `Cola.js` [13] browser-based constraint-layout library. We extended this method to handle group-hierarchy containment inside rectangular regions using separation constraints, as described in [105]. The layout obtained by this heuristic *Force-Directed Grid Snap* (FDGS) approach for a 45-node graph is shown in Figure 3.12(a).

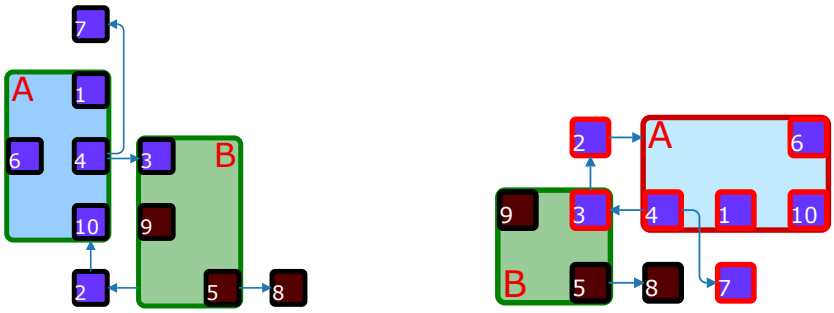
From this initial layout, two types of additional constraints are added to the model (Section 3.4) to massively reduce the search space. First, we generate inequality constraints over the x- and y-positions for pairs of nodes (Figure 3.10), locking their horizontal and vertical order. To allow the solver to move a neighbourhood of nodes, these constraints are selectively relaxed, as described below. Additional constraints on edge-length further restrict the search to only equal or shorter edges. This is done by bounding the Manhattan distance between adjacent nodes by their Manhattan separation from the FDGS layout.

In order to obtain improvements in reasonable time, we iteratively relax the ordering constraints for a subset δ of nodes and run the solver on this relaxed model. New ordering constraints for the nodes in δ and potentially tightened bounds on the edge-lengths are derived from the resulting layout and added to the model for the next iteration, where a solver warmstart is used to speed up the computation. The relaxation is divided into two main parts, represented by the `for` loops in Algorithm 1, lines 6 and 18. The first part processes neighbourhoods of nodes, the second is a post-processing to find the best placement for *free leaves*, that is, leaf nodes not contained in any other node.



(a) Horizontal ordering constraints (b) Edge-length constraints

Figure 3.10: Constraints types derived from the grid-snap layout to be added to the layout model. Ordering constraints for the pairwise relative positions between nodes in x and y dimension are added for all nodes pairs. (a) shows the horizontal ordering constraints for node 3, which restrict its x position to be less or equal to those of 6 and 10, and larger or equal than those of 9, 5, and 2. The constraints for the y position are obtained in the same way. (b) shows the edge-length constraints for node 3, in this case all distances to adjacent nodes have bound 1.



(a) Starting layout: container A is selected for neighbourhood detection. Nodes with a blue fill are selected for relaxation. This includes the nodes contained in A, nodes 2 and 7 as directly connected free leaves, and node 3 as directly connected contained node.

(b) Result: Nodes moved to a new position are highlighted with red outline. Node 3 was repositioned within container B, decreasing the container size, and container A was assigned a more horizontal shape, allowing the drawing to fit into a much smaller grid, moving all relaxed nodes. The length of the edge between nodes 4 and 7 was decreased from 4 to 3. Grid size has been reduced from 5×5 to 5×3.

Figure 3.11: A step in the *Large Neighbourhood Search* heuristic.

In each iteration of part one, the selection of nodes for relaxation is as follows: first, we select a node c, whose neighbourhood will be relaxed, to be included in δ (Function getNextNodeOrContainer in Algorithm 1). In case there are contained nodes, we first pick c from the list of containers ordered by size, that is, the largest container in the first iteration, followed by the second largest, and so on. Otherwise, we consider each node as an empty container, and in each iteration select one of them in random order. We now add further nodes to δ up to a small bound b on the size of δ. In our implementation, $b = 10$ gave a good trade-off between running time and quality improvement. First, we randomly select up to b nodes in c. If c has fewer nodes than b, we add further nodes in the following order until $|\delta| = b$: The group $lc1$ of free leaves that are adjacent to c. The group of free leaves that are adjacent to nodes in $lc1$. Contained leaf nodes that are adjacent to c.

After relaxing the ordering constraints for the nodes in δ, we run the solver for a limited time t. Figure 3.11 illustrates such a step of the heuristic. We choose $t = |B|/5$ such that the total run-time for the algorithm is always bounded proportionally to the size of the graph. The result is used to initialise the next iteration and the solver warmstart.

We iterate until each container (or each node, in case there are no containers) has been selected once (for-loop at Line 6 of Algorithm 1). Afterwards, we relax sets of free leaves independent of a container, and rerun the solver, until all free leaves have been relaxed once (for-loop at Line 18 of Algorithm 1). The final layout is shown in Figure 3.12(b).

We evaluated the LNS heuristic on our graph corpus. The reduction in search space leads to significantly faster solves as can be seen in Figure 3.8. Figure 3.9 compares the layout quality of LNS and FDGS with the optimal as obtained by SAT. Across all graphs in our corpus, the mean quality ratio for FDGS was ~ 1.24 while for LNS it was ~ 1.1, i.e. LNS was typically twice as close to the optimal as FDGS. Visually, FDGS gives a much less compact layout with a significantly larger total edge-length, compared to layout refined by LNS as evidenced by the side-by-side comparisons in Figures 3.12 and 3.16.[3]

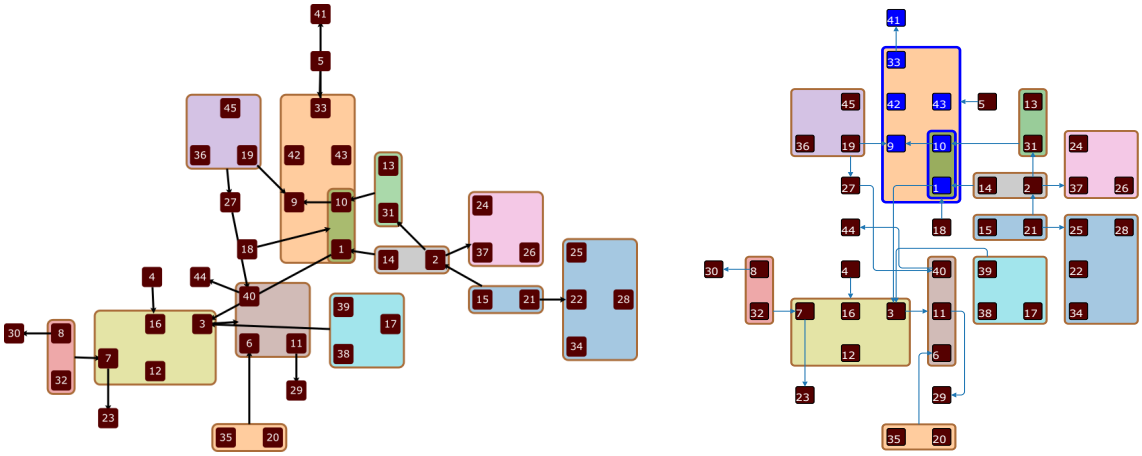
3.8 Additional Experiments

In this section we discuss some work that was done in attempts to achieve better running times. However, as the improvements yielded only minor gains, we did not explore these thoroughly.

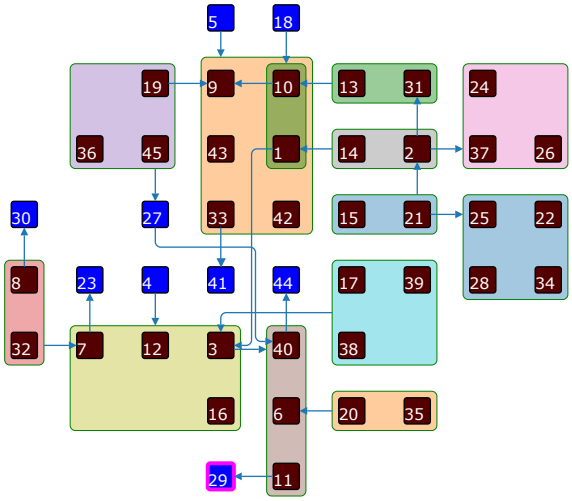
In Mathematics, objects of a set can be split into equivalence classes based on a defined notion of equivalence. In the case of our layout problem, we can divide nodes into equivalence classes based on their connections. A pair of nodes in a graph are equivalent if they have exactly the same connections.

While using the 'powergraph' edge compression technique [104], we realised that the compressed graph ended up with several nodes that did not have any edges. Figure 3.13 shows a simplified example, where all the nodes in the compressed network end up with no edges. The positions of nodes 1 and 2 could be interchanged without affecting the quality of the layout. Similarly, it does not matter in which order nodes 3 and 4 are placed. Getting rid of these nodes from the network layout problem would result in fewer

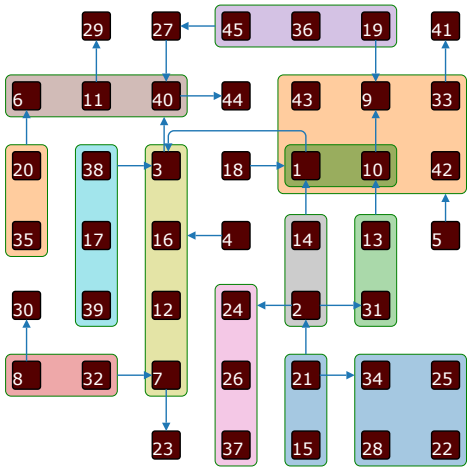
[3] More examples highlighting the difference in quality between FDGS layout and that obtained by FDGS with LNS refinement are available on-line [6].



(a) Left: Force-directed layout with containment. Right: Force-directed layout with grid-snap. Solve time: 2.2 seconds. Grid Size $= 10 \times 11$. Total Edge Length $= 33$. Number of Edge Crossings $= 4$. Objective $= 35.89\%$ higher than optimum. The first neighbourhood considered by the LNS search is highlighted in blue.

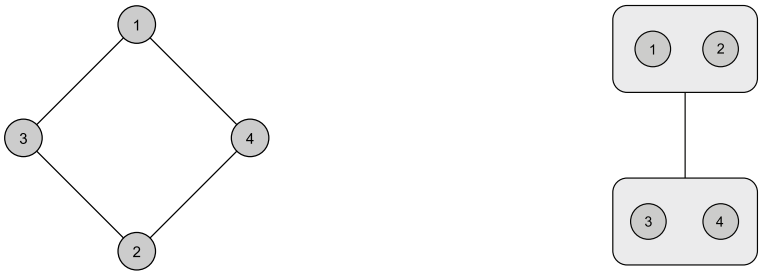


(b) LNS outcome. Highlighted nodes are the last neighbourhood with relaxed constraints, the node with pink outline was the only one moved by the solver. Solve Time $= 43$ seconds. Grid Size $= 9 \times 8$. Total Edge Length $= 29$. Number of Edge Crossings $= 4$. Objective $= 17.94\%$ higher than optimum.



(c) Optimum. Only one edge has more than unit length. Grid Size $= 7 \times 7$ and is optimal. Total Edge Length $= 24$. Number of Edge Crossings $= 0$. Solved by SAT in 99.2 seconds.

Figure 3.12: A network with 45 nodes arranged in four ways: the force-directed approach that forms the basis of our LNS approach; the grid-snap approach that allows us to derive the constraints for the LNS approach; the final outcome of the LNS approach; and the final optimal outcome of the SAT.



(a) A small network with four nodes. Node 1 has connections to nodes 3 and 4, but so does node 2.

(b) The same network as in the figure on the left, but the edges have been compressed using the *powergraph* [104] technique.

Figure 3.13: An example showing a network with two equivalence classes, nodes 1 and 2, belong to the same equivalence class, while nodes 3 and 4 belong to another equivalence class.

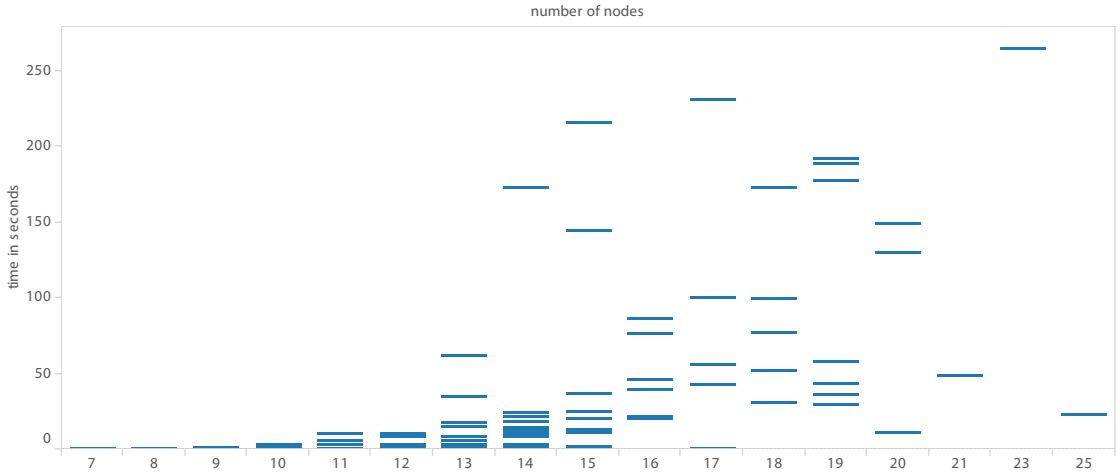
variables for which the solvers need to find optimal values and thus reduce the search space.

We modified our model, such that equivalent nodes would be removed from the pool of variables. Instead, they would be allocated a reserved space in the group to which they belong. Thus, after the solver finds an optimal layout, they are reinserted into the reserved spaces. We evaluated the running time of the *CPLEX* [2] solver for finding optimal layouts for networks with different sizes (from the corpus of randomly generated scale-free graphs discussed in Section 3.6) using this reduction technique. Without the equivalence class reduction, *CPLEX* was able to find optimal layouts for networks with 7, 9, and 11 nodes, in under 3, 60, and 300 seconds respectively (as shown in Table 3.1). With the equivalence class reduction, *CPLEX* was able to find optimal layouts for networks with 12, 17, and 25 nodes in under 3, 60, and 300 seconds respectively, as seen in Figure 3.14. Using this technique significantly reduced the solving time for instances where a large number of equivalent nodes existed, however, it obviously did not enhance those with few or no equivalent nodes, as exhibited by some outliers in Figure 3.14(a).

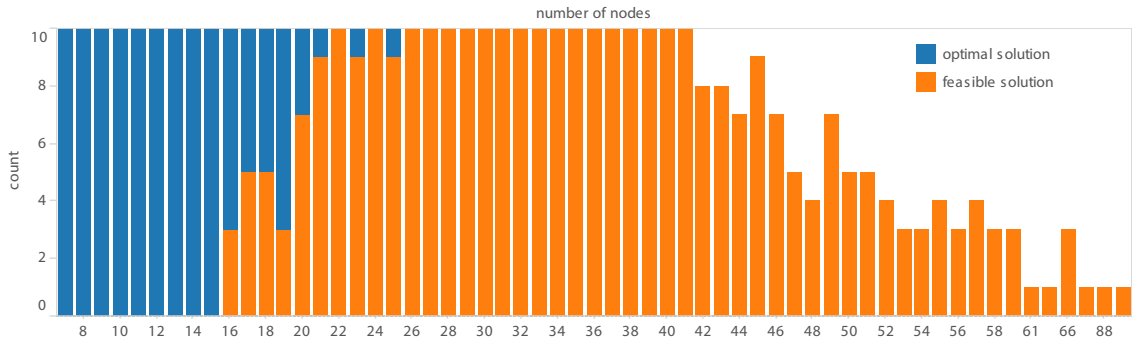
We also conducted an experimental evaluation to understand the effects of the size of the neighbourhood in the LNS approach on the running time. We varied the number of nodes whose constraints are relaxed during the iterative steps of the LNS approach, and ran the *CPLEX* solver on networks with different sizes (from the corpus of ROME graphs mentioned in Section 3.6). The results are shown in Figure 3.15. The running time of the solver increases exponentially with the increase in the size of relaxed neighbourhoods.

3.9 Conclusion

We have introduced a new ultra-compact, grid-like layout aesthetic for node-link diagrams with arbitrary containment that is motivated by the grid arrangements that are used almost universally by designers in typographical layout. We have explored whether generic constrained optimisation techniques (MIP, CP and SAT) are now fast enough to be used for high-quality drawings of this kind. We found that SAT was the most effective, and quite practical for producing high-quality layouts for graphs of up to 20 nodes in under a second — useful, for example, in interactive contexts where it is possible to obtain an aggregated



(a)



(b)

Figure 3.14: The results of the experimental evaluation of the model equipped with filtering out nodes with similar equivalence classes. Figure 3.14(a) shows the running time of *CPLEX* to find an optimal solution, while Figure 3.14(b) shows the number of feasible versus optimal solutions achieved out of the ten instances for each graph order.

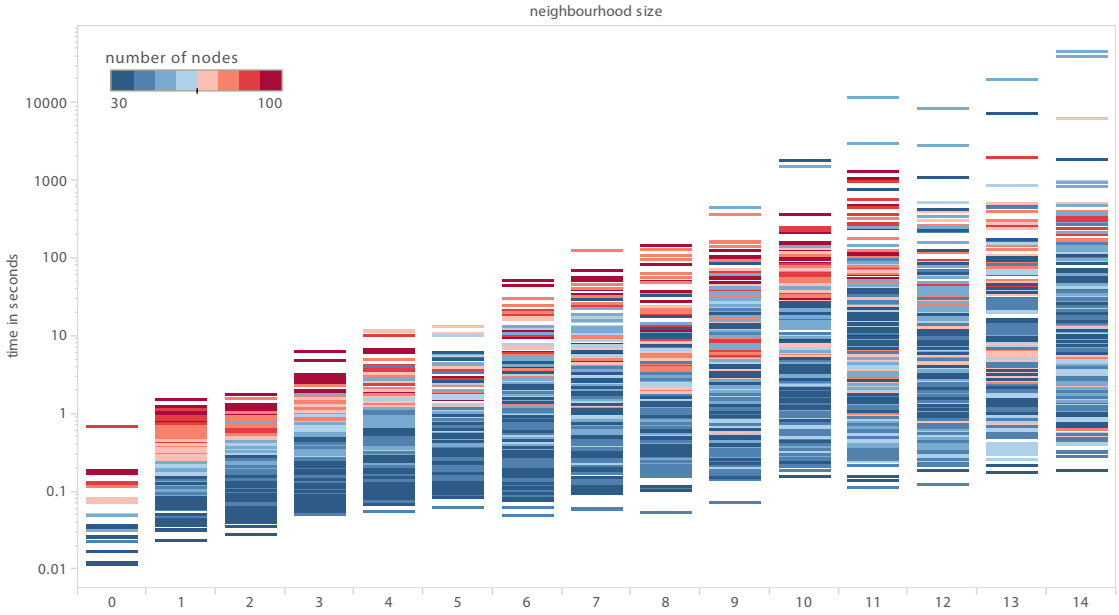
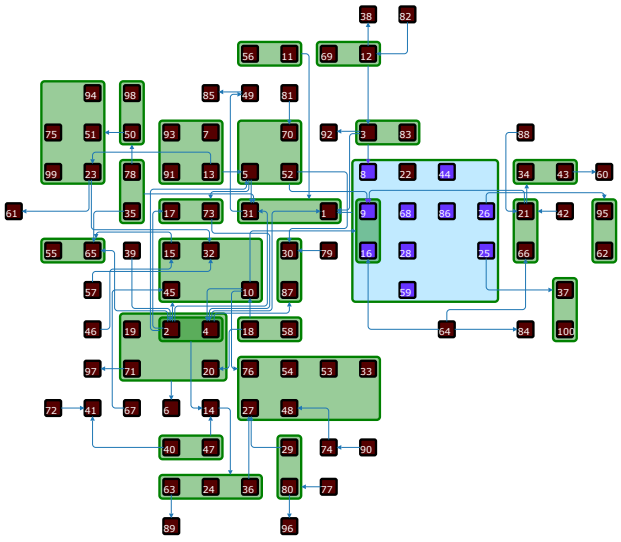
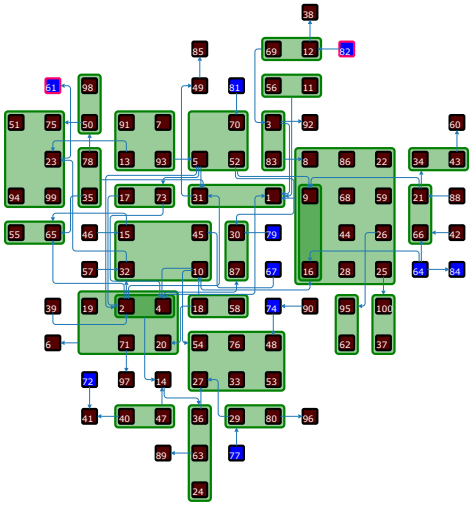


Figure 3.15: The results of the experimental evaluation on the running time of the *CPLEX* solver when the neighbourhood size is varied in the LNS approach



(a) Force-directed layout with grid-snap. Grid Size = 16 × 14. Total Edge Length = 138. Number of Crossings = 52. Solve Time = 14.4 seconds. The first neighbourhood considered by the LNS search is highlighted.



(b) LNS. Highlighted nodes are the last neighbourhood with relaxed constraints, the node with pink outline was the only one moved by the solver. Grid Size = 13 × 14. Total Edge Length = 135. Number of Crossings = 50. Solve Time = 348 seconds.

Figure 3.16: Two diagrams of a network with 100 nodes, created using a force-directed layout and our LNS approach.

or partial view of a larger network; and graphs of up to 50 nodes in a few minutes may be useful for producing canonical off-line views.

Although in this chapter, we discuss solving a set of models for network layout to optimality, it is open for debate as to whether our particular model represents the 'best possible visualisation', and we do not claim this. Rather, this is precisely one of our core motivations: by rapidly modelling different types of layout through declarative techniques, we are able to rapidly experiment with different approaches to the layout problem. A number of the ideas for layout presented here (e.g. re-orientable nodes, ultra-compactness, multi-directional flow layout; as well as arbitrary group containments), are to our knowledge very novel and very difficult to experiment with, than by any other means.

Another promising use of the optimal techniques is in finding a baseline against which approximate methods may be compared to assess quality. This was demonstrated in our evaluation of the LNS meta-heuristic approach. Our evaluation showed that the LNS method could produce compact layouts for graphs with up to 100 nodes in a few minutes that are usually within 20% of the objective function's optimum.

We also explored enhancements to improve the scalability of our approach by reducing the number of variables. Such modifications only slightly improved the solving time and it is known that detailed representations of large networks become difficult to read, regardless of layout quality, thus,

Chapter 4

Cognitive Load - A Survey

"nanos gigantum humeris insidentes"

Bernard of Chartres,
which translates into: "dwarves seated
on the shoulders of giants"

For decades, researchers in information visualisation and graph drawing have focused on developing techniques for the layout and display of very large and complex networks. Experiments involving human participants have also explored the readability of different styles of layout and representations for such networks. In both bodies of literature, networks are frequently referred to as being 'large' or 'complex', yet these terms are relative. From a human-centred, experiment point-of-view, what constitutes 'large' (for example) depends on several factors, such as data complexity, visual complexity, and the technology used. In this chapter, we survey the literature on human-centred experiments to understand how, in practice, different features and characteristics of node-link diagrams affect visual complexity.

This survey was conducted in collaboration with my supervisors Tim Dwyer and Karsten Klein, Daniel Archambault, Stephan Diehl, Helen C. Purchase, and Hsiamg-Yun Wu. We have submitted a review paper to the Information Visualization Journal.

4.1 Introduction

There has been much work done on designing algorithms that can efficiently scale to create pictures of very large graphs (see Section 2.2. However, what remains a more open question is whether pictures of very large and complex networks require a mental effort that exceeds the capabilities of an average human brain.

Eick and Karr [116] define the term 'visual scalability' as the capability of visualisation tools to effectively display large data sets. They also discuss factors affecting visual scalability, like human perception, monitor resolution, visual metaphors, interactivity, data structures and algorithms, as well as the computational infrastructure. A more recent discussion of these, and similar factors, are presented by Jankun-Kelly *et al.* [178]. They also distinguish perceptual and cognitive scalability: "though elements may be perceivable, they may still exhaust cognitive resources".

In cognitive psychology, Miller's "seven plus or minus two" [228] is commonly accepted as a rule-of-thumb for the limitation on peoples' working memory. Working memory is an example of one of the 'cognitive ceilings' that might affect peoples' ability to reason about large networks. Do working memory and other cognitive limitations have implications for the size and complexity of graphs that we should be trying to visualise?

Huang *et al.* [169] propose a framework for cognitive load in the context of graph visualisation. Based on results from cognitive psychology, they sketch a model that relates cognitive load during graph analysis tasks to mental effort and task performance in terms of response time and accuracy. They discuss a number of other factors affecting cognitive load: the domain (e.g. a highly technical and specific domain requires the user to relate their knowledge to the visual); the data itself (e.g. structure of the graph); the task (does it require deep understanding of the graph structure?); visual representation (does it follow best practice layout and design principles); demographic (e.g. the experience of the users); and time pressure. They report on one study of cognitive load that confirms some of these effects, but their model which suggests step changes in performance due to load, while compelling, is not fully validated.

In a more general evaluation of effect of display type on visualisation cognition, Yost and North [337] demonstrated that more pixels make it possible to show more data without significant loss of performance. Going from 2 to 32 mega pixels led to a 20-fold increase in displayed data: the task completion times tripled, while accuracy only decreased from 95% to 92%. However, the data, visualisations and tasks considered (such as search and comparison) are relatively simple compared to networks and associated tasks involving understanding of connectivity. It cannot be assumed that such results carry over, or that display size is the only—or even most—significant limitation. The goal of the survey presented in this chapter is to better understand these cognitive limitations. To differentiate this human aspect of scalability of node-link diagrams from technological or technique specific limitations, we use the term 'cognitive scalability'.

This topic is important for our field, as such insights can guide the design of future techniques. For example, we are attempting to find tacit knowledge in past studies concerning the number of nodes and edges that are too difficult to work with in a single view. If we can establish such numbers, then it might suggest that we need to direct efforts away from algorithms and rendering techniques that can scale to huge numbers of network elements. Instead, for such large networks, we could focus on interactive ways to explore neighbourhoods (e.g. [107, 304]) or abstractions (e.g. [16, 30, 103]) instead of attempting to display the full set of nodes and links.

Thus, we survey a large number of papers reporting empirical studies of node-link diagrams, being exhaustive within the corpora of core visualisation proceedings and journals. We aim to establish a consensus for definitions of adjectives like 'large' or 'dense' for node-link diagrams that are too complex to be easily comprehensible or useful for standard graph analysis tasks. We also provide an overview of the types of networks and tasks, as well as experimental design of these experiments.

In general, we find that the limits of scalability of the node-link network visualisation paradigm are rarely addressed directly. Rather, there seem to be tacit assumptions (or possibly unreported pilot findings) about what size node-link diagrams are usable for different tasks, and experiments stay within these bounds while testing specific techniques.

Our key findings are that only a small range of graph sizes and structures have been used in experimental evaluations of graph visualisation techniques, mostly limited to small and sparse graphs. In particular, three quarters of studies use graphs with 100

nodes and 200 edges or less, and the remaining studies test interactive techniques, such that only a small portion of the graph is shown on the screen at a time. These findings are discussed further throughout the chapter and listed in full in the conclusion.

This chapter is structured as follows: Section 4.2 discusses related surveys, primarily in graph visualisation; Section 4.3 outlines our scope, methodology, and describes our categorisation framework; then we present the results of our survey in Sections 4.4 , 4.5 and 4.6, followed by a discussion on trends in the network visualisation community in Section 4.7.

4.2 Related Surveys

In the fields of graph drawing and visualisation, a number of surveys have considered scalability from different perspectives. In particular, there has been much discussion of the scalability of algorithms and computer hardware to compute node-link diagram layout. Such papers tacitly acknowledge that very "large" and "dense" graphs are difficult to read and hence propose interaction techniques to navigate aggregated graphs. They may even report on studies of the readability of networks using different layout, interaction or rendering techniques. Yet, rarely do they explicitly address the question of what is the largest (most complex) such diagram that people can usefully comprehend.

Many past surveys characterise the techniques available for graph visualisation. The surveys of Herman *et al.* [157] and von Landesberger *et al.* [311] are both of this type, focusing on techniques for graph visualisation and their strengths and weaknesses. Elmqvist and Fekete [119] characterise techniques in information visualisation that use hierarchical representations as a form of data abstraction. Recent surveys have also focused on specific areas of network visualisation including multi-faceted graph visualisation [153], group structures in graphs [308, 309], matrix reordering techniques [60], edge bundling [214], and networks in social media [78].

These surveys organise graph visualisation methods at the technique level, or specialise in a particular technique and present a survey of research in the area in-depth. These surveys do not focus on questions about how scalable these representations are from a human-centred perspective.

In the area of dynamic graphs, surveys have been conducted on dynamic networks [59] and dynamic data in information visualisation in general [41]. There have also been reviews focused on the human-centred effectiveness of animation, small multiples, and drawing stability (mental map preservation) by summarising experimental results and providing guidelines for visualisation designers. One such work by Archambault and Purchase [33] summarises empirical results that relate to mental map preservation in dynamic graph drawing. In a later work [34], based on the results of new studies, they review the conditions where animation and small multiples are effective and present new results for diagrams of low drawing stability. These papers focus on dynamic network visualisation and do not consider network visualisation in general. While providing a survey of human-centred effectiveness of visualisations to some degree, they do not consider cognitive scalability of the representations directly.

In this chapter, we review evaluations of node-link visualisations of static and dynamic graphs. What is unique to our survey is that it examines cognitive scalability of node-link visualisations of graphs through the lens of human-centred experiments, to gain bounds on the sizes of graphs that have been displayed to the human while still usefully supporting analysis tasks. We seek to answer this question by surveying the literature

of controlled experiments involving human participants to test node-link diagram representations of networks. Our summary information about the networks, techniques, and tasks considered in these studies also presents an up-to-date snapshot of the evolution and state-of-the-art, controlled network visualisation evaluation.

4.3 Methodology

In this section, we clarify the scope of this survey, including the venues that we examined, and describe our categorisation framework. We have tried to be as systematic as possible in covering complete conference and journal venues with clearly defined constraints, as detailed below.

4.3.1 Scope of Survey

To the best of our ability, we have sought to include in this survey all papers with a human-centred experiment (formal user study) where at least one of the conditions is a node-link representation. Both static and dynamic graph drawing studies were considered. For much of our analysis we focus on individual studies, where papers could contain multiple studies. Throughout the survey, 'paper' refers to the publication that presents the study, while 'study' refers to an individual experiment.

Our time range begins with the earliest formal human studies in network visualisation of which we are aware [260] and ends on the 31^{st} of March 2018. We consider the major conferences and journals listed in Table 4.1 and, examine all publications from this date, or the founding of, the conference/journal. This date is based on the above limit and accessibility of the venue. Well-known articles outside these venues are also included and listed under 'Other'.

For most venues, we were able to read all titles and further examine the abstracts of papers that were relevant to our survey. The only exception was *ACM CHI*, where there were far too many papers: *CHI* accepts up to 600 papers in total each year. Instead, we found relevant papers at *CHI* using the *HCI Bibliography*.[1] A query of this database, limited to the CHI conference and using search terms '`(network | graph) visuali* study`', returned 25 results, of which close inspection revealed 12 to be relevant, as detailed in Table 4.1[2].

4.3.2 Categorisation Framework

Information about each paper was collected and coded according to a number of criteria. Section 4.4 reports our findings on the sizes of graphs used in experiments. We identified the number of nodes and edges used within each study and computed the density of the graphs. If the study was on dynamic graphs, we counted the number of timeslices. We noted if this information was explicitly stated or whether it needed to be derived or inferred (only nodes, only edges, or both). Exact numbers were sometimes unavailable, but we estimated the sizes based on figures of the stimuli (e.g. [172,175]). If authors provided

[1] http://hcibib.org/bs.cgi

[2] We also provide our curated bibliography as an online Tableau story
`https://public.tableau.com/profile/vahan\#!/vizhome/cognitiveScalability/CognitiveScalability`

Table 4.1: Venues considered in this survey.

Venue	First Paper	# of Studies	# of Papers	References
ACM CHI	2006	16	13	[22, 24, 77, 103, 123, 161, 210, 230, 234, 282, 289, 341, 343]
Diagrams	2006	5	5	[80, 244, 264, 319, 336]
EuroVis & CGF	2009	16	13	[37, 47, 90, 101, 138, 139, 245, 246, 274, 275, 287, 299, 346]
GD	1995	25	23	[32, 36, 49, 53, 67, 70, 73, 127, 143, 147, 173, 194, 198, 221, 227, 247, 256, 259–262, 347]
IVJ	2002	19	14	[35, 92, 126, 141, 163, 169, 187, 203, 217, 243, 266, 310, 318, 320]
PacificVis	2008	7	6	[31, 71, 160, 168, 171, 174]
InfoVis & TVCG	2003	61	47	[23, 38, 44, 45, 48, 68, 72, 86, 88, 106, 110, 140, 142, 149, 156, 159, 166, 176, 181, 183, 191, 206, 207, 211, 222, 237, 252, 263, 271, 276, 286, 290, 298, 300, 302, 305, 314, 324–326, 328, 330, 332, 333, 335, 338, 342, 345]
Other		3	3	[167, 172, 175]
Total		**152**	**124**	

the algorithm and parameters used to generate the graphs, we computed the size based on this information (e.g. [160, 161, 318, 330]).

Section 4.5 discusses other factors we found relating to scalability and, as such, is divided into three parts: HCI factors, graph drawing factors, and study design. The first part presents our findings on factors relative to human computer interaction and scalability. We coded and described the types of tasks [212] and interaction [334] used in each study. Application areas—if any—and the challenges they pose are also collected and are discussed in this section. The second part presents graph drawing factors related to scalability. We gathered information about the types of graphs, whether they were static or dynamic, and if they had attributes. We recorded information about the graph structure and noted if the data was real or generated. We also coded the layout algorithm used in the experiment. The third part discusses factors with respect to study design, including the number and nature of participants and whether the studies were within or between subject. In addition, we discuss the results of the study, and present the studies that are interesting in this data set.

Section 4.6 presents information about how the authors of experiments decide how large a graph should be presented to a participant. In particular, it discusses pilot studies and justification for using graphs of a certain size. In Section 4.7, we discuss the evolution of studies and their design over time in our community. Information about how studies have evolved and their venues is presented here also. The final sections of this chapter include a discussion and a conclusion.

4.4 Basic Measures of Complexity

In this section, we consider measures of size that can be used to describe the graphs used in studies.

There are numerous measures that could be used when considering complexity of graph visualisations. However, we have found that in reports on graph study design and methodology, typically, very few are considered. The number of nodes is the most commonly reported measure.The number of edges and/or density, on the other hand, is less frequently provided - although it is known to be a significant factor [141]. More edges, inevitably leads to more edge crossings, which is known to affect readability [256]. When the data changes over time (i.e. dynamic graphs) additional measures, such as number of timeslices, become similarly important to gauge complexity.

Table 4.2: Summary of graph sizes used in usability studies of graph visualisation. We used $\frac{|E|}{|V|(|V|-1)} \times 100$ to calculate density, and $\frac{|E|}{|V|}$ to calculate linear density.

Measure	Minimum	Maximum	Average	Median	Lower Quartile	Upper Quartile
nodes	2	113 M.	457,242	49.5	18.5	116.5
edges	1	1.8 B.	7.87 M.	73	23	242.5
density	0.014	115.5	10	5	1.7	10
linear density	0.38	102.8	3.5	1.5	1	2.4
timeslices	2	15	6	6	3	7

Since our interest is primarily in findings regarding the scalability of network visualisation, we report on the distribution of graph sizes considered within studies. Table 4.2 summarises the minimum, maximum, average, median, and upper/lower quartiles for these metrics. While there is a large range across all these metrics, in each case the median is closer to the minimum. This skewed distribution implies that the majority of studies use small graphs.

In order to have a better understanding of the range of sizes of graphs used across and within user studies, we plotted for each study the number of nodes against the number of edges for both the smallest graph used in each study and the largest (Figs. 4.1, 4.2). Each minimum and maximum pair is connected by a link showing the range of graph sizes evaluated in that study. The circles are: hollow ◎ if the number of nodes and edges was not mentioned by the authors; double enclosed ◉ if only the number of nodes was mentioned; full ● if both number of nodes and edges were mentioned, or both metrics were otherwise apparent (e.g. if number of nodes n was given and the graph type was a tree, we assumed $n-1$ edges were present). In cases where the information was not mentioned we estimated the graph sizes from the figures (e.g. [70, 72, 142, 191, 246, 282, 299, 300, 305, 341, 345, 346]). We also used the hue of the circles to differentiate between static ■ and dynamic ■ graphs.

Where precise sizes were not provided, we did our best to infer approximate sizes. For example, in some cases we were able to estimate the graph sizes from the figures ([70, 72, 142, 191, 246, 282, 299, 300, 305, 341, 345, 346]).

The following subsections discuss each of the four measures of Table 4.2 in more detail.

4.4.1 Number of Nodes

The number of nodes in a graph is an important, but incomplete indicator, of complexity. It is clear that, for most tasks, difficulty is affected by introducing more nodes to a connected graph. We would assume that most experimenters pilot, or at least consider, graphs with different numbers of nodes to avoid tasks that are too trivial or impossible. Yet out of the 152 studies covered in this survey, 28 studies do not mention node count at all.

Among the studies that are included in our survey, 15 studies use graphs with more than 1,000 nodes [37, 68, 103, 207, 217, 221, 234, 302, 318, 325, 328, 335, 347] and another nine that use graphs with more than 500 nodes [181, 210, 230, 245, 263, 290, 335, 338].

Among these studies, the majority aim to evaluate tools that use interactive exploration to extract parts (e.g. neighbourhoods) of the graph (20 / 24 studies, 83%). Some of these studies evaluate aggregation techniques, thus they would require graphs with a

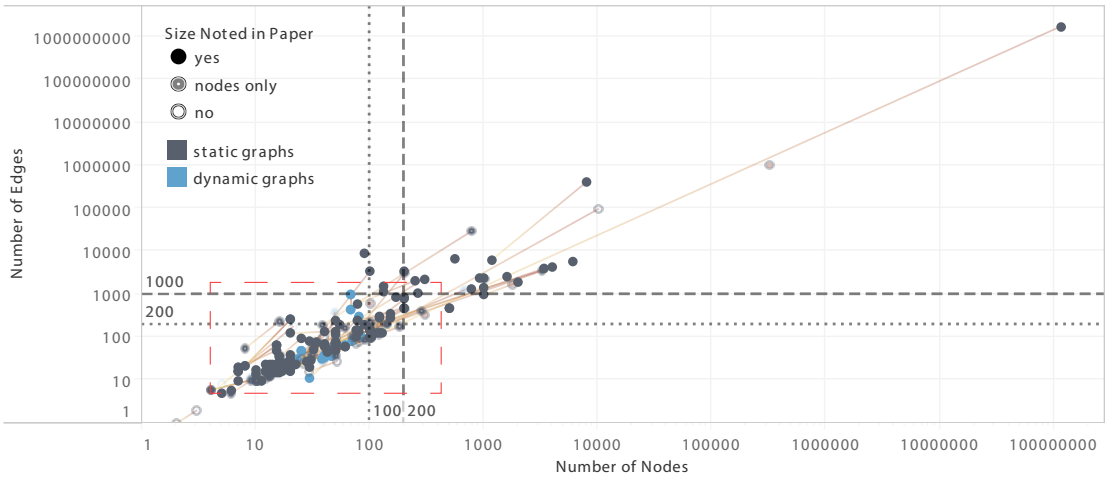


Figure 4.1: The size of graphs used in user studies. The x-axis shows the number of nodes, while the y-axis shows the number of edges. Both axes have a log scale. The grey circles represent static graphs, while the blue show dynamic graphs.

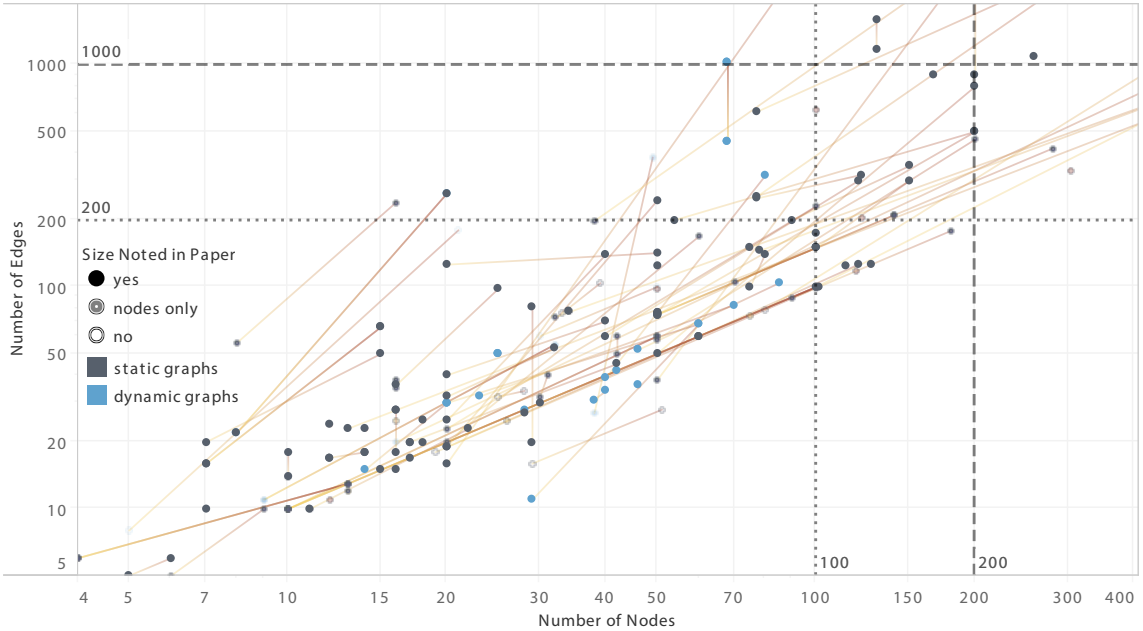


Figure 4.2: The section, marked by red dotted lines in Fig. 4.1, magnified for better readability.

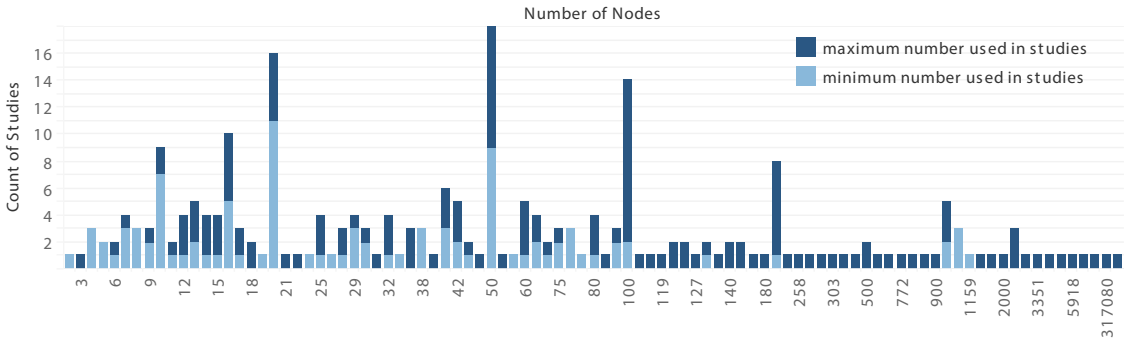


Figure 4.3: Histogram of the minimum (light blue) and maximum (dark blue) number of nodes of graphs used in studies.

large number of nodes, in order to highlight the benefits of compressing several nodes into fewer representations.

It is problematic to infer cognitive scalability of graph visualisation in the presence of interactivity because most of these studies do not ask the participants to perform the tasks on the whole graph. Rather, only a part of the graph is visible. When the authors do not report the precise number of nodes actually visible to the user, there is little that can be inferred about cognitive scalability. Eight of these 24 studies also use smaller networks with fewer than 100 nodes in their evaluations. This is shown by the long lines (Fig. 4.1, 4.2) that connect different graph sizes used by the same studies.

In order to understand the selection of number of nodes, we plotted a histogram of the number of nodes. Fig. 4.3 shows a number of spikes around specific numbers of nodes: 20, 50, and 100. The biggest spikes are at graphs with 50 nodes, which were used by 18 studies, followed by 20 and 100, used in 16 and 14 studies respectively. We believe that spikes at these round numbers suggest experimenters choose the number of nodes arbitrarily. These round numbers were not identified by empirical research and a formal study on ceiling and floor effects for graph cognitive scalability might lead to a better selection of the number of nodes.

Nine additional studies use graphs with more than 200 nodes. Similar to the above, some of these studies only show subparts of the network [90, 206, 247, 319], while some evaluate tools that scale well with large networks [71, 72, 90, 143, 147, 160, 206, 319]. Four out of these nine studies have also used networks with less than 100 nodes.

To conclude, only 56 out of 152 studies use graphs with more than 100 nodes. Most of the studies that use a large number of nodes, use abstractions or aggregation to show only parts of the graph at a time, but do not report on the number of nodes seen at a given abstraction. In general, it is not clear why most researchers choose to use graphs with 100 nodes or less (121 / 152 studies, 80%). Some authors mention pilot studies where they discovered ceiling or floor effects, which could be seen as indications of cognitive scalability. We discuss these in Section 4.6.1; however to our knowledge, controlled studies were not conducted to verify or explain these effects.

4.4.2 Number of Edges

The number of edges in a graph is also an important indicator of complexity, especially in node-link diagrams, where the edges are drawn as lines. A large number of edges necessitates an increased number of crossings and overlap (collinear and therefore ambiguous

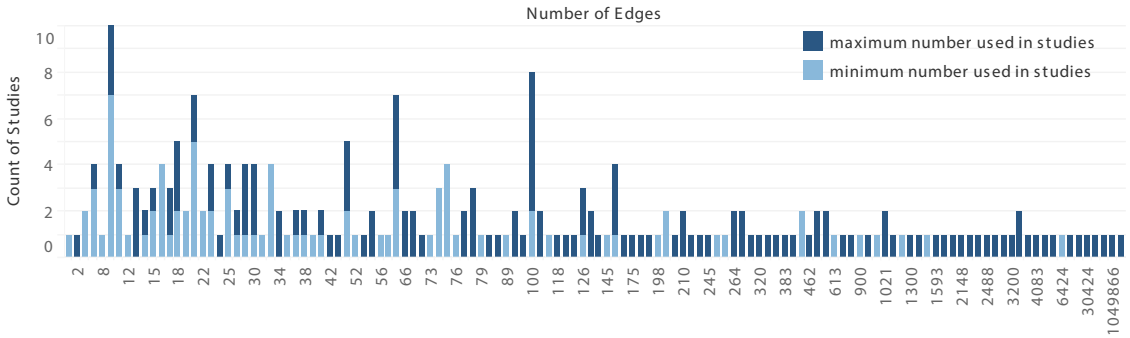


Figure 4.4: Histogram of the minimum (light blue) and maximum (dark blue) number of edges of graphs used in studies.

lines). An excessive number of links often lead to what are known as 'hairball' visualisations. Among the 152 studies covered in this survey, 124 explicitly mention the number of nodes, while only 87 specify the number of edges.

There are only 28 studies that use graphs with 1,000 edges or more. Most of these allow the participants to look at parts of the network instead of performing the task on the whole network [37, 103, 143, 156, 206, 211, 217, 230, 234, 245, 247, 252, 286, 290, 302, 310, 318, 325, 338, 347]. Similar to studies that use a large number of nodes to highlight the benefits of aggregation or interaction methods, some studies use a large number of edges to show the benefits of edge compression, bundling, or highlighting techniques. Many evaluate tools or techniques where they, by design, scale well to handle graphs with a large number of edges. For example, a study by Giacomo *et al.* [143] evaluates a technique that highlights edges in order to enhance the readability of graphs that have many edge crossings. Another category of studies that use a large number of edges, evaluate visualisations (e.g. adjacency matrices) that scale well with a large number of edges in comparison to node-link diagrams [48, 141, 156, 207, 247, 332].

Again, just as we saw spikes in frequency at round numbers of nodes used for study graphs (Fig. 4.3), in Fig. 4.4 we see spikes particularly at 10, 20, 60, and 100 edges. However, the spikes are less pronounced: only ten, seven, seven, and eight studies, respectively. The smaller spikes for number of edges compared to number of nodes, tends to suggest that experimenters choose a specific number of nodes first, then adjust the density, presumably to control the difficulty level for their specific tasks. We found that 97 studies use graphs with less than 100 edges, while 75 studies use graphs with 100 edges or more.

In summary, we were surprised that about half of the studies do not report the number of edges. We would argue that without this information, graph evaluations are difficult to reproduce. The majority of studies use graphs with less than 1,000 edges (126 / 152 studies, 83%). The 28 studies that use graphs with 1,000 edges or more, either evaluate tools that require dense community structures, aim to show that node-link diagrams fail to perform well on graphs with a large number of edges, or highlight the benefits of aggregation and interaction techniques.

4.4.3 Density

The previous section mentions studies that use graphs with a large number of edges. Nonetheless, most of these graphs have very low densities (relatively few edges to the number of nodes).

In this section we focus on studies that explicitly consider the effect of density on readability. We used $\frac{|E|}{|V|(|V|-1)}$ to calculate density. This represents the ratio of the number of existing edges E to the number of all possible edges for the number of nodes V. For simplicity, we multiply this ratio by 100 to achieve percentages.

There are also other ways to derive density. The so-called *linear density* $\frac{|E|}{|V|}$ is often used to compare the number of edges E to the number of nodes V. In this measure, tree-like graphs will have density close to 1. Ghoniem *et al.* [140] suggest using *square-root density* $\sqrt{\frac{|E|}{|V|^2}}$, since its range is bounded to the interval $[0, \frac{1}{\sqrt{2}})$. Any of these definitions can be used; the choice depends on utility [224].

Fig. 4.5 shows the density of real and generated graphs used in studies. Most studies use graphs with densities of 50% or less. There are only three studies that use graphs with densities higher than 50% [159, 310, 333]. All three studies evaluate methods that are tailored to scale well for dense networks.

In fact, all studies that use graphs with ten nodes or more and a density of more than 20% evaluate matrix-like visualisations that scale well for dense graphs [141, 187, 332, 341] or evaluate edge-compression and edge-bundling tools [45, 110].

In summary, in the surveyed studies, dense graphs are mostly small and have less than ten nodes. The only studies that evaluate visualisations using dense graphs ($> 20\%$ density) with more than a trivial number of nodes, tend to be those testing matrix diagrams or edge compression techniques. With the exception of these types of studies, all studies that use graphs with more than 50 nodes, choose to use sparse graphs with a density of less than 10%. 119 out of 152 studies (78%) use graphs with a density of less than 10%, while 129 out of 152 studies (85%) use graphs with less than 20% density.

4.4.4 Number of Timeslices

In the case of dynamic graphs or static graphs with dynamic attributes, and in addition to the number of nodes, and the number of edges or density, the number of timeslices is an important measure of cognitive scalability.

Among the 152 studies (described in 124 papers) covered in this survey, 22 use dynamic graphs. Compared to the reporting of size metrics discussed above, it seems dynamic graph evaluation papers are reasonably consistent about reporting the number of timeslices.

Fig. 4.6 shows a histogram of the number of timeslices. For previous metrics, we used the minimum and maximum values to derive the charts, but since there are only a few studies that use dynamic graphs, we included all the values. There is no visible difference between odd and even numbers. There is, however, a clearly visible spike at six timeslices. Seven studies, out of the total 22, use dynamic graphs with six timeslices.

We suggest that six timeslices is typically chosen as it is small enough for a small multiples representation of a dynamic graph to fit on a screen with each timeslice being at a reasonable scale. Farrugia and Quigley [126] conduct two studies in order to compare animated displays to static ones. They used two graphs with six timeslices each. For the

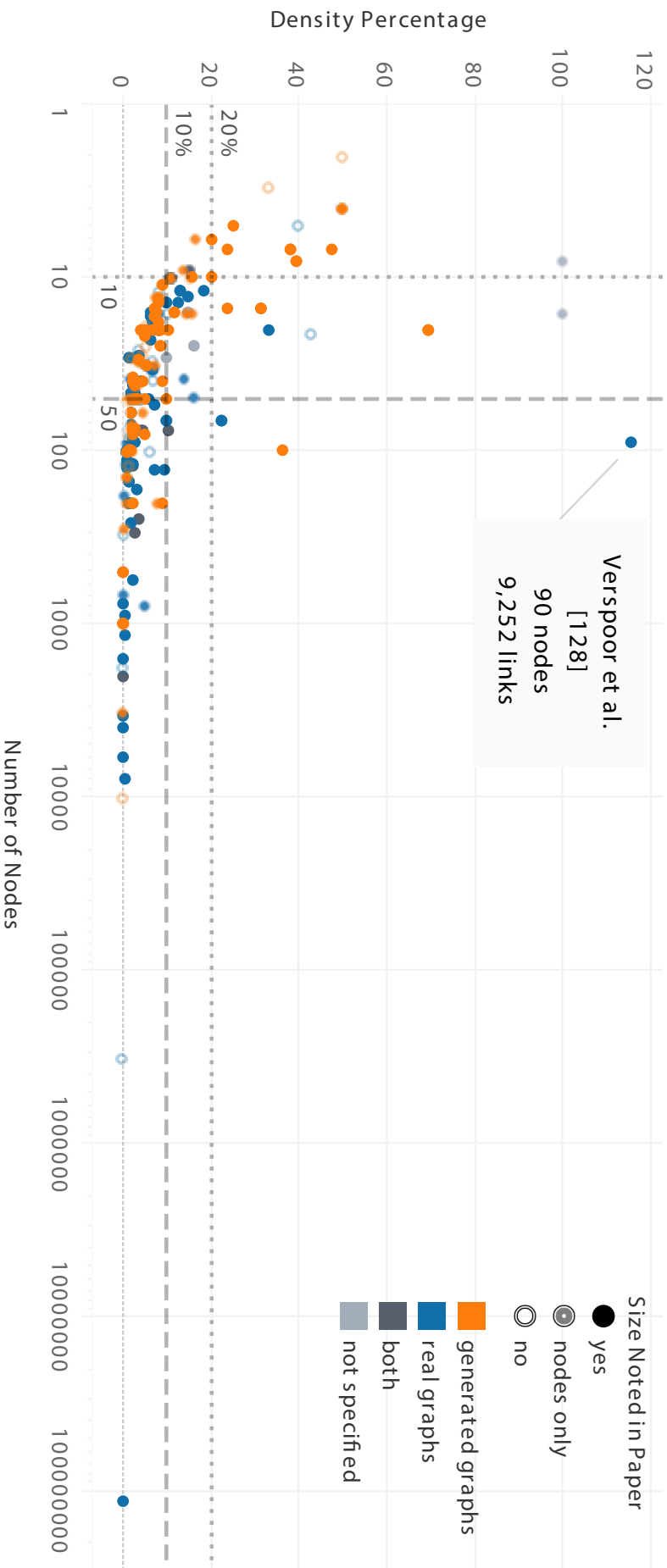


Figure 4.5: The density of graphs used in the studies. Out of 152 studies, 129 studies use sparse graphs with densities of less than 20%, while 119 studies use graphs with less than 10% density.

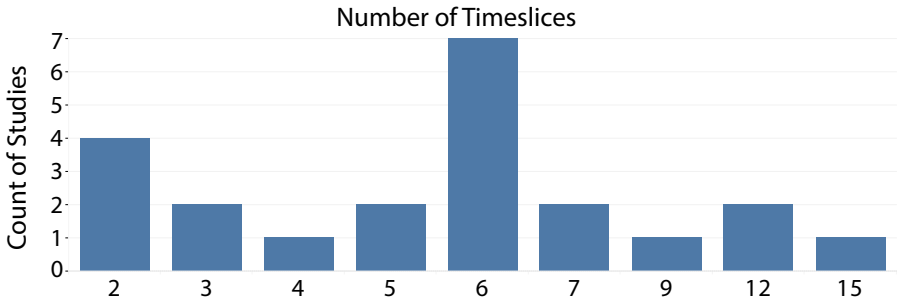


Figure 4.6: Histogram of the number of timeslices of dynamic graphs used in the surveyed studies.

static view, they placed the six timeslices next to each other on a 2×3 grid. Similarly, Archambault and Purchase [35] conduct a study that evaluates the effects of three factors–static versus animated presentation of dynamic attribute values, force-directed versus hierarchical layout of constant graph structure, and 'with history' versus 'without history' persistence, for displaying graphs with dynamic attributes. They use six timeslices, with each covering one-sixth of the screen.

Shi *et al.* [286] conduct the only study that tests a variety of time slice counts. They demonstrate the scalability of various aggregation techniques against the more typical small multiples display of timeslices. They use four dynamic graphs with different numbers of timeslices. They use graphs with 15, 12, 4, and 5 timeslices and 674, 109, 298, and 16 nodes respectively. For their small-multiples condition, they use either a 6 or 24-cell grid to make the full set of timeslices for each graph visible. Unlike the studies above, where the number of timeslices were chosen to fill the small multiplies grid, in this study the odd numbers of timeslices would have left part of the screen unused.

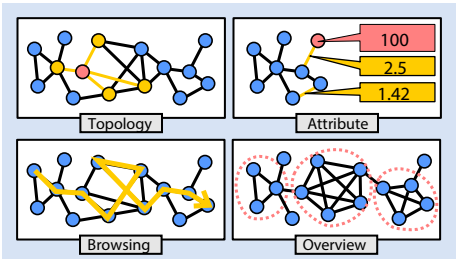
Others choose the number of timeslices according to the norms of specific application areas. North *et al.* [243] use a directed graph with 46 nodes, 36 edges, and 12 timeslices in their evaluation of three existing visualisation methods for dynamic graphs. They claim that this is a typical size of pathways used by biologists.

In addition to the number of timeslices, the number of graph elements that change from one timeslice to another is important. Authors report these values in different ways. For example, some state the maximum number of changed elements [32], while others report on the average [262, 264].

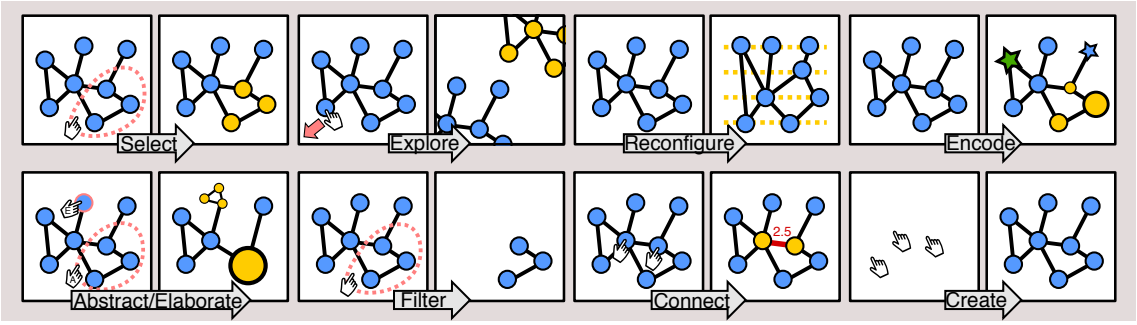
In summary, the number of timeslices for dynamic graphs is often small. Most studies use less than ten timeslices (except for [243, 286]). Moreover, they often consider only a fixed number of timeslices (except Shi *et al.* [286] who use four variations). Some studies pick the number of timeslices to best fit their visualisations on screen. Others choose what is common in the respective application area.

4.5 Other Factors and Scalability

In addition to the basic measures of the previous section, other factors and their interplay influence the cognitive scalability of graph visualisations. In the following sections, we discuss HCI, graph drawing, and study design factors.



(a)



(b)

Figure 4.7: Task and interaction taxonomies. Task taxonomy (a) proposed by Lee et al. [212], which mainly includes *Topology*, *Attribute*, *Browsing*, and *Overview* tasks. Interaction types (b) introduced by *Yi et al.* [334], which include *Select*, *Explore*, *Reconfigure*, *Encode*, *Abstract/Elaborate*, *Filter*, and *Connect*. However, *Create* is newly added in our taxonomy.

4.5.1 HCI Factors

Graph visualisations are often tailored to efficiently serve domain-specific tasks. Modern visualisation tools are equipped with various interaction techniques. Interaction plays a key role in allowing graph visualisations to scale to larger data sets. They were rarely investigated in earlier visualisations, but have been widely discussed in recent times.

In this section, we discuss the interplay between different types of tasks, interaction techniques and application areas on one hand, and the basic measures of graph size on the other.

Tasks

To analyse and understand the trend, as well as the impacts of scalability in relation to tasks, we classified the various tasks used in studies into four main categories based on the taxonomy of Lee *et al.* [212]. Fig. 4.7(a) shows a summary of tasks investigated in our survey: *Topology-based* tasks allow participants to detect node adjacency, accessibility, common connection, and connectivity. *Attribute-based* tasks support identification of nodes and links by the data attributes associated with them. *Browsing* tasks include following or revisiting a path. High-level or deliberately more abstract tasks were classified as *Overview* tasks. Note that we considered tasks that asked the participants to find the shortest path to be in the *Topology-based* category, even if they required some path following. We categorised path following tasks as *Browsing*, only if participants were explicitly asked to follow a certain path.

Table 4.3: The type of tasks used within the studies.

Task	References	# of Studies
Attribute	[35–37, 44, 47, 68, 70, 80, 90, 103, 123, 127, 138, 141, 142, 149, 159, 163, 173, 176, 181, 183, 187, 203, 210, 211, 243, 252, 266, 274–276, 282, 287, 289, 290, 298–300, 326, 341–343, 345, 347]	53
Browsing	[24, 32, 37, 48, 53, 86, 101, 106, 163, 166, 168, 174, 194, 206, 211, 230, 237, 274, 289, 290, 318, 324, 341, 342]	26
Overview	[22, 24, 31, 36–38, 44, 49, 68, 103, 127, 141, 147, 149, 156, 159, 173, 181, 191, 203, 207, 217, 221, 237, 243, 247, 259, 262, 275, 286, 290, 300, 302, 310, 318, 328, 332, 335, 338, 341, 346, 347]	52
Topology	[22–24, 31, 36–38, 45, 47, 49, 67, 70–73, 77, 80, 88, 90, 92, 101, 103, 106, 110, 123, 126, 127, 139, 141, 143, 156, 159–161, 163, 166–169, 171–175, 181, 183, 187, 191, 194, 198, 206, 210, 211, 217, 222, 227, 230, 234, 237, 243–247, 252, 256, 260–264, 266, 271, 274–276, 286, 287, 289, 290, 298–300, 305, 310, 314, 318–320, 325, 326, 330, 332, 333, 336, 341, 342, 345, 346]	114

Table 4.3 shows how the different tasks were used in the 152 studies (noting that several studies used more than one task). *Topology-based* tasks were most common (47%), followed by *Attribute-based* (22%) and *Overview* (21%). The majority of the studies (75%, 114 out of 152 studies) used tasks that are categorised as *Topology-based*.

This might be an indication that researchers perceive tasks concerned with understanding the structure of the graph as good evaluation measures for graph visualisation. However, the number of studies that use *Topology-based* decreases as graph size increases, while *Overview* tasks become more popular. They become equally popular (37%) in studies that use graphs with 300 nodes or more. For graphs with 1,000 nodes or more, *Overview* tasks are used excessively (43%), while *Topology-based* tasks become less common (30%). We believe that this is expected, since *Overview* tasks do not require a detailed understanding of the graph, but deal with general properties and estimates.

In order to understand the combinations of tasks in each study, we plot these as composite nodes in Fig. 4.8. Six studies use only *Topology-based* tasks when using graphs with 500 nodes or more [71, 72, 234, 245, 325]. The first three studies [71, 72, 234] use graphs with a simpler structure, i.e. trees. Moreover, the trees they use have less than 1,000 nodes. The other three studies [234, 252, 325], which use graphs with more than 5,000 nodes, allow participants to interact with the graphs and show only parts of the graph at a given time. This implies that tasks and interactions are mutually reinforcing each other in large graph visualisation.

Marner *et al.* [221] use a large graph with 7,885 nodes and 427,406 edges in their study. They project the graph on a wall-sized display and ask participants to untangle it until they achieve a better overview. We believe that even though their process allowed for an *Overview* task, it would have been almost impossible for the participants to perform more complex tasks, such as path-finding or counting triangles. Similarly, Kwon *et al.* [207] use graphs of up to 113 million nodes and 1.8 billion edges in their study, but only ask the participants to rate the similarities between the graphs. Note that among all tasks, *Topology-based* tasks have been widely used in most of the studies since the connectivity plays a key role for understanding graph structures. Path finding is the most common task (66 over 114 in our survey) among *Topology-based* tasks, and therefore can serve as the primary task for demonstrating the usability of graph visualisation. For example, following the investigation by Bae and Watson [140], and Ghoniem *et al.* [48], it is reasonable to use node-link diagrams for path-finding purposes when the graph is less than 200 nodes.

Another interesting study using *Topology-based* tasks is conducted by Moskovich *et al.* [230]. They use two graphs with 1,000 nodes. The sparser one has 1,485 edges, while the denser one has 2,488 edges. We believe that they wanted a graph large enough to make

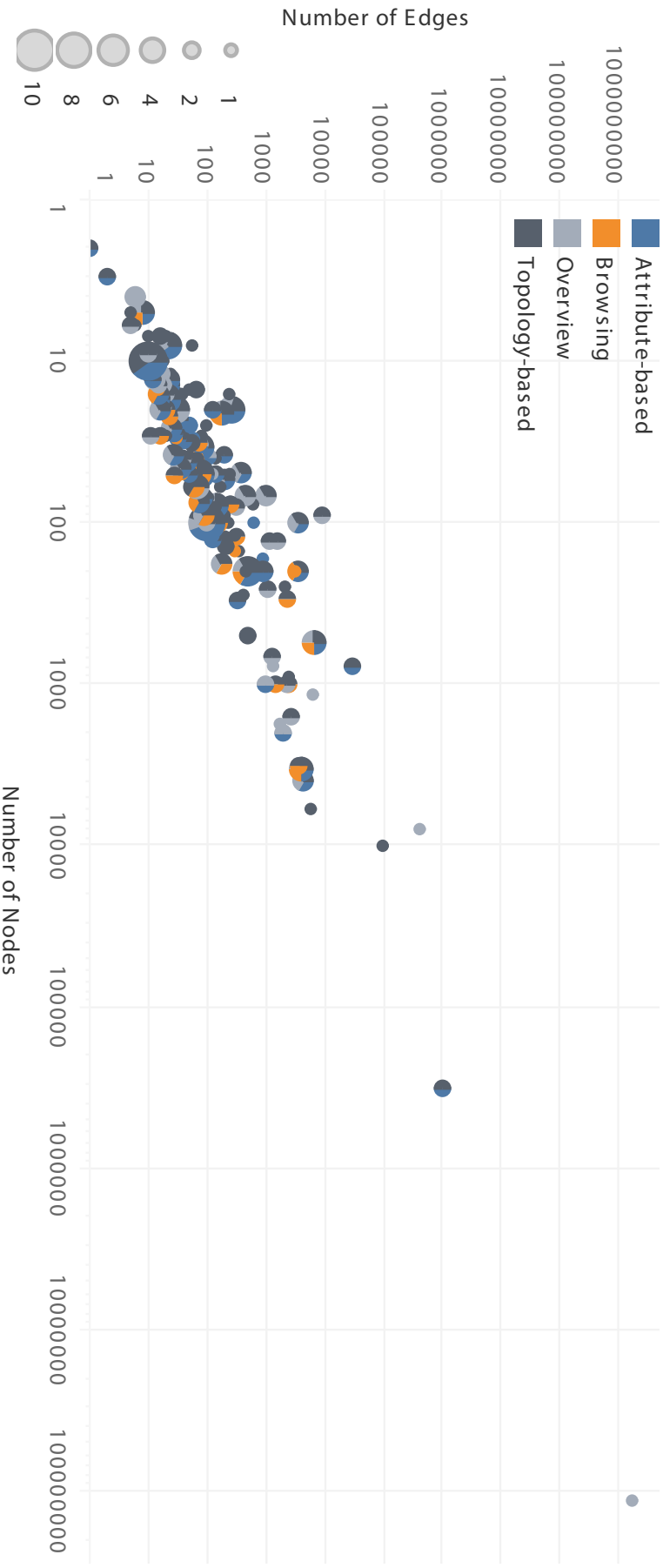


Figure 4.8: Types of different categories of tasks used in studies with relation to the number of nodes (on the x-axis) and the number of edges (on the y-axis). The areas of the circles reflect the count of studies that use similar values. Both axes have log scales.

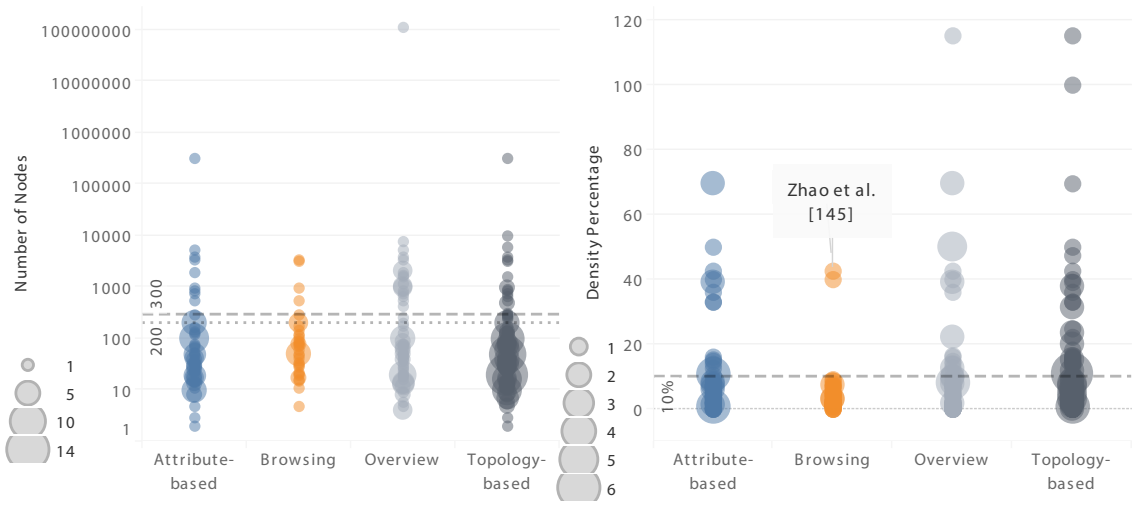


Figure 4.9: Types of different categories of tasks used in studies with relation to the number of nodes (on the left) and the density (on the right). The areas of the circles reflect the count of studies that use similar values.

the task difficult, especially when finding the immediate neighbours of a given node. The authors propose an interactive navigation technique *Bring-and-Go*, which aids this task by bringing all adjacent nodes closer to a selected node. This study demonstrates the complexity of these tasks on graphs with a large number of edges. Both error rates and performance times increase significantly from the sparser graph to the denser graph.

Some authors discuss the choice of graph sizes in relation to the results of their studies. Huang *et al.* [169] justify the increase in errors by explaining that human perception and cognitive systems become overburdened when dealing with large graphs, even with 25 nodes and 98 links. Okoe *et al.* [245] use a graph with 900 nodes and 2,500 edges. However, they mention a high error rate of more than 50%. They associate this to the difficulty of the tasks. They explain that the large number of edges voided the highlighting advantage of the evaluated technique. Wong *et al.* [325] mention that they tried to use graphs that were not too complex, but their attempts were not successful for all the tasks.

Only nine out of the 53 studies that require participants to perform *Attribute-based* tasks, use more than 200 nodes [37, 68, 90, 103, 181, 210, 252, 290, 347]. Two of these studies aggregate multiple nodes into singular representations: *motifs* [103] and *metanodes* [37]. It is natural to assume that *Attribute-based* tasks would be difficult on graphs with too many elements, since *Attribute-based* tasks are related to data attributes associated to nodes and links. While this is backed by our survey for number of nodes, the number of edges ranges between 10 and 1,000 for most studies using *Attribute-based* tasks.

For *Browsing* tasks, we assumed that experimenters would use sparse graphs. This is backed by our findings as shown in Fig. 4.9. Also, in addition to the density of the graphs, studies that require performing *Browsing* tasks use graphs with fewer number of nodes (≤ 200), with the exception of five outliers [37, 206, 230, 290, 318]. Four of these outliers use interactive highlighting to assist the participants in performing the task [206, 230, 290, 318], while the fifth [37] uses *metanodes* and *metaedges*, which are aggregations of multiple nodes and edges into singular representations. As seen in Fig. 4.8, *Browsing* tasks are only used on very sparse graphs ($< 10\%$), except for one study by Zhao *et al.* [341] that uses graphs with 40% and 42.9% densities. The latter allows participants to

highlight specific nodes and their connections, thus showing small subsets of edges at a given time.

In summary, *Topology-based* tasks are the most common tasks to evaluate graph visualisation; nonetheless, the number of *Topology-based* tasks is relatively reduced when using graphs with more than 300 nodes. In such cases, *Overview* tasks become more common, especially when the tasks are assisted with multiple interaction techniques. Sparse graphs ($< 10\%$) with few nodes (≤ 200) are used when asking the participants to perform *Browsing* tasks. Similarly, *Attribute-based* tasks are not common in studies that use graphs with more than 200 nodes.

Interaction

Early graph experiments did not make much use of interaction; their focus being on the interpretation of static graph drawing–often being simply presented on paper (e.g. [256, 259, 260]). In more recent years, several experiments have tested the worth of node-link diagrams using interactive systems that permit more extensive exploration of the relational information. In such cases, the use of interaction techniques is often a crucial component of the study design.

We used the interaction taxonomy of Yi *et al.* [334] as a means of classifying the different types of interaction used in the experimental studies: *Select* allows users to mark objects on screen as interesting; *Explore* enables users to navigate a hidden subset of data; *Reconfigure* changes spatial arrangement; *Encode* allows users to transform data values to their preferred visual mapping (e.g. colour, size, or shapes); *Abstract/Elaborate* enables users to adjust the level of detail of a data representation; *Filter* allows users to prevent the display of data fulfilling given conditions, and; *Connect* allows the highlighting of relationships between data. For the purposes of this survey, we added an additional category: *Create*, which allows users to generate graph drawings.

Of the 152 studies, 80 used no interaction at all. Table 4.4 shows how the remaining 72 used interaction (noting that several studies used more than one interaction technique). Of all the occasions when interaction techniques were used, *Abstract/Elaborate* was most common (20%), followed by *Select* (19%), *Explore* (16%) and *Reconfigure* (16%). Thus, there is no single category that dominates (unlike our finding with the task category).

Fig. 4.10 shows the different types of interaction used in the studies, with respect to graph size. We note that 25 of 60 studies (42%) use a single interaction type for those graphs with 500 nodes or fewer, while for those studies using graphs with more than 500 nodes, this ratio drops to 32% (6 out of 19 studies).

Tiny graphs (e.g. 4 nodes and 5 edges) are easy to understand without the support of interaction (see lower dashed lines shown in Fig. 4.11), although even studies that used graphs with as few as ten nodes and ten edges used interaction. The majority of studies that do not use any interaction use graphs with 100 nodes or less (70 / 80 studies, 88%) and 500 edges or less (71 / 80 studies, 89%) (upper dashed lines in Fig. 4.11). These bounds allow us to group the interaction techniques into two sets: techniques used on graphs with greater than 100 nodes and 500 edges: *Abstract/Elaborate*, *Connect*, *Explore*, *Reconfigure*, *Select*; and techniques used on graphs with fewer than 100 nodes and 500 edges: *Encode*, *Create*, and *Filter*. If we consider the degree of effort required by the user in applying these techniques, the first set can be considered relatively straightforward – the techniques are usually tested by having participants apply the built-in functionality to look at the data in a different way until the answer is obvious. By contrast, *Encode*

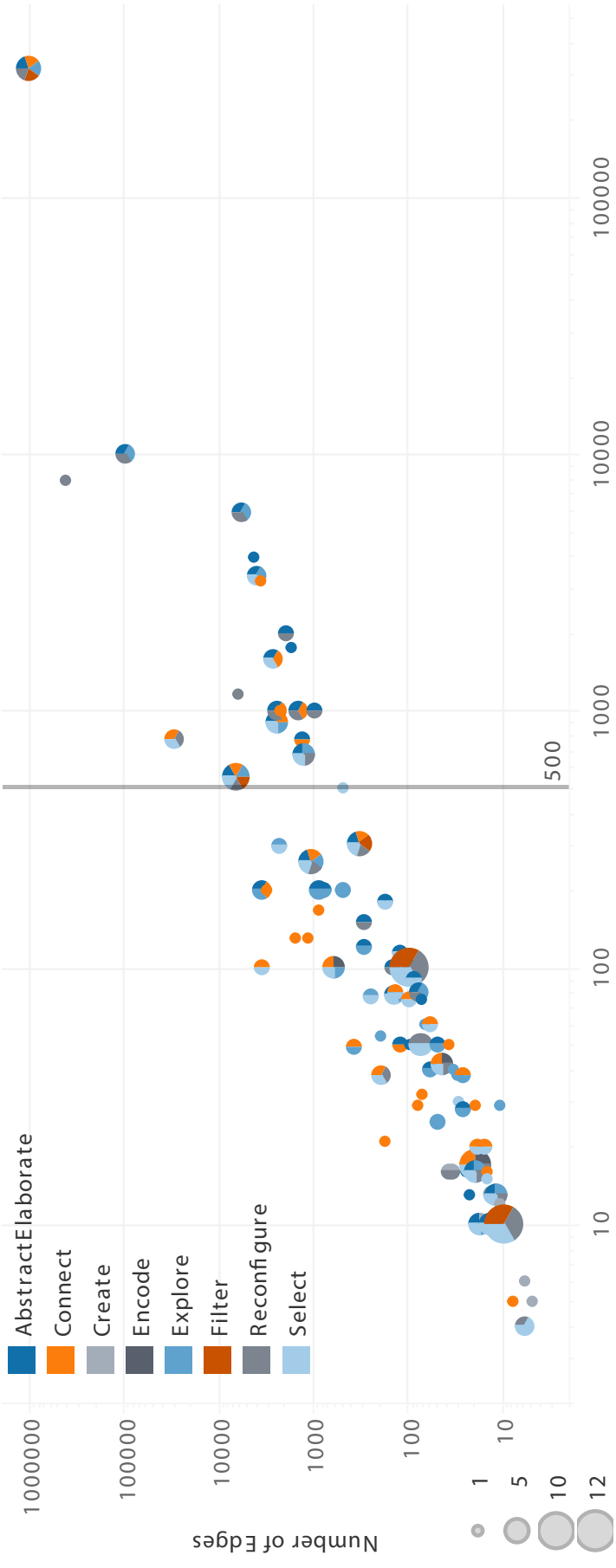


Figure 4.10: Types of different interaction used in studies with relation to the number of nodes and the number of edges. Both axes have log scales.

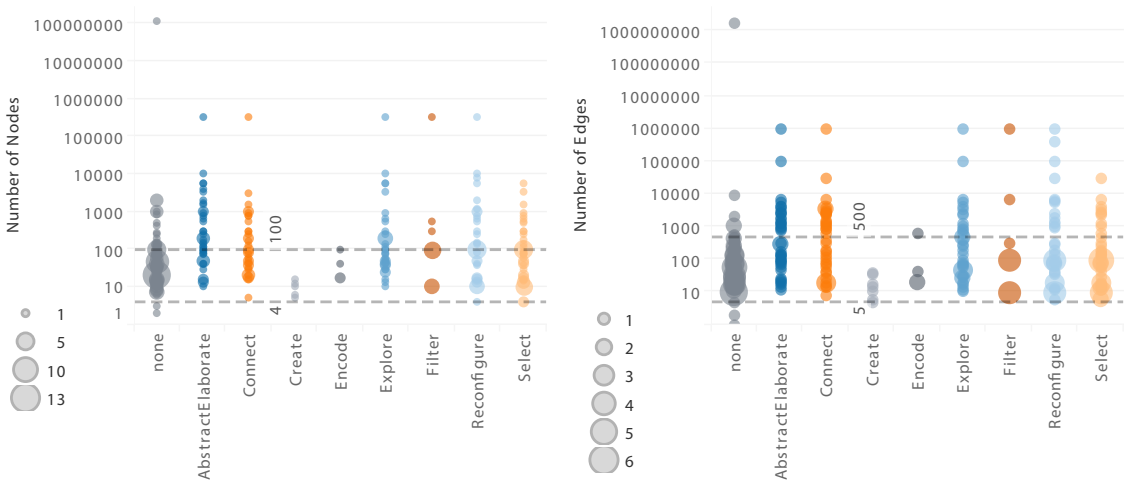


Figure 4.11: Interaction types used in studies with relation to the number of nodes (on the left), and the number of edges (on the right).

Table 4.4: The type of interaction used within the studies.

Interaction	References	# of Studies
None	[22–24, 31, 32, 35, 36, 45, 47, 49, 67, 70, 71, 73, 80, 92, 101, 110, 126, 143, 147, 149, 159–161, 166–169, 171–175, 181, 183, 187, 191, 203, 207, 227, 243, 246, 256, 259–262, 264, 282, 298–300, 310, 320, 328, 330, 332, 333, 335, 336, 345, 346]	80
Abstract-Elaborate	[24, 37, 68, 86, 90, 103, 123, 139, 142, 198, 211, 217, 230, 234, 237, 245, 247, 252, 263, 276, 286, 287, 289, 290, 302, 319, 324–326, 338, 347]	32
Connect	[48, 77, 88, 90, 123, 127, 141, 156, 163, 176, 194, 210, 211, 217, 230, 245, 247, 252, 290, 314, 318, 319, 338, 341–343]	29
Create	[222, 244, 263, 305]	4
Encode	[123, 176, 342]	4
Explore	[37, 38, 68, 123, 127, 138, 139, 176, 198, 206, 211, 234, 245, 247, 252, 263, 271, 274–276, 286, 287, 290, 325, 326]	26
Filter	[44, 90, 123, 252, 266, 290]	9
Reconfigure	[24, 44, 90, 106, 123, 191, 210, 221, 230, 234, 247, 252, 266, 286, 290, 305, 325, 326, 342, 347]	25
Select	[37, 53, 68, 72, 86, 88, 90, 106, 141, 176, 191, 194, 206, 210, 217, 237, 245, 247, 263, 266, 286, 290, 326, 342]	30

and *Create* require more effort on the part of the users, since (possibly creative) decisions need to be made.

Interaction tasks can come with a time trade-off, however, as an example, finding labels inside an abstract node (which represents a set of aggregated nodes), could be more time consuming due to the interaction, than the task of finding labels outside of it, which is always completed faster [103]. Care, therefore, needs to be taken in the interpretation of the timing data collected (especially if participants are allowed unlimited time to perform their task), since such data might include superfluous interaction.

To conclude, the combination of multiple interactions can improve the visualisation of large graphs, and this is more significant as graph size increases.

Application Areas

There are several domain-specific studies where the aim is enhanced understanding of the content, and so the tasks were clearly focused on the domain knowledge (e.g. [244, 266]). For example, Tanahashi *et al.* [299] performed a comparative study of four different ways of presenting data for the purposes of introducing information visualisation to novices,

where a graph drawing was one of the visualisation types. Their analysis not only considered the efficacy of the visualisations, but also looked at two different types of learning (active and passive), and two different teaching methods (top-down and bottom-up). They were therefore able to propose guidelines for writing effective information visualisation tutorials. North *et al.* [243] compared two different types of evaluation (benchmark and insight) using three different visualisation alternatives depicting gene expression data, and all tasks were related to understanding of the data.

While some application areas have well defined and restricted characteristics for the networks under investigation, in others there can be a large range of impacting factors that potentially affect scalability. In the life sciences, a variety of network types are investigated, ranging in size from a few dozen to a few million nodes, and showing a similar variety in other network characteristics, e.g. density or diameter. In addition, the required tasks can differ significantly between use cases, e.g. from deciding reachability to the detection of dynamic patterns, which affects the limits of readability. RNA sequence graphs, like the ones used in [221], can have up to several million nodes and edges and dense local structures, but they often have a very sparse global structure, making the visual detection of so-called repeats (loops that indicate repetitive structures in RNA sequences), a feasible task. On the other hand, metabolic pathways like the ones used in [347] are often planar or near-planar graphs with low local and global density, requiring that the semantics associated with the metabolic flow are incorporated in the layout and the visual representation. While these pathways are parts of a large and complex network of metabolic reactions in an organism, the visual analysis is often restricted to such sparse sub-networks that have a specific functionality, e.g. the synthesis of a particular biomolecule.

4.5.2 Graph Drawing Factors

For several decades now, the field of Graph Drawing has led to the development of efficient algorithms for layout computation as well as graph visualisation metaphors, and has also investigated the impact of the resulting visualisations on readability and task performance. For some years, a main focus has been on the computational complexity and scalability of algorithms, but since the development of methods that scale to several million nodes and edges, the focus has shifted to the visual complexity and human interpretability of the resulting layouts [112].

As layout methods differ in computational scalability, in their performance on certain graph classes, and also in the features and characteristics of the resulting layouts, the interplay between graph structure and layout method used in studies will strongly impact the limits of cognitive scalability. Comparisons between different methods are rare, as the selection is often motivated by real-world application requirements. While constraint-based methods can create high-quality layouts, which might be of interest for studying the limits of cognitive scalability, the methods do not scale well regarding the computational complexity. Thus, for large graph sizes, researchers have to resort to fast heuristics, e.g. multi-level force-based methods.

Graph Type and Structure

While the number of nodes and the density can give a first indication on the complexity of a graph with respect to cognitive scalability, a more fine-grained description of the structure is necessary to investigate its impact. Certain graph classes might result in layouts

with minor quality, e.g. low diameter graphs like friendship networks can lead to hairball drawings when standard force-directed methods are applied. On the other hand, graphs with a globally sparse structure like the ones used by Marner *et al.* [221] are well suited for untangling and structure identification tasks, even when the number of nodes is huge, as the global and local structures simply scale with the size.

A further distinction can be made regarding the use of directed and undirected graphs. While 103 of the studies used undirected graphs, only 33 of them used directed graphs, and 15 used trees. With the exception of the two near-tree sparse graphs from the *OnGrax* study [347], and one graph from a study on directed edge representations [160], all graphs with more than 200 nodes are either undirected or trees. Some studies based their graph selection on real world examples. In some cases, the graphs were taken from specific application areas. Most commonly: social networks [24, 35, 67, 123, 126, 127, 169, 173, 210], followed by co-authorship networks [23, 44, 156, 286, 341] and biological networks [191, 221, 234, 243, 347]. *Shi et al.* [286] use four dynamic graphs from two domains– communication and co-authorship networks.

Layout Method

The distribution of layout methods in the investigated studies shows the expected dominance of force-directed methods. Variants of this class of methods scale well computationally, and appear to be the preferred method used in a variety of publicly available graph visualisations and systems. Together, with the linear time tree layout methods, these methods are the only ones used for studies with more than one thousand nodes. More than 50% of the studies used a force-directed layout to draw the networks, followed by multiple types, used in around 18% of the studies. Note that our classification of layout methods specifies the initial layout the subjects are presented with, and does not consider whether the subjects could manually, or by means of an algorithm, change the layout in an interactive interface.

While the practical computational scalability of many methods changed due to improved algorithms and implementations, the relative performance stayed the same with force-directed and tree layouts being by far the fastest, and methods that require solvers, e.g. constraint-based methods, being slowest.

Borkin *et al.* [68] present a radial-based tree layout to display file system provenance and motivate their choice, with the failure of node-link diagrams to show a high-level summary for the large-scale provenance data graphs. Their study results indicate that users were more efficient with the interactive radial layout representation than with an existing conventional node-link diagram tool.

The range of the basic graph size metrics for each of the layout methods fits the expectation. Manual layouts are only performed on graphs of 120 nodes or less, with the notable exception of the study on the collaborative graph visualisation system *OnGrax* [347]. In this system, however, an initial layout was given, which was manually created by domain experts, and the user could simply rearrange this layout manually.

4.5.3 General Study Design Factors

Just over half the studies (92 / 152 studies, 61%) follow a typical design of asking participants to perform graph reading tasks under different conditions (the independent variables - often different layouts, different visualisations or different interaction techniques) and

collecting response time and accuracy data as the dependent variables (with some also collecting preference choices). Most studies rely on these three dependent variables to measure the efficiency of the visualisations at hand.

With respect to graph size, there is no discernible difference in the sizes used for typical graph reading studies and the others–both categories have similar distribution of graph sizes.

Some experiments collected process data as the main dependent variable (e.g. [71, 72]). Others collected eye-tracking data [77, 237, 319, 335]. *Wong et al.* [326] counted the number of mouse actions (click, move, zoom, pan, and turn) as a measure of extent of interaction with different forms of visualisation of labelled graphs, while *Nekrasovski et al.* [234] looked closely at the mouse drag action.

There were 129 studies that used a within-participants study design, where 22 used between-participants, and one used a mixture of both. Within-participants studies are popular because they have the advantage of eliminating any effects relating to variability between the participants, and, while they may be subject to the learning effect, the effects of this are easily mitigated by appropriate randomisation. They do, however, tend to take longer than between-participants experiments, where the participants can have more time to work with only a selected few of the stimuli (rather than all of them, as it is in the within-participants case). Thus, the tasks for within-participants studies tend to be smaller and simpler than those used in between-participants studies.

We might expect that there would be more between-participants' experiments in recent years, since such experiments are more suitable for crowd-sourcing: within-participant experiments tend to take too long to be appropriate for crowd-sourced participants. However, this is not the case - there is a similar publication year profile for both categories of study.

4.6 Size Rationale

In the previous sections we discussed how different metrics could affect scalability. This was mainly done based on what we could gather from the aim and the results of the studies. In "Basic Measures of Scalability" we reviewed the range of number of nodes and edges for graphs in studies, and discovered that most studies using large numbers of nodes and edges want to highlight the advantages of using specific techniques. However, in some cases, the authors explicitly mention the reasons for picking a specific number of nodes or edges. Some authors performed pilot studies which allowed them to determine the size of graphs that would best suit their evaluation. Others were based on the authors' experiences and understandings of the requirements. We discuss these further in the following sections.

4.6.1 Pilot Studies

The papers in our survey rarely mention pilot studies that determine a ceiling or floor affect to scalability. We believe that many conduct pilot studies but do not mention them explicitly. In this section, we discuss a select few that provide the reasoning for their pilot studies and the decisions made with regards to factors that affect cognitive scalability.

Archambault *et al.* [37] mention that the largest graph size for their study was determined by pilot studies. They start with small, medium and large sized graphs; however the pilot participants could not complete the tasks on the large graphs. Thus, instead of

the large graph, they use a smaller graph. The largest graph they use has 3,351 nodes and 4,083 edges. Archambault and Purchase [35] use a pilot study to find a reasonable graph size. They use a maximum of 50 nodes and 100 edges. Similarly, Dawson *et al.* [92] mention a pilot study to balance density and difficulty. They use graphs with 75 nodes and 150 edges.

Some design their pilots to find reasonable graph densities to allow the participants to finish the tasks in a restricted time limit. Kobourov *et al.* [198] find 120 nodes and 2.5 density as maximum measures to complete the tasks under two minutes, while others use pilot studies in order to explore different layouts and graph structures for specific tasks; e.g. Netzel *et al.* [237] aim at finding thresholds for assisting tasks of finding the longest link and biggest cluster.

Saket *et al.* [276] mention a pilot study where they chose 50 nodes as minimum and 200 nodes as maximum. They also chose three density levels N, $2N$, and $4N$. Borkin *et al.* [68] conducted a pilot study to determine the boundaries between the easy and difficult tasks. They mention that graphs of 10s, 100s, 1000s, and 10,000s of nodes were compared. They classified trees with 42 to 346 nodes as easy, while trees with 1,192 to 5,480 nodes as hard. Similarly, Marriott *et al.* [222] note that their pilot study showed that larger graphs beyond 6 nodes were too difficult to be memorised. Conversely, Xu *et al.* [330] rely on a pilot study and choose 100 as the maximum number of nodes for their graphs. They further increase this to 200 in their second study, due to the lack of a ceiling effect in their first study.

4.6.2 What is Small? What is Large?

Testing the boundaries of readability/scalability may generally not be a primary goal for experimental studies. Instead, researchers might pick a size range that they consider acceptable in order not to have scalability as a confounding factor in their results. In addition, technical limitations, such as screen size and resolution, and also typical requirements from application areas, like characteristics of occurring networks of interest, might play an important role in the determination of graph sizes, but are often not reported explicitly and will also change over the years. Furthermore, there might be standard benchmark sets used, or simply graphs picked based on availability, instead of using graphs that allow one to investigate scalability effects.

Some studies do not mention pilots, but justify their choices of graph size as an attempt to meet particular requirement for their studies. Archambault *et al.* [38] use real graphs, with the largest having 60 nodes and 68 edges. They explain that they chose two data sets each consisting of realistic size and structure.

Sometimes, the rationale is task-oriented. For example, Kieffer *et al.* [191] justify their use of small graphs to allow the participants to manually draw the graphs in a reasonable amount of time. Similarly, Purchase *et al.* [263] use two graphs with 10 nodes and 11 and 18 edges, to make it manageable for the task of drawing the networks. Blythe *et al.* [67] note that they use a small graph in order not to overwhelm the participants with the amount of information. They use a small graph with 12 nodes and 24 edges.

Others, such as Hlawatsch *et al.* [159], justify using small graphs to avoid the need for interaction. They use ten graphs with eight nodes and 22 to 40 edges. They also use ten graphs with 20 nodes and 147 to 264 edges. Similarly, Alper *et al.* [23] chose not to vary the graph size drastically in order to avoid the requirement of zooming.

Zhao *et al.* [343] justify their size selection to fit the diagrams to screen. They use graphs with 167 nodes and 902 edges in their evaluation of a visualisation called *MatrixWave*. Kadaba *et al.* [183] mention that they needed graphs that were small enough to be memorisable. They use a daisy-structured graph with 11 nodes. In a second study they use smaller graphs with 3 nodes and 2 edges. Holden and Van Wijk [161] explain that they wanted to generate graphs with an adequate number of vertices, without causing a large amount of visual clutter resulting in an excessively high edge density.

In contrast, some studies use large graphs in order to highlight the benefits and improvements of some techniques with respect to scalability. Lee *et al.* [211] justify their selection of graphs with 200 nodes as complex graphs. They also state that 200 nodes is considered to be an upper bound for currently studied food webs. Huang *et al.* [171] mention that they choose graphs with a density ranging from 10% to 20%, and their drawings have the same crossing ratio of 40% in order to have reasonably complex graphs. The largest graph they use has 50 nodes and 245 edges. Dwyer *et al.* [106] state that the graphs they used were larger in size (50 nodes), than ones (17 nodes) used in another study which inspired their work, while Okoe *et al.* [247] justify their selection of a graph with 258 nodes and 1090 edges as larger than previously used graphs, yet sufficiently small to be evaluated in a browser.

To conclude, some authors provide a rationale for their selection of graphs with a specific range of node and edge counts. These often resemble our discussions and hypotheses, nonetheless, we wanted to keep a clear separation between what constitutes our opinions and the rationale provided by the experimenters.

4.7 Research Trends

By analysing our survey data from a historical perspective, we tried to gain some insights about the development of the research community, including trends with respect to graph sizes, participants, tasks, and interaction types used in the studies.

Of the studies surveyed in this chapter, 40% were published in TVCG/InfoVis, 16% at Graph Drawing, 13% in the Infromation Visualisation Journal, and another 11% at CHI. Historically, the first seven studies [67, 70, 256, 259, 260] were all published at Graph Drawing between 1995 and 2000. The first study [266] in the Information Visualisation Journal appeared in 2002. The first one [324] in TVCG/InfoVis was in 2003, whereas the first one [234] at CHI in 2006, and the first one [346] at EuroVis in 2009.

The seminal papers [67, 260] of 1995 focused on static graphs and were followed by another ten papers (with a total of 16 studies) on static graphs. The first study on dynamic graphs [262] was published in 2006, more than ten years after the first one on static graphs. As we cover a period of 24 years from 1995 to 2018, it is reasonable to compare the first and second half of the studied period with the first half ending in 2006, and the second half starting in 2007. Whilst during the first half, the average publication frequency was around two studies per year, it considerably increased to 11 studies per year for the second half. Of these studies, on average, nine focussed on static and two on dynamic graphs.

Fig. 4.12 shows the number of nodes of static and dynamic graphs used in studies in different years. While for static graphs the number of nodes increased from below 20 in 1995 to several thousand ten years later, the graph size of dynamic graphs stayed 100 or below (with one exception [286] in 2015).

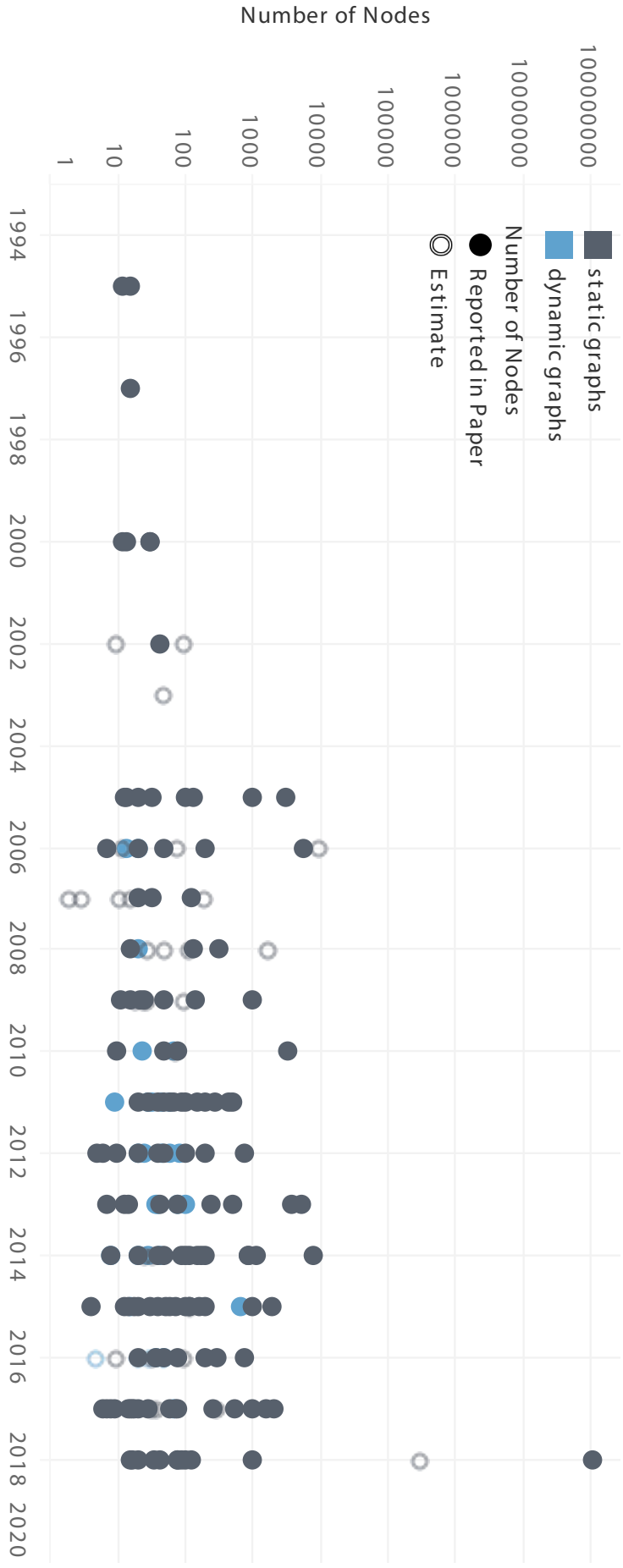


Figure 4.12: The number of nodes of static versus dynamic graphs used in studies for the period from 1995 to 2016.

Since we observed a considerable increase in the graph size for static graphs, we expected that researchers would also try to recruit more participants for their studies. It turns out that there was actually a big difference, if we compare the first and the second half of our studied period. In the first half the median number of participants, for studies that used static graphs, was 14. In the second half it increased to 21. Interestingly, a closer look at the first half of the studied period reveals a dramatic drop of the median number of participants. It decreased from 75 for the original 7 studies [67, 70, 256, 259, 260] (all published at GD) of the period from 1995 to 2000 to nine for the subsequent 11 studies [141, 173, 266, 318, 320, 324, 326] from 2001 to 2005 (mostly published in the Information Visualisation Journal).

4.8 Conclusions

4.8.1 Specific Findings

We report the following key findings:

- There are some clear 'default' numbers of nodes and edges used by most studies. At a maximum, most studies use graphs with 100 nodes, followed by 50 nodes and 100 edges, while their smallest graphs have 20, followed by 50 and 10 nodes, and 10 edges. The round numbers used by most studies suggest the choice of size to test is somewhat arbitrary, as opposed t being based on empirical evidence of cognitive limitations.

- 80% of studies (121 / 152 studies) use graphs with 100 nodes or less, while only 37% (56 / 152 studies) use graphs with more than 100 nodes.

- 74% of studies (113 / 152 studies) use graphs with 200 edges or less, while only 34% (52 / 152 studies) use graphs with more than 200 edges.

- 70% of studies (23 / 33 studies) that use graphs with more than 200 nodes use interaction and aggregation techniques to show only parts of the network to the participants at a given time.

- Most of the studies that use graphs with more than 1,000 edges evaluate tools that are intended to be able to cope with a substantially large number of edges, or are oriented towards performing well for networks of specific structure, e.g. densely connected communities.

- Only 12% (18 / 152 studies) of the studies surveyed use graphs with a density of more than 20%.

- Studies that use graphs with more than 20% density either use small graphs (< 10 nodes), evaluate matrix representations, or evaluate edge bundling and compression techniques.

- 32% of studies (7 / 22 studies) that use dynamic graphs use six timeslices. This is often to best fit small multiples representations to screen.

Table 4.5: Four categories of graph size based on number of nodes.

	# of Nodes	# of Studies	# of Studies with no interaction	# of Studies with Overview Tasks
small	≤ 20	62 (41%)	39 (63%)	19 (31%)
medium	$[21,50]$	50 (33%)	31 (62%)	12 (24%)
large	$[51,200]$	56 (37%)	27 (48%)	16 (29%)
v. large	> 200	33 (22%)	10 (30%)	19 (58%)

Table 4.6: Four categories of graphs based on linear density.

	Linear Density	# of Studies	# of Studies with no interaction	# of Studies with Overview Tasks
tree-like & disconnected	$[0,1.0]$	49 (32%)	20 (41%)	15 (31%)
sparse	$[1.01,2.0]$	67 (44%)	40 (60%)	20 (30%)
dense	$[2.01,4.0]$	31 (20%)	18 (58%)	7 (23%)
v. dense	> 4.0	26 (17%)	12 (46%)	13 (50%)

- The most common type of tasks is *Topology-based* (114 / 152 studies, 75%), however for graphs with 1,000 nodes or more, *Overview* tasks prevail (13 / 18 studies, 72%).

- *Attribute-based* tasks are commonly used for graphs with 200 nodes or less (47 / 53 studies, 89%) and less than 10% density (42 / 53 studies, 79%).

- *Browsing* tasks are commonly used for graphs with 200 nodes or less (24 / 26 studies, 92%) and less than 10% density (25 / 26 studies, 96%).

- Studies with larger graphs (> 500 nodes) tend to use multiple types of interaction, while using a single type of interaction is more common in studies with smaller graphs (≤ 500 nodes).

- The majority of studies with no interaction used graphs with 100 nodes or less (70 / 80 studies, 88%) and less than 10% density (57 / 80 studies, 71%).

- The tasks of *Abstract/Elaborate*, *Connect*, *Explore*, *Reconfigure*, and *Select* were used on larger graphs, while *Create*, *Encode*, and *Filter* were used on smaller graphs (≤ 100 nodes and $< 10\%$ density).

4.8.2 Discussion

While there has been a significant focus on computational scalability of node-link diagrams layout and rendering, in a race to visualise the largest networks, it seems researchers have understudied human cognitive limitations in understanding such diagrams. A better knowledge of cognitive scalability in this regard would have several benefits. In tools that allow the user to interactively explore a large network through neighbourhood or aggregated views, tool developers could more intelligently control the number of elements in these views. Furthermore, if our community could give clear and informed guidance to users of graph visualisation it would help them to select the right tool for their purpose. For example, in creating figures for papers, biologists reporting on the interactions of particular proteins may be better off showing a focused neighbourhood around those specific proteins rather than providing a hairball. Similarly, if the users were experimenters, they

would choose their corpora, design their tasks and pick layout methods based on these well-defined limits.

Thus, the aim of this survey was to explore factors that would affect cognitive scalability via reviewing existing empirical studies that have used node-link diagrams. We have noticed that controlled experiments tend to focus on graph datasets of size within a fairly limited window (tens to a few hundred nodes and low density). We also discovered that even though the most common type of tasks performed on a network, in general, is related to the topology of the network, overview tasks become more popular for larger networks. Similarly tasks related to detailed attributes associated to the nodes or edges and browsing are not common on networks with more than a couple hundred nodes. With regards to interaction, studies with large graphs tend to allow for more than one type of interaction. Furthermore, we discovered that some interaction types, such as *Create*, *Encode* and *Filter* are only used on small graphs, while others, e.g., *Abstract/Elaborate*, *Connect*, *Explore*, *Reconfigure*, and *Select* are also used on large graphs.

This survey has also helped identify some weaknesses in the design and reporting of empirical studies that use node-link diagrams. For example, several studies do not report on the sizes of the graphs used, while others do not report on the number of elements visible to the participants at a given time. We provide a list of recommendations to overcome these weaknesses in future studies.

A motivation for this work was to identify or validate terms that are used to categorise ranges for graph size. Table 4.5 presents four categories with respect to number of nodes. According to the surveyed studies, there are clear cuts at 20, 50, and 200 nodes. We categorise these into ranges that represent small, medium, large and very large graphs. We categorise sparse, dense, and very dense graphs based on linear density in Table 4.6. Hopefully this breakdown gives future researchers a clear motivation for selecting different graph sizes for their studies. For example, there are only seven studies that use dense graphs with overview tasks.

Our findings indicate a threshold at 200 nodes and 10% density. This threshold is respected by empirical studies that include tasks requiring a detailed analysis of the network. Nonetheless, we believe that this threshold is a result of the expert intuition of the researchers, rather than empirical research. A controlled study is needed to validate and refine this threshold, thus in the next chapter, we present a user study that explores the cognitive scalability of node-link diagrams.

Chapter 5

Cognitive Load - A User Study

"here be dragons"

Used on maps to connote uncharted territories.
The classical phrase was:
"here are lions"

Certain layout characteristics are known to be detrimental to the readability of node-link diagram representations of network data, as discussed in Section 2.4. There has been a large amount of work that focuses on designing algorithms that compute layouts that minimise such characteristics. In Chapter 3, we present a novel *Ultra-Compact Grid Layout* model that results in diagrams with optimal quality, with respect to specific characteristics. However, it is widely understood that even the best network layout algorithms ultimately result in 'hairball' visualisations when the graph data reaches a certain degree of complexity, requiring simplification through aggregation or interaction (such as filtering) to remain usable. There has been little work towards understanding the cognitive limits of humans when working with network diagrams. To this end, we perform a controlled study to understand workload limits when performing tasks that require a detailed understanding of the network topology, such as finding shortest paths in graphs with various orders and densities. We use performance measures (accuracy and response time), subjective feedback, and physiological measures (EEG, pupil dilation, and heart rate variability) in order to explore the effects of graph size (order and density) on cognitive load. We also explore the effects of layout features, including number of crossings, angle of crossings, Euclidean distance, and turning angles on cognitive load. Our results show that there is a significant drop in the efficiency of node-link diagrams to find shortest paths, when using graphs with more than 50 nodes and a density of four.

The work discussed in this chapter was done in collaboration with my supervisors Tim Dwyer, Kim Marriott and Michael Wybrow. We have submitted an article describing this work to the IEEE Conference on Information Visualization (InfoVis) 2018.

5.1 Introduction

Visualisation helps analysts to understand and explain complex data. However, there exist factors that limit the amount of information that can be visualised. Scalability is a

major issue in visualisation design. Eick and Karr [116] discuss how human perception, monitor resolution, visual metaphors, interactivity, data structures and algorithms, and computational infrastructure affect visual scalability. For network visualisation, the last five factors have been well explored [179]. However, human perception and cognition remain understudied.

Surveys like that of Jankun et al. [179] speak about the so-called 'hair-ball effect', wherein, node-link diagram representations of small-world or scale-free graphs beyond some unquantified density threshold, are no longer useful for understanding the connectivity of all but peripheral nodes in the visualisation. We show an example of a 'hairball' in Figure 1.5.

Many studies in the last two decades, as discussed in Section 5.2 and Chapter 4, propose refinements to visualisation techniques and conduct experiments to see if these are more readable than standard node-link diagram visualisations. To be precise, typically the independent variable in such studies is visualisation technique. Sometimes a few different datasets may be used, but the data size is usually chosen in advance (e.g., through piloting), to offer a degree of complexity that allows the chosen tasks to be completed correctly in a reasonable time. As a result, this does little to address the basic gap in our understanding of node-link diagram efficacy.

We conducted an experiment where 22 participants performed a challenging—but commonly used—connectivity understanding task on force-directed, node-link visualisations for a range of data sizes and densities. In contrast to existing studies, we used the same visualisation technique throughout our study, and varied the data size and density in an attempt to establish some baseline measurements on human performance. In addition to the commonly analysed measures of speed and accuracy of task completion and subjective feedback, we also collected physiological measures known to be associated with mental effort: brain electrical activity, heart rate, and pupil size. The goal was to establish guidelines for potential limits of scalability in node-link diagram representations that may inform visualisation designers about ideal amounts of data to show and at what point it becomes necessary to limit the number of nodes and links that are displayed to the user, e.g., through selective filtering or aggregation techniques. Our data will also provide a benchmark against which other researchers can systematically compare other graph structures, different visualisation techniques, and so on.

Our results indicate that the efficiency of node-link diagrams significantly deteriorates for scale-free graphs with more than 50 nodes and a density of four. Individual analyses of accuracy, response time, subjective feedback, and EEG data agree with this finding.

However, we found that cognitive load did not uniformly increase with the number of nodes and density. Our second major contribution is to investigate the intrinsic and visual factors that affect cognitive load. The results showed a high correlation between these factors and subjective feedback and accuracy, however, there exist other variables that impact response time, pupil dilation, heart rate, and brain electrical activity, than properties of the graphs.

Our research informs the design of future network visualisation algorithms and software by clarifying the visual features that algorithms should focus on to reduce cognitive load, and in addition, the cognitive limits on the use of node-link diagrams for path following tasks. Furthermore, we have presented an experimentation model that combines both performance, self-reported and physiological factors to evaluate cognitive load, which could be applied to future user studies to evaluate other kinds of visualisation.

In the next section, we provide some definitions of terminology used in this chapter. We also discuss previous work related to this chapter. In Section 5.3 we present our user study in detail and discuss the results. In Section 5.4, we present some intrinsic and visual characteristics of the stimuli used in our study. We also show correlations between these characteristics and the results of the study. Section 5.5 presents our exploration of visualisation efficiency based on the results of our study. Section 5.6 notes some limitations to our work and describes directions for future work.

5.2 Background and Related Work

5.2.1 Cognitive Load

One of the most famous works on limits of human cognition is that of Miller [228], which suggests that humans can hold up to seven, plus or minus two, pieces of information in their short-term memory.

Cognitive Load Theory suggests that humans process information using limited working memory [294]. The theory was initially developed in the fields of education and instructional design. It suggests that cognitive load, induced by different instructional formats, should be reduced by designing instructional materials that are void of problem-solving, so that cognitive processing capacity can be devoted to knowledge acquisition.

Cognitive Load Theory was further extended to discuss three main types of cognitive load: intrinsic, extraneous, and germane. Intrinsic cognitive load is associated with the inherent difficulty of the instruction or task. Extraneous cognitive load depends on how the instruction and information are presented. While germane cognitive load refers to processing, acquiring and automating schemata [76, 295].

Three main types of measures can be used to assess cognitive load: subjective feedback, performance-based (accuracy and response time), and physiological [93]. Different measures are chosen and accepted in different fields based on their particular characteristics, such as, sensitivity to detect changes in cognitive load, task interference, operator acceptance, and implementation requirements.

Even though physiological measures have been abundantly used to measure cognitive load in systems engineering and psychology, to our knowledge, there are no studies that use physiological measures to evaluate graph visualisations. While performance-based and subjective measures are often used. Part of the contribution of this chapter is our initial exploration of the applicability of the different measures to network visualisation. For example, as discussed in Section 5.3, pupil dilation may be affected by the number of white pixels showing in the visual stimuli and, therefore, not be an accurate measure of cognitive load.

Some studies have used physiological measures to evaluate other visualisation types with respect to cognitive load.

Anderson *et al.* [28] conducted a user study comparing the cognitive load of participants when identifying the larger interquartile range on a variation of box plot types. They measured task difficulty, response time and cognitive load. They used the spectral differences in the alpha and theta frequency bands of the signals acquired by EEG as an indicator of cognitive load. The results showed a correlation between these three measures, with an increase in response time and cognitive load, as tasks became more difficult.

Peck *et al.* [251] used functional near-infrared spectroscopy to compare the cognitive load imposed by pie charts versus bar charts. They also used accuracy, response time, and subjective feedback (NASA-TLX). They asked participants to estimate differences between two highlighted sections, and given either a pie chart or a bar chart. The results do not show any difference in cognitive load between the two visualisation idioms. This is perhaps attributable to the task not really involving problem-solving, but relying mostly on visual perception.

5.2.2 Network Visualisation Readability Studies

We discussed several studies that evaluate the readability of network visualisations in Chapter 2. We have also conducted a thorough literature survey of 125 graph visualisation studies, focusing on the scale of data, which is further discussed in Chapter 4. The following is only a brief synopsis of that survey, but to our knowledge, only one study directly uses cognitive load as a measure to evaluate graph visualisations. Huang *et al.* [170] conduct a study that explores cognitive load in node-link diagrams. They propose and utilise a visualisation efficiency measure based on the approach proposed by Paas and Van Merrienboer [249], which combines mental effort and performance measures. They rely on manipulating visual, data, and task complexities, to show that cognitive load is affected by these complexities. However, the graphs they use are fairly small and they do not use physiological measures to calculate mental effort. Instead they use subjective feedback as their only measure of mental effort, but their results also confirm that subjective feedback is a good indicator of cognitive load.

Task response time is a fairly standard measure of visualisation efficiency used across many network visualisation studies. A study by Ware *et al.* [320] explores the effects of different layout features on response time in a shortest path-finding task on node-link diagrams, which they attribute to 'cognitive cost'. Their results indicate that the number of hops on the shortest path has the highest cognitive cost, followed by the number and degree of turns along the shortest paths.

Marriott *et al.* [222] conduct a study that explores the effects of layout features of node-link diagrams on memorability. The participants were asked to look at a node-link diagram for three seconds, after which they were to recall and redraw it. Memorability relies on working memory and is an aspect of cognitive load. However, the graphs were very small due to the task at hand, with only 5-6 nodes and 5-6 edges. The results of the study show that attaining certain features in the visualisation aids memorability.

There has been a lot of research in exploring the effects of layout features of node-link diagrams on readability, in particular, using shortest path finding as a task [92, 167, 174, 198, 256, 260].

Other empirical studies compare node-link diagrams with other visualisation types and techniques. For example, Ghoniem *et al.* [140] compare the effectiveness of node-link diagrams with adjacency matrices, with respect to different tasks. The study is unusual in testing relatively dense graphs, e.g. up to 100 nodes and 3,600 edges. For such graphs they found matrices provided better support than node-link diagrams for many tasks, the exception being path finding, which remains very difficult in matrices regardless of density.

Otherwise, there are many more studies evaluating different representations of network data, but they rarely significantly vary the size of the graph data, preferring one or two data sets carefully chosen through piloting, to be well within the capabilities of at least

one of the techniques being tested (e.g. [147,210,275]). We are aware of a few studies that involve large graphs (e.g. hundreds or thousands of nodes) [37, 103, 221, 230, 234, 318], but they all use interactive query or aggregation techniques, allowing the user to filter the input graph, so that only a small subset of the nodes and links are actually shown to the participant.

In Chapter 3 we point to a lack of results indicating that large network visualisations are sufficiently readable to support graph topology understanding, and to motivate the development of our *Ultra-Compact Grid Layout* algorithms that provide higher quality layout for smaller graphs. The work detailed in this chapter aims to provide some data to address this gap in our understanding of basic scalability of a standard graph visualisation technique.

5.3 User Study

We believe that there exists an inherent limit of graph sizes beyond which humans will fail to perform tasks that require a detailed understanding of part of the graph, e.g. path-finding tasks. General research on cognitive load suggests that human cognition has its limits, however, these limits are under-explored in graph visualisation.

We designed a study aiming to find a limit in graph complexity beyond which detailed tasks cannot be performed. Our study builds on the study by Huang *et al.* [170], which explored some aspects of human cognition and mental effort with respect to three different types of complexity: visual, data, and task. Our study focuses on cognitive load in node-link diagrams with respect to visual and data complexities using a much greater range of graph sizes and densities to systematically look for limits of scalability. We also explore physiological measures of EEG, pupil dilation and heart rate variability.

The task chosen for our study was to find the shortest path between two nodes in a given graph. Finding the shortest path is one of the most common tasks used in studies on node-link diagrams. Participants were shown a range of graphs with different sizes and densities arranged using force-directed network layout. They were instructed to identify the shortest path and specify the number of points on this path, if they could.

The study had 22 participants: 14 male, 8 female. 18 participants were in the age range 20–30, while four were aged 35–45. All participants had a background in Computer Science. The participants were asked about their familiarity with node-link diagrams and the shortest path problem. Nine participants frequently encountered node-link diagrams and the shortest path problem, while 13 occasionally did.

For this work we limit our search in this space to two dimensions; the number of nodes and edge density. Of course, graph complexity depends on additional variables that control the structure of the graph. To keep our study under two hours, we were forced to limit it to a single type of structure. We chose scale-free networks generated, as discussed, in the next section.

5.3.1 Graph corpus

We generated 42 graphs using code written in *JavaScript* and based on the *Barabási-Albert* [51] model. We preferred not to use standard generators, since most take—as a parameter—the total number of nodes, and the number of edges added, at each iteration. Alternatively, we wanted to specify the total number of nodes and edges. We also wanted our generated graphs to be similar to real-world graphs. The *Barabási-Albert* model is

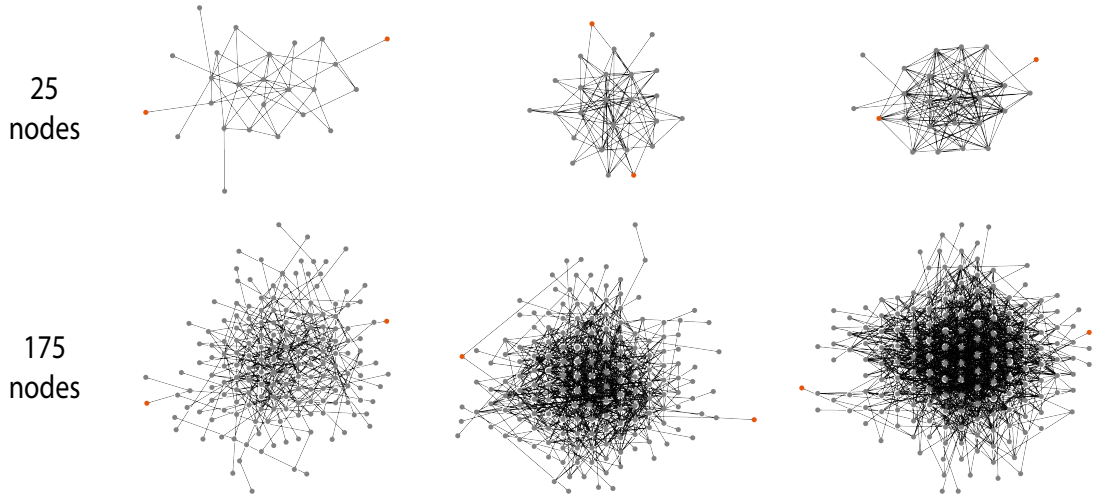


Figure 5.1: Six of the 42 stimuli used in the study. The diagrams in the top row represent the smallest networks in the corpus in terms of node count (25 nodes), while the diagrams in the bottom row represent the largest networks (175 nodes). The density of the networks increases from left to right. The diagrams were created using the force-directed constraint-based layout implemented in *WebCola* [13]. A thin white halo was added around the nodes to help identify link-node crossings against incident links on nodes.

known to produce graphs with small-world characteristics. Such graphs are common in nature and are frequently studied. For example, in cell-biology [20], bibliography [125] and the internet topology [124].

The number of nodes in the generated graphs ranged from 25 to 175 nodes in increments of 25. We experimented with different sets of number of nodes, but our pilot studies showed that, at 25 nodes, the accuracy was at a maximum, while at 175 nodes, the task was already too difficult.

To calculate the edges, we used densities of 2, 4 and 6, where

$$density = number\ of\ edges / number\ of\ nodes.$$

We chose these densities because real-world examples often have densities of less than 10 [224] and the results of our pilot studies showed that the graphs became unreadable beyond these values.

The graphs were arranged using the force-directed layout of *WebCola* [13] and saved as drawings in SVG format. We presented these drawings to the participants in a random order using a Latin square design.

5.3.2 Setup

Study set up is depicted in Figure 5.2.

The study was run on a Windows 10 Dell Latitude E7440 laptop, equipped with 2.7 GHz i7 processor and 8 GB RAM. The visual representations were displayed in a 1920×1080 pixel area on a 22-inch HP monitor. Mozilla Firefox 46.0 was used to display the visualisations and collect participant responses.

A Tobii Pro X3-120 eye tracker [11] was used throughout the study. This was directly linked to the laptop.

Figure 5.2: One of the participants during the study (face obscured for anonymity) wearing the *g.nautilus* EEG cap. The eye-tracker device is mounted under the display and the heart rate monitor is worn under clothing.

A *g.Nautilus* [4] electroencephalography (EEG) cap was also linked to the laptop to record the electrical activity of the brain via *g.Recorder*; a software provided by *g.tec* [4]. The cap exposes 32 data channels with dry electrodes spatially organised, based on the International 10-20 EEG placement system, with Modified Combinatorial Nomenclature. Additional reference and ground electrodes were attached to the back of the participants' ears. The EEG sampling was set to 250 Hz. An analogue bandpass filter was applied between 0.5 Hz and 100 Hz. A notch filter was used to suppress 48 Hz to 52 Hz power line interferences. Sensitivity was set to +/- 2250 mV.

A Polar H10 heart rate sensor was used to acquire heart rate information. This was linked to an iphone 4 via Bluetooth and *HRV Logger* [5].

5.3.3 Procedure

The participants were shown an explanatory statement and were asked to sign a consent form. They were then presented with a short tutorial explaining the concept of shortest path and the task requirements.

For each experimental task, a pair of nodes were highlighted in orange and the participants were asked to find the shortest path and take note of the number of nodes between these end nodes. The correct answers for our tasks ranged from one to six. We also allowed participants to answer with 'unsure' so that they did not need to guess.

In order to pick the end nodes in each layout, we first selected the furthest node from the centre of the diagram and then selected the nearest node to the opposite side of the vector, passing through the centre. Due to the small-world nature of the graphs and the force-directed nature of the layout, this would lead to non-trivial shortest paths.

Each participant had to perform the task 45 times, of which three were training. The stimuli were shown in a randomised order and a Latin square was used to balance their occurrences. Each task was preceded by five seconds of blank screen to serve as a rest period, which served as a baseline for the physiological measures.

After each task, the participants were asked to provide subjective feedback based on the nine-grade symmetrical category scale used by Huang *et al.* [170] and evaluated by Bratfisch *et al.* [69].

5.3.4 Results

We categorise our dependent measures into three categories: performance-based, subjective, and physiological. We analyse these with respect to our independent variables: number of nodes, number of edges, and density.

We show the results for the performance-based measures in Figures 5.3 and 5.4, the subjective ratings of difficulty in Figure 5.5, and the results of the EEG data analysis in Figure 5.6.

We show the results with respect to each stimulus. The stimuli are arranged in increasing order of node count on the horizontal axis and increasing order of density on the vertical axis.

The red ($p < .001$) and pink ($p < .05$) lines in the figures indicate statistically significant differences found using the Kruskal-Wallis test. The solid lines are transitive; i.e. they indicate significant differences between all graphs of smaller size in comparison to all larger graphs, while the dashed arcs indicate significances between the pairs connected by the arrow heads. All significant results reported have $p < .001$, unless otherwise stated.

Performance-based

In the performance-based category, we consider the accuracy and response time of the participants to complete the tasks. For accuracy, we consider 'unsure' as an incorrect answer. The response time excludes the time to submit the answers and only considers the time when the participants perform the task by attempting to find the shortest path.

Figure 5.3 shows percentages of correct, incorrect and unsure answers for each stimulus. The number of nodes and density of the stimuli are shown on the horizontal and vertical axes respectively. For graphs with 25 nodes, we found a significant drop in accuracy between densities of 2 and 6, while for graphs with 50 and 75 nodes, we found a significant drop in accuracy when using graphs with a density of more than 4.

We found that accuracy significantly deteriorates when using graphs with 100 nodes or more, compared to graphs with 25 nodes. The results also show a significant drop in accuracy for graphs with a density of 4, beyond 50 nodes. Accuracy significantly decreases for graphs with a density of 2, between graphs with 75 nodes or less, and graphs with 125 nodes. This is also evident between graphs with 50 nodes or less, and graphs with 150 nodes.

With the exception of one outlier, the accuracy of participants, when shown graphs with 125 nodes or more, is very close to the threshold that represents the probability of submitting a correct answer by random selection.

Figure 5.4 shows the response time of the participants in seconds with respect to each stimulus. The number of nodes and density are shown on the horizontal and vertical axes respectively.

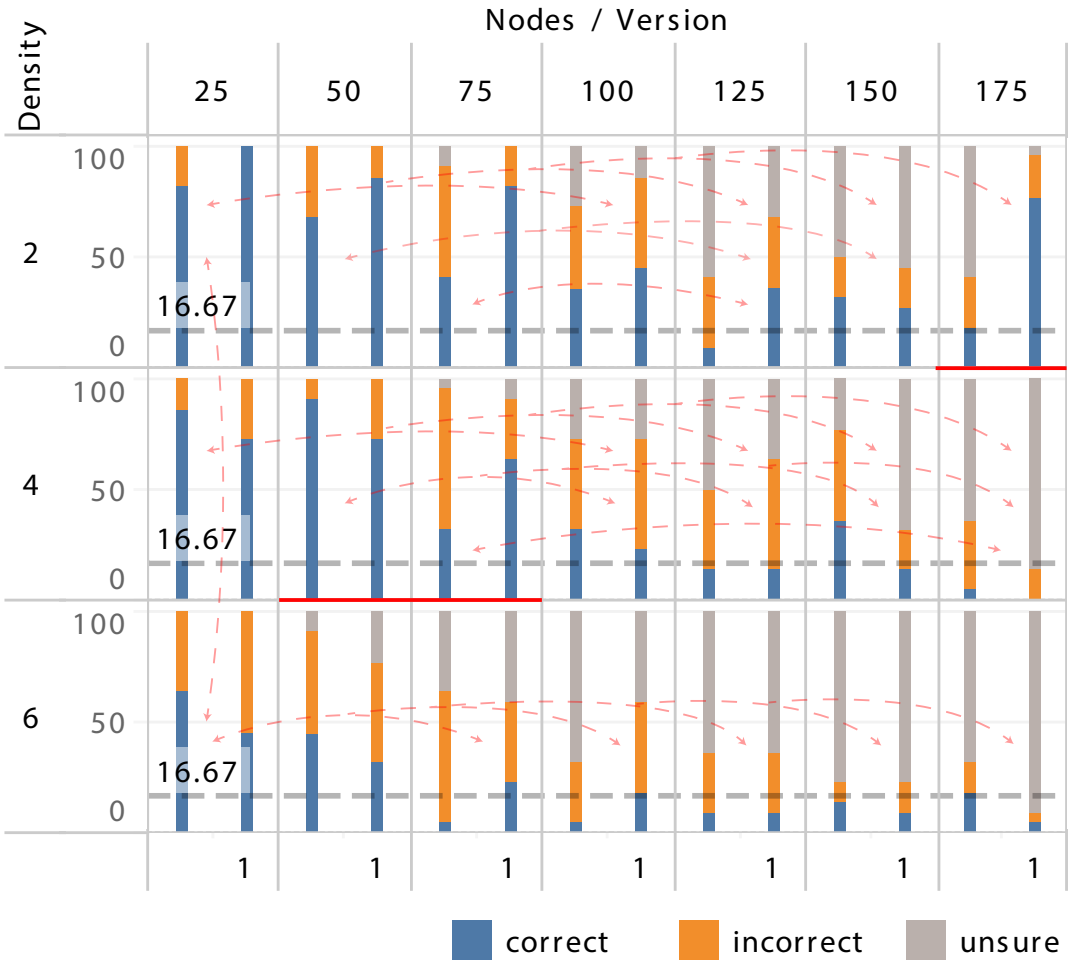


Figure 5.3: The percentage of correct answers in comparison to incorrect and unsure answers for each stimulus across the 22 participants. The stimuli are ordered in increasing number of nodes and density. The black dashed line at 16.67% indicates the percentage of correct answers that could be due to mere chance. The red lines show statistical significance ($p < .001$) between the differences of specific graph sizes. The solid lines represent transitive significance, while the dashed lines represent significance only between the arrow tips.

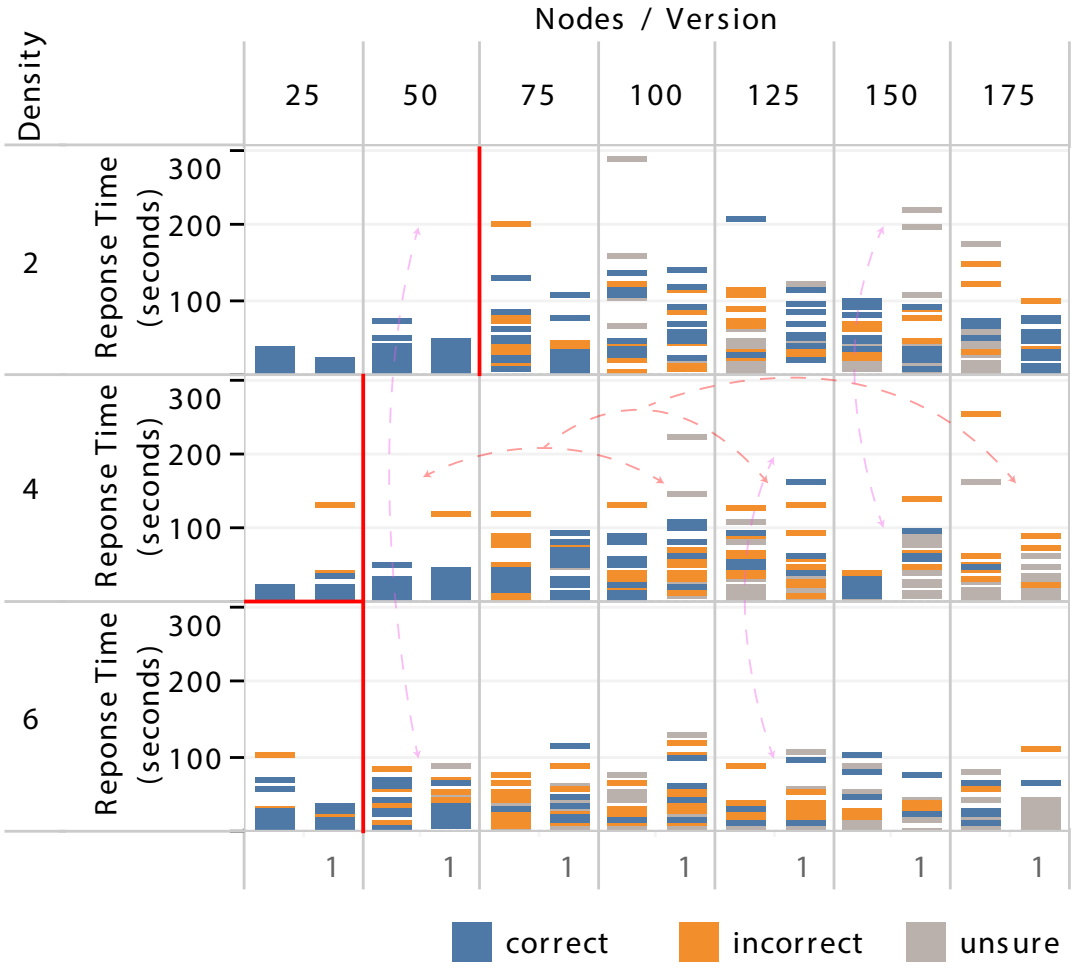


Figure 5.4: Response time with respect to each stimulus. The node count, density and version of the stimuli are also shown. Similar to the previous figure, the red lines show statistical significance ($p < .001$) between the differences of specific graph sizes, while the pink lines show a statistical significance of $p < .05$. The solid lines represent transitive significance, while the dashed lines represent significance only between the arrow tips. Even though, 'unsure' answers are included in the figure, they are excluded from the statistical analysis.

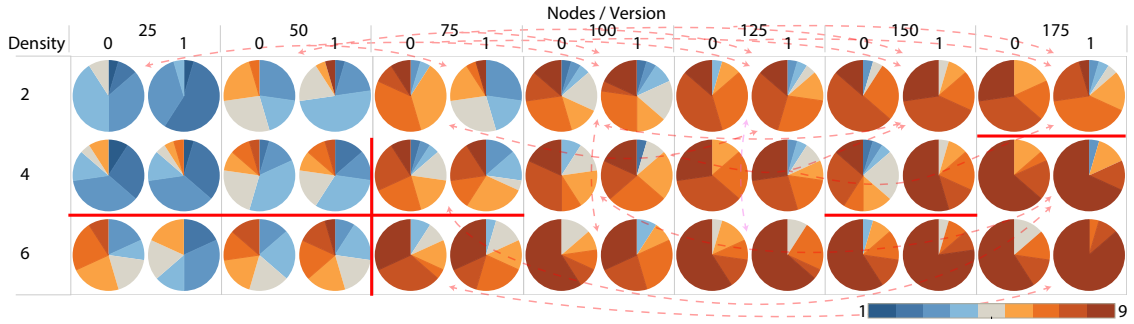


Figure 5.5: The difficulty rating shown for each stimulus. The stimuli are ordered by increasing node count on the horizontal axis and increasing density on the vertical axis. The lines indicate statistical significance ($p < .001$) between the ratings.

Our results indicate that response time significantly increases beyond 50 nodes when using a density of 2 and beyond 25 nodes when using a density of 4 or 6. They also indicate a significant increase in response time beyond a density of 4 for graphs with 25 nodes.

Subjective

The participants were asked to rate the difficulty of each trial. They were given a nine-grade symmetrical category scale used by Huang *et al.* [170] and validated by Bratfisch *et al.* [69]. The scale uses the following terms: 'very very easy', 'very easy', 'easy', 'rather easy', 'neither easy nor difficult', 'rather difficult', 'difficult', 'very difficult', 'very very difficult'.

Figure 5.5 shows the rating of each stimulus. The number of nodes and density are shown on the horizontal and vertical axes respectively.

Our results show that for graphs with a density of 2, the participants rated graphs with 25 nodes as significantly more difficult than graphs with 75 nodes or more. They also rated graphs with a density of 2 and 50 nodes as significantly more difficult than graphs with 100 nodes or more. For denser graphs with densities of 4 and 6, the difficulty ratings were significantly higher for graphs with 50 nodes or less, compared to graphs with 75 nodes or more.

The participants found density 6 graphs to be more difficult than graphs with densities of 2 or 4 for graphs with 75 nodes or less. The results also show that graphs with 175 nodes and a density of 2 were significantly more difficult than graphs with 175 nodes and densities of 4 or 6.

Physiological

Pupil Dilation: We used Tobii Studio to record the eye tracker data from the Tobii Pro X3-120. We used the average of the two eyes in order to reduce noise. In cases where we had pupil size information of one eye, we used that alone.

For each task, we used the five seconds pre-task resting period to extract an average baseline, then we calculated pupil dilation by subtracting the average pupil size during the inter-trial rest period from the peak pupil size during task performance. We used peak dilation instead of mean pupil dilation, since the latter does not work well with tasks that vary in length across participants [58].

We did not find any significant differences in peak dilation between different graph sizes.

Heart Rate Variability: We recorded the beats per minute (bpm) and r-r interval for each participant using a polar H10 heart rate monitor.

We used rmssd and pnn50, which are common measures for heart rate variability analysis, in order to analyse our results. However, we did not find any significant differences between the trials.

Electrical Activity of the Brain: We used EEGlab [96] to process and analyse the EEG data exported from g.Recorder.

Cognitive and memory performance are identified to be reflected within the alpha (8 - 12 Hz) and theta (4 - 8 Hz) frequency bands [196]. Thus, we filtered the EEG data by applying a low-pass filter of 15 Hz and a high-pass filter of 1 Hz. This filter would also get rid of possible noise and muscle artefacts [87].

We also manually checked for bad channels and tasks. We fixed bad channels by filtering them out and recreating data using interpolation, whereas, we filtered out the bad tasks completely. We also removed the total EEG data of one participant. Overall, we found 21 bad channels and 26 bad tasks across the remaining 21 participants.

The acquired EEG data is a two-dimensional table. One dimension represents time, while the other represents the electrodes. We cut this continuous data into segments, in order to analyse changes in the EEG data with respect to our tasks.

We considered frequencies of 3, 4, 5, 6, 7 Hz for the theta band and 8, 9, 10, 11, 12 Hz for the alpha band. We used the Fast Fourier transform function in *MATLAB* [9] to extract the power of these frequencies during the period of each task. We then measured the difference between the mean power of the alpha frequencies and the mean power of the theta frequencies. We call this measure 'EEG cognitive load'.

After a visual check on the results, we realised that the 'unsure' answers were adding a lot of noise. Since we do not have a way to differentiate 'unsure' answers that were the result of giving up too early, we filter out all the 'unsure' answers.

Figure 5.6 shows the results of the cognitive load measured using the EEG data for each visual stimulus. The different node counts are shown using the columns, while the three different densities are shown using the three rows.

The results show a significant increase in EEG cognitive load for graphs with 25 nodes or more, for all densities. The results also show a significant increase in EEG cognitive load beyond 50, and 75 nodes for graphs with a density of 2. For graphs with these node counts; (25, 50 and 75), there is also a significant increase in EEG cognitive load beyond a density of 4.

5.3.5 Discussion

Accuracy, response time, subjective feedback and electrical activity of the brain show that the task of path finding between two nodes becomes significantly difficult beyond a density of 4. They also show a cut-off at 50 nodes. In terms of accuracy, participants were as accurate as random selection beyond graphs with 100 nodes. Response time and electrical activity of the brain push this cut-off even further down to 25 nodes.

Even though we tried to constrain the visual complexity of our stimuli by limiting ourselves to the use of node-link diagrams to visualise the study corpus, we found some changes in the results that were not explained by the data complexity (node count and density). For example, the second graph (version 1) with 175 nodes and a density of 2,

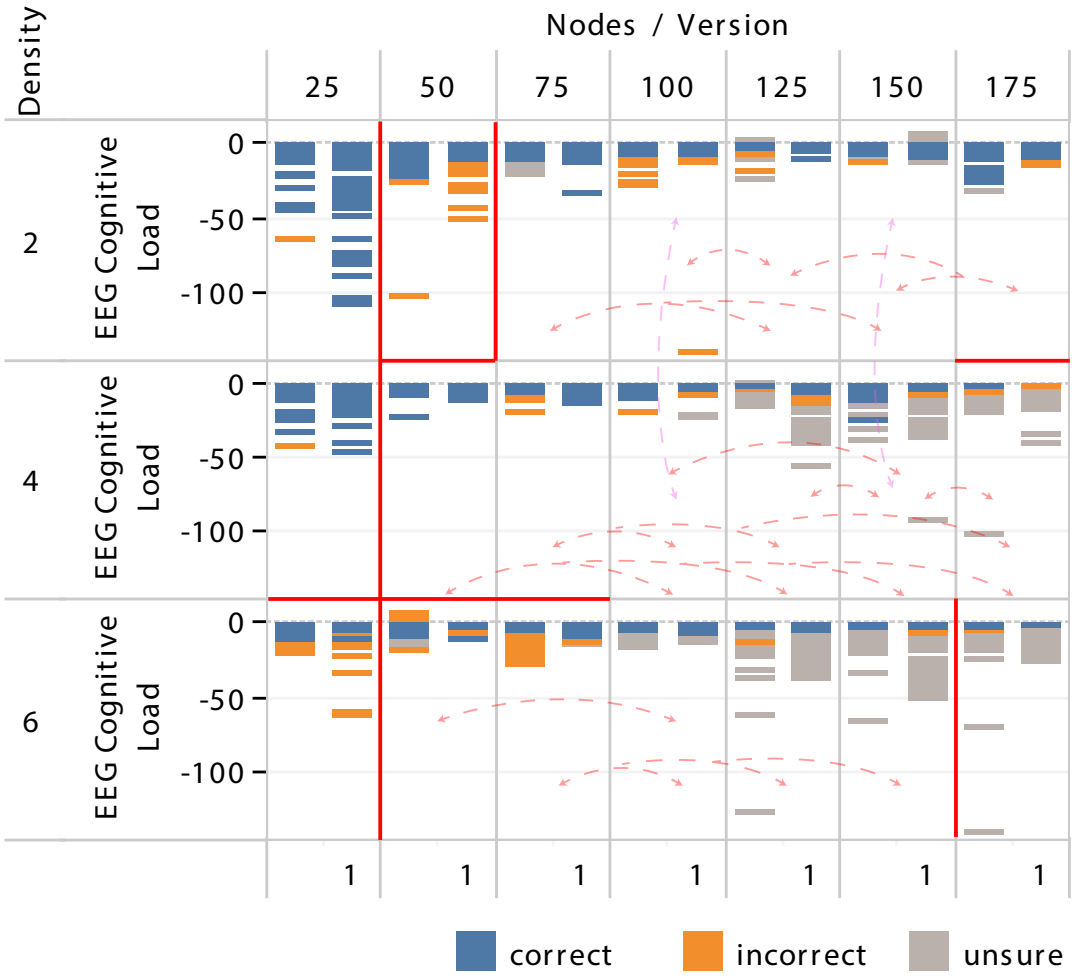


Figure 5.6: EEG cognitive load with respect to each stimulus. Number of nodes and density are shown on the horizontal and vertical axes respectively. Even though, the 'unsure' answers are shown in the figure, they are filtered out in the statistical analysis. The red lines represent statistical significance of $p < .001$ between the differences in cognitive load, while the pink lines have a lower significance of $p < .05$. The solid lines have a transitive nature, while the dashed lines are specific to the sections they point to.

stands out as an outlier in Figure 5.3 with an average accuracy of 77.27%; its counterpart (version 0) has an average accuracy of 18.18%. After investigating the layouts of these two graphs, we found obvious differences in the visual complexity, which could affect cognitive load. We discuss our exploration of these factors in the next section.

5.4 Factors Affecting Cognitive Load

In the previous section, we analyse the results of our study with respect to the number of nodes and the density of each of the graphs. However, we believe that these are just coarse measures of complexity, and that there also exists other factors that affect cognitive load.

There has been a lot of research to understand how different layout features affect the readability of node-link diagrams. We discussed these works in more detail in Section 2.

5.4.1 Graph Metrics

We considered two types of metrics: intrinsic and visual. The intrinsic metrics are related to the data, while the visual metrics are layout features acquired due to the visual representation.

In addition to the number of nodes and density, we considered other intrinsic metrics. For each of the 42 graphs in our corpus, we measured the number of all possible shortest paths using breadth-first search, the number of intermediate nodes on the shortest path, and the average degree of nodes on the shortest paths.

For our visual metrics, we counted the number of link-link crossings, the number of node-link crossings, the average number of link-link crossings on the shortest paths, the average number of node-link crossings on the shortest paths, the average penalty for small angles between crossings on the shortest paths, the average turning angle on the shortest paths, the average distance of nodes on the shortest paths to the Geodesic path (Geodesic path deviation), and the average Euclidean distance of the shortest paths.

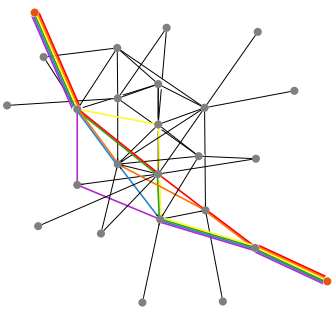
The penalty for small angles was calculated by measuring the angle between two line crossings and subtracting it from 90 degrees. This would apply a high penalty for small angles, while adding small penalties for angles closer to 90 degrees.

Figure 5.7 shows examples where some of these factors are highlighted. The graph shown in the examples is from our study corpus. It has 6 possible shortest paths (Fig. 5.7(a)), each with 4 intermediate nodes. The nodes on the shortest paths have an accumulated degree of 56 (Fig. 5.7(h), average = 28.33/path). The node-link diagram representing the graph has 57 link-link crossings (Fig. 5.7(b)) and 6 node-link crossings (Fig. 5.7(c)). There are 33 link-link crossings (Fig. 5.7(d), average = 8.33/path) and 1 node-link crossing (Fig. 5.7(e), average = 0.17/path) on the shortest paths. The sum of the Euclidean distance of the links on the shortest paths is 2803.66 (Fig. 5.7(f), average = 1067.47/path). The sum of the distance between the nodes on the shortest paths from the Geodesic path is 874.46 (Fig. 5.7(g), average = 415.66/path). The sum of the penalty for small angles of line-line crossings on the shortest paths is 925.92 (Fig. 5.7(i), average = 217.08/path). Lastly, the sum of the turning angles on the shortest paths is 426.72 degrees (Fig. 5.7(j), average = 105.58 degrees).

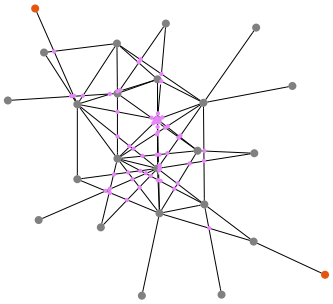
Figure 5.9 shows some intrinsic and visual characteristics of all the graphs used in our study. The metric at 0 degrees clockwise is the number of intermediate nodes on the shortest path. 60 degrees clockwise represents the penalty for small crossing angles. At 120 degrees clockwise we show the number of link-link crossings, while at 240 degrees clockwise, we show the number of node-link crossings. At 240 degrees clockwise, we show the number of possible shortest paths.

As expected, our analysis shows that the number of hops decreases with the increase in density. It also shows that the number of crossings increases with the increase in node count and density.

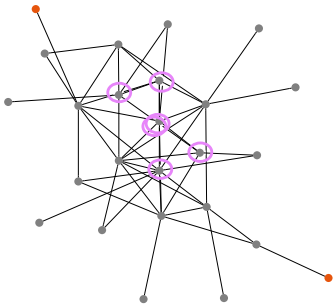
Similarly, Figure 5.8 shows some metrics that are associated to the shortest paths of the graphs used in our user study. The metric at 0 degrees clockwise is the average penalty for small angles between crossings on the shortest paths. The average degree of nodes on the shortest paths is shown at 60 degrees clockwise. The average Geodesic path deviation is shown at 120 degrees clockwise. The average Euclidean distance of the shortest paths is represented at 180 degrees. The metric shown at 210 degrees clockwise is the average count of link-link crossings on the shortest paths. Finally, the point at 240 degrees clockwise represents the average turning angle on the shortest paths.



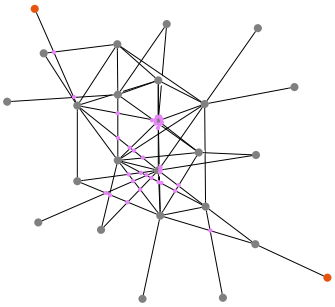
(a) Shortest paths



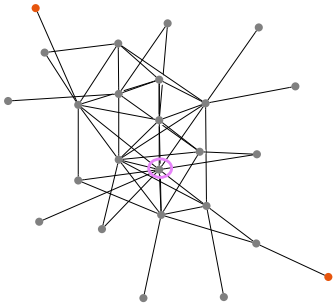
(b) Link-link crossings



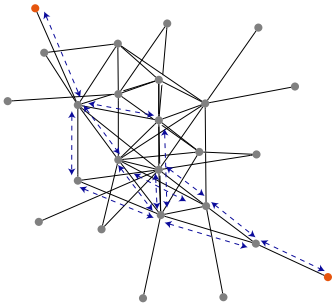
(c) Node-link crossings



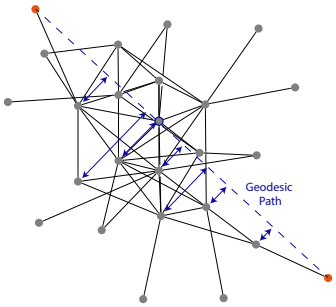
(d) Link-link crossings on the shortest paths



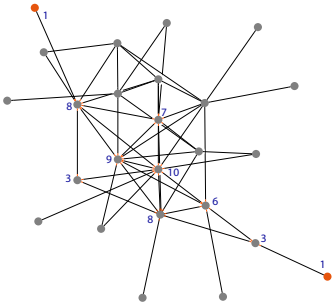
(e) Node-link crossings on the shortest paths



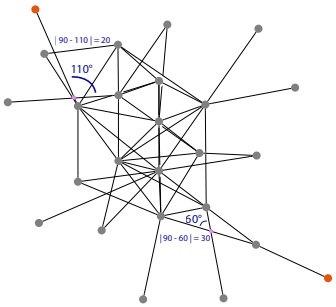
(f) Euclidean distance of the shortest paths



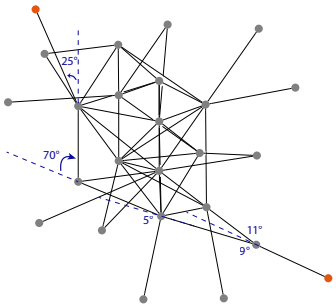
(g) Geodesic path deviation



(h) Degrees of nodes on the shortest paths



(i) Crossing angles on the shortest paths



(j) Turning angles on the shortest paths

Figure 5.7: Intrinsic and visual metrics for a sample graph used in the study with 25 nodes and 50 edges (density 2, version 0).

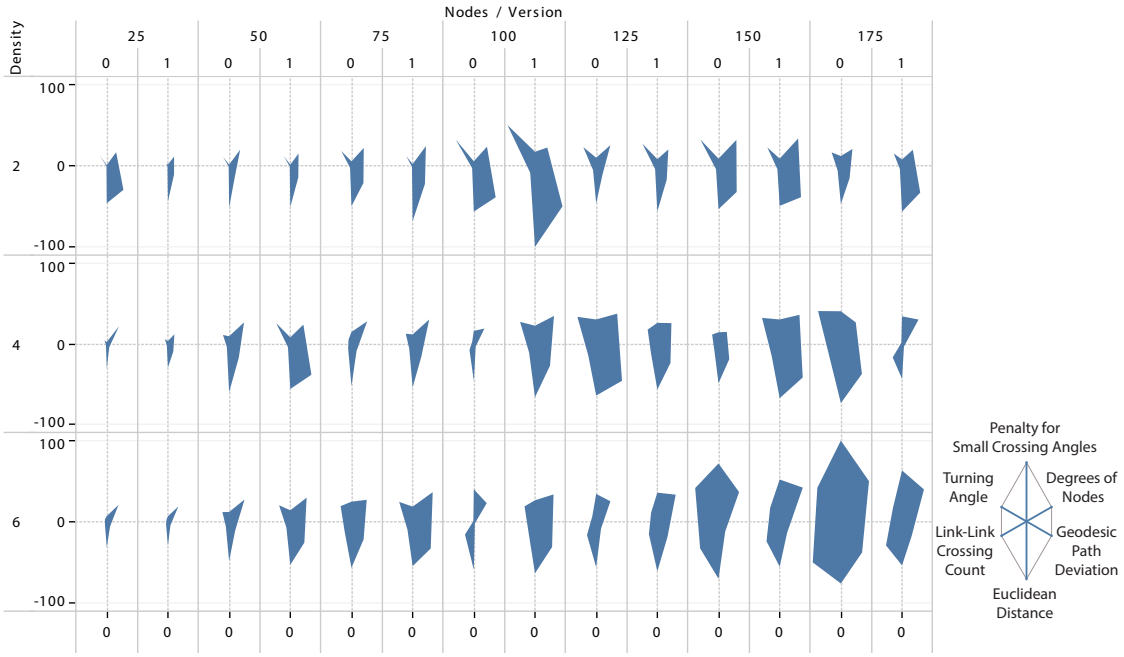


Figure 5.8: Intrinsic and visual characteristics of the 42 graphs used in the user study of Section 5.3.

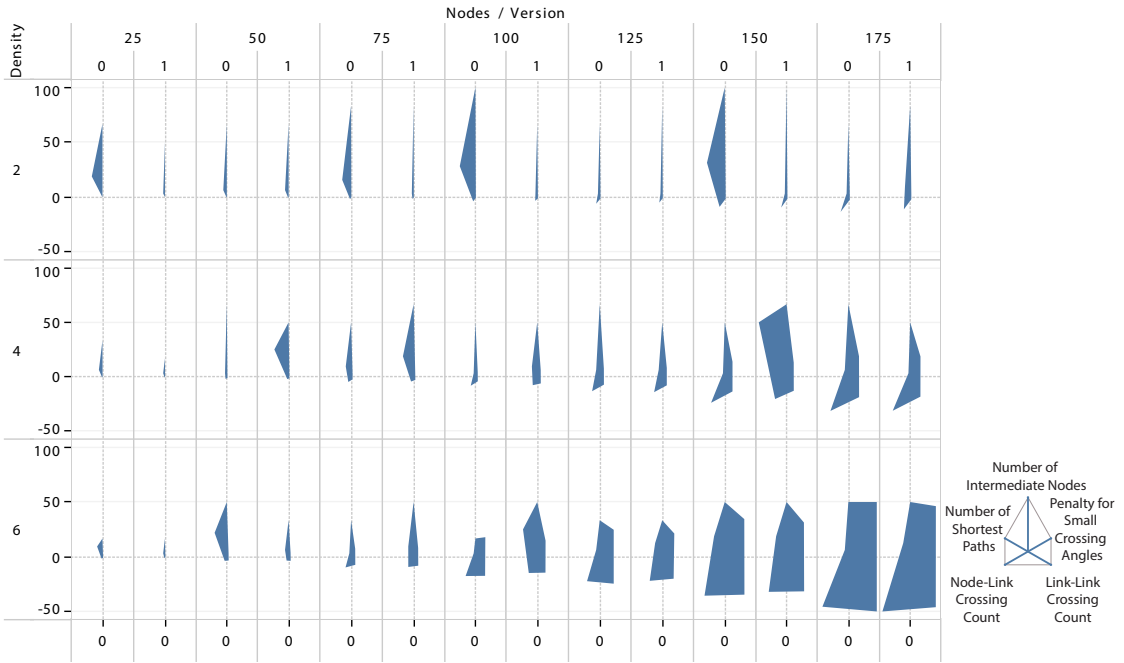


Figure 5.9: Intrinsic and visual characteristics related to the shortest paths of the 42 graphs used in the user study of Section 5.3.

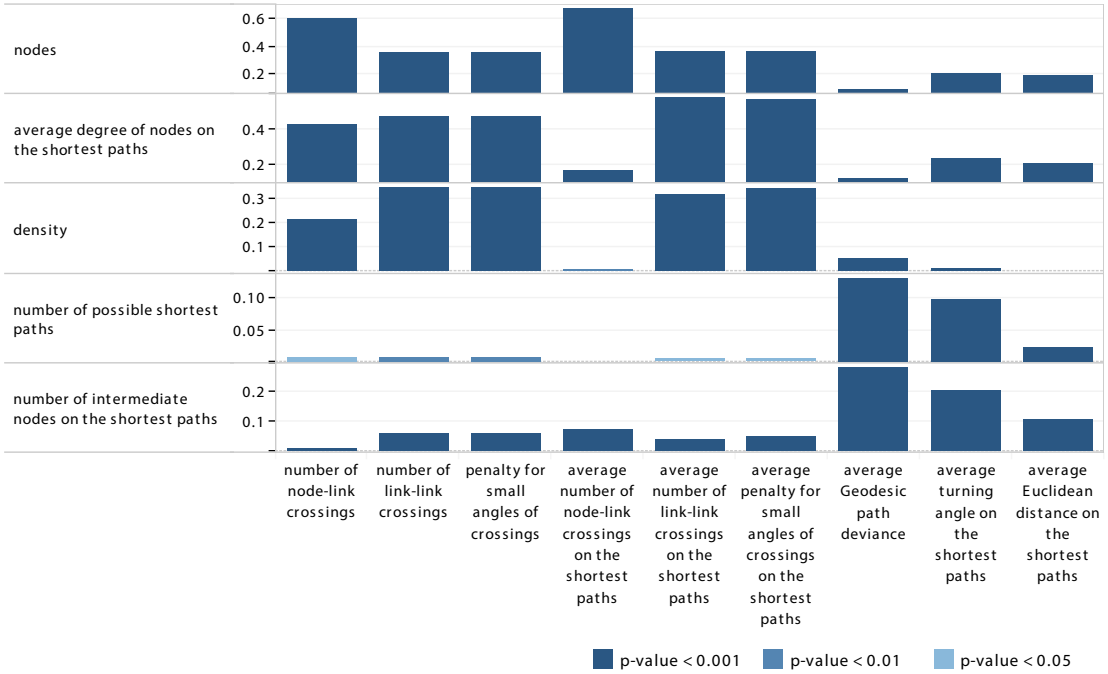


Figure 5.10: The linear correlation (adjusted R-squared) between different graph characteristics. The visual factors are represented by the columns, while the intrinsic factors are represented by the rows. The y-axes have different scales.

An obvious pattern is the increase in node degrees on the shortest paths as the node count and density increase. Moreover, to fully explore these dependencies, we retrieved the independent linear correlations of each factor with respect to all other factors.

Figure 5.10 shows how changing one factor affects other factors. It is evident that an increase in the average degree of nodes on the shortest paths, the number of nodes, and density, greatly affect the number of link-link crossings and—naturally—the penalty for small angles of crossings.

Similarly, the number of nodes and density are highly correlated, with the average number of link-link crossings on the shortest paths and the average penalty for small angles on the shortest paths.

It is interesting that we did not find a correlation between the average degree of nodes on the shortest paths with the average number of link-link crossings on the shortest paths.

The correlations of the number of nodes with visual factors are very similar to that of density. However, the average turning angle on the shortest paths and the average Euclidean distance on the shortest paths show a higher correlation with the number of nodes than with density, while the number of node-link crossings shows a higher correlation with density than the number of nodes.

The correlations of the average degrees of nodes on the shortest paths are similar to the combined correlations of the number of nodes and density, with the exceptions of the average link-link crossings on the shortest paths and the average penalty for small angles on the shortest paths.

The correlations of the number of possible shortest paths and, the number of intermediate nodes on the shortest paths, are highly similar. They correlate the most with the average Geodesic path deviance and the average turning angle on the shortest paths.

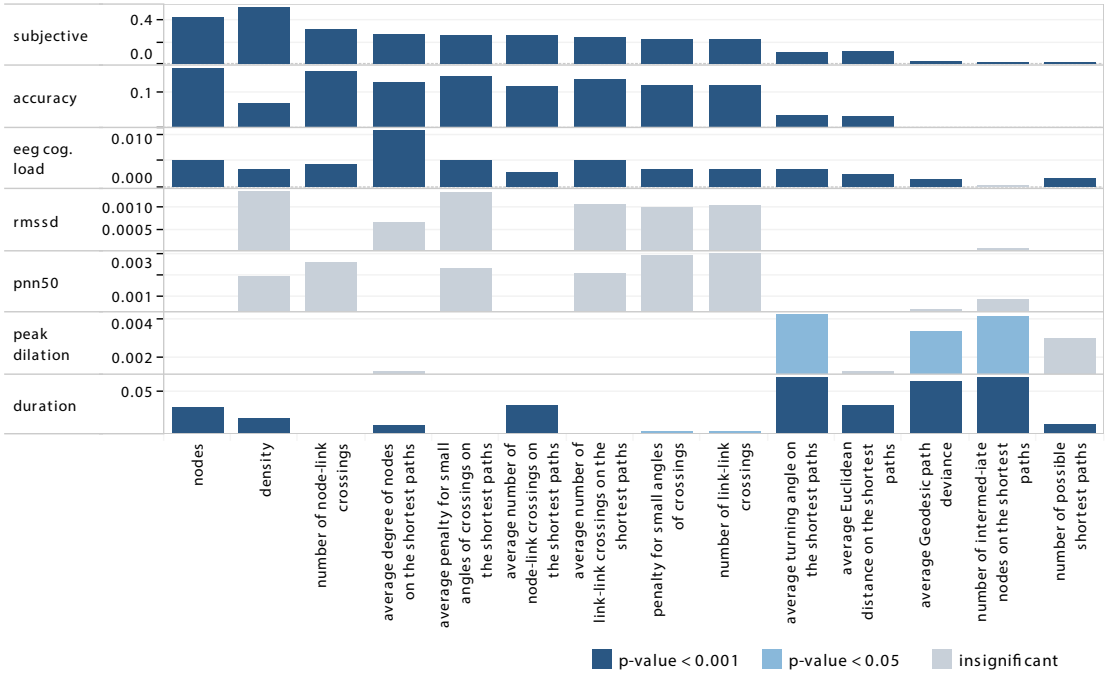


Figure 5.11: The linear correlation (adjusted R-squared) of intrinsic factors with visual factors of the graphs used as stimuli and the results of our user study. The y-axes have different scales.

5.4.2 Correlations

After identifying the features of our graphs and exploring the correlations of intrinsic factors with visual factors, we explore possible correlations of both, intrinsic and visual factors, with the results of our user study. In other words, in this section, we analyse the dependent variables of our study with respect to the independent variables. The independent variables being the intrinsic and visual factors mentioned in the previous section.

Figure 5.11 shows the correlations of intrinsic factors with visual factors (independent variables), and resulting measures from our study (dependent variables). Two distinct patterns are visible.

Subjective feedback, accuracy and cognitive load, measured using the EEG, have similar correlations, where most factors have an effect. Density has the highest impact on subjective feedback, while the number of nodes has the highest impact on accuracy, and the average degree of nodes has the highest influence on cognitive load measured, using the EEG.

Pupil peak dilation and response time also have similar correlations. However, they are mostly affected by the steep turning angles on the shortest paths, the Euclidean distance of the shortest paths, and the number of intermediate nodes on the shortest paths.

We did not find any significant correlations between the factors and the heart rate variability measures. Steep turns on the shortest paths are correlated with all the other measures.

There seems to be other factors that affect pupil dilation and cognitive load measured using the EEG. Similarly, but to a lesser extent, response time is affected by other unknown variables. Changes in graph characteristics are highly correlated with the subjective feedback of participants. They also show a good correlation with accuracy.

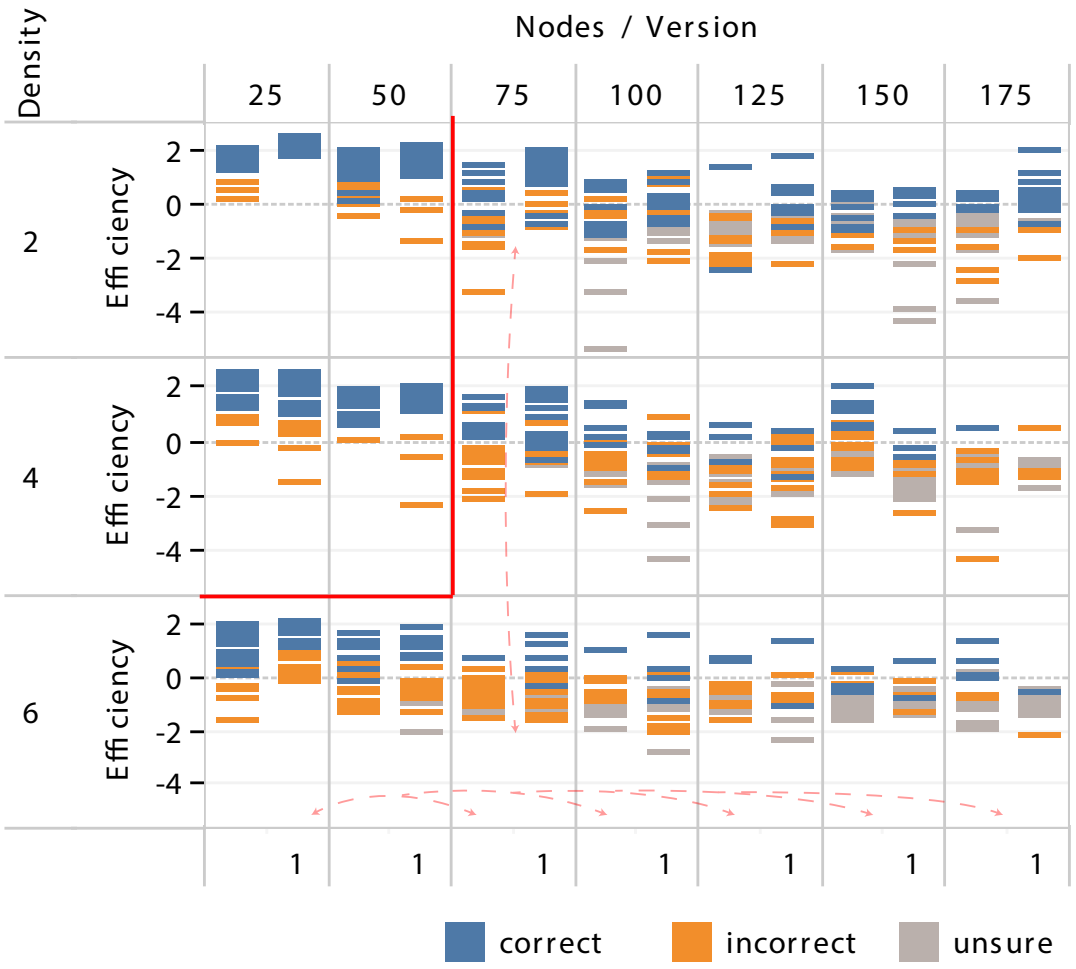


Figure 5.12: The efficiency of finding a shortest path on node-link diagrams of graphs with different node counts and densities. The node count is represented by positions on the horizontal axis, while the densities are shown on the vertical axis. The red lines indicate significant differences ($p < .001$). The solid lines represent transitive differences, while the dashed lines are restricted to the sections they point to.

5.5 Visualisation Efficiency

Paas *et al.* [249] propose a computational method to measure instructional efficiency in relation to cognitive load. Huang *et al.* [170] extend this method to measure visualisation efficiency. They define efficiency as the difference between standardised accuracy, response time, and mental effort. They explain that efficiency increases when accuracy is high, difficulty is low and response time is short. For their study, they use subjective feedback as a measure of mental effort.

5.5.1 Efficiency Using Subjective Feedback

Similar to the study by Huang *et al.* [170], we use the subjective rating of difficulty as mental effort and calculate visualisation efficiency for each of the stimuli used in our study.

We show the results in Figure 5.12. The node count and density of each graph are shown using the horizontal and vertical axes respectively. The red lines in the figure indicate statistically significant differences found using the Kruskal-Wallis test ($p < .001$). The solid lines are transitive; i.e. they indicate significant differences between all graphs of smaller size in comparison to all larger graphs. The dashed arcs indicate significances between the arrow tips.

The results show that for all graphs with 75 nodes or more, except for those with a density of 6, efficiency was significantly less than the ones with 50 nodes or less. For density 6, graphs with 75 nodes or more were significantly less efficient than those with 25 nodes.

The results also show that even for small graphs with 50 nodes or less, efficiency significantly drops beyond a density of 4. Also for graphs with 75 nodes, efficiency significantly dropped for density 6 compared to density 2.

This confirms our previous finding that shortest path finding becomes significantly more difficult beyond 50 nodes and a density of 4.

5.5.2 Efficiency Using EEG Data

Subjective feedback is one of the most commonly used measures in empirical studies [248]. It is inexpensive, does not interfere with the primary task of the study, and is well validated. The results of our study also showed a high correlation of subjective feedback with the intrinsic and visual factors of the graphs, as discussed in Section 5.4. However, due to recent innovations in wearable sensors, physiological measures are gaining popularity. We swapped subjective feedback as the measure for mental effort in the efficiency calculation, with the EEG cognitive load measure (discussed in Section 5.3). We name this, 'efficiency 2.0', to avoid confusion.

Figure 5.13 shows the results of the modified efficiency calculation for each stimulus used in our study. The node count and density of each graph are shown using the horizontal and vertical axes respectively. The results show that for all graphs with 75 nodes or more, except for those with a density of 6, efficiency was significantly lower than for graphs with 50 nodes or less. Similarly, for all graphs with 50 nodes or more, including ones with density 6, efficiency was significantly lower than those with 25 nodes. Also, for graphs with 75 nodes or less, efficiency dropped significantly when the graph density increased.

The modified measure is more sensitive to changes in cognitive load. However, both measures reflect the results of the individual measures and confirm the cut-off at 50 nodes, and a 4 in density. Choosing one over the other depends on the evaluation needs of individual studies. Electroencephalograms are still rather expensive and some of our participants expressed discomfort when wearing the EEG cap.

5.6 Limitations and Future Work

An obvious and intentional limitation of the user study, discussed in Section 5.3, is the use of one visual representation - specifically node-link diagrams. We aimed to minimise the effects of visual complexity on cognitive load, in order to emphasise the effects of data complexity. Future research can do the opposite to explore the effects of visual complexity on cognitive load. Results attained from different visual representations can also be compared to our data relative to node-link diagrams.

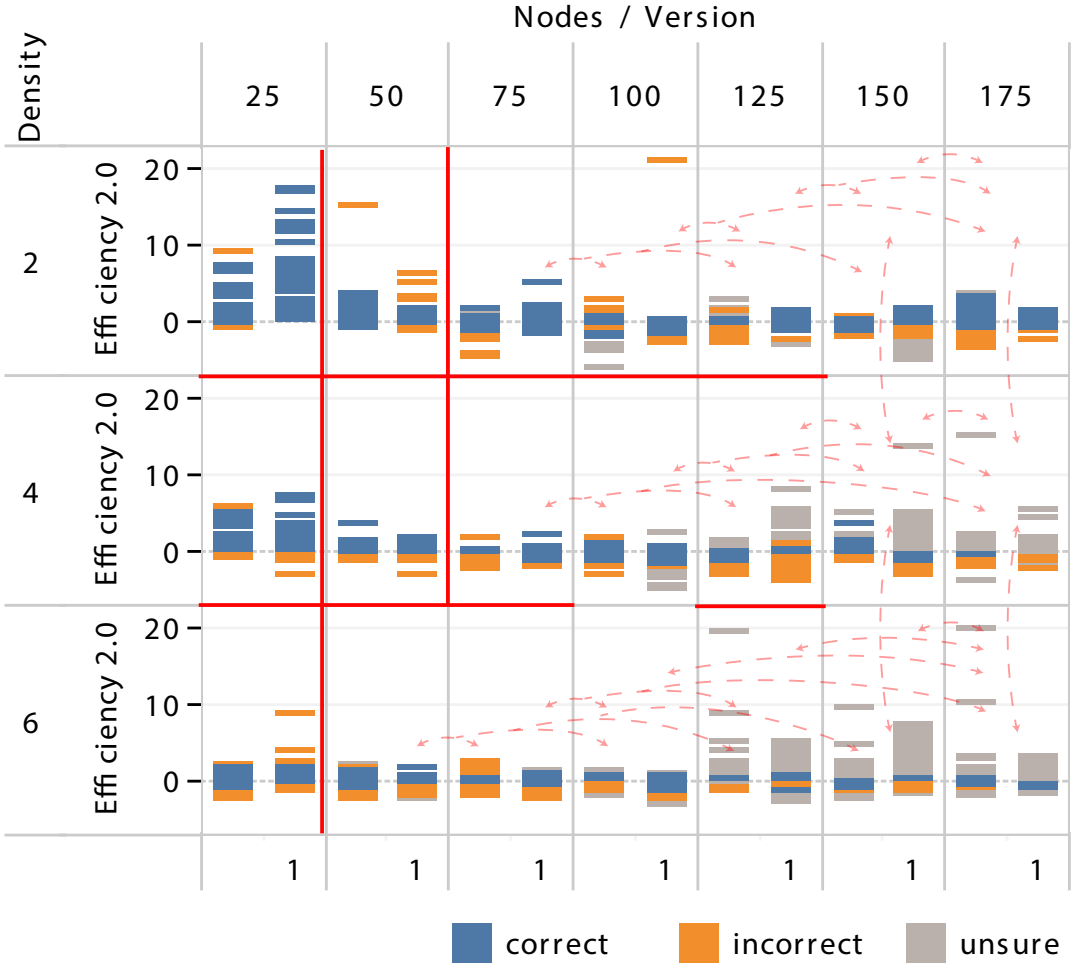


Figure 5.13: The modified efficiency of finding a shortest path on node-link diagrams of graphs with different node counts and densities. The node counts are represented by positions on the horizontal axis, while the densities are shown on the vertical axis. The red lines indicate significant differences ($p < .001$). The solid lines represent transitive differences, while each dashed line is restricted to a pair.

We use scale-free graphs, but in future work, we intend to use other types of network data. We would like to explore how cognitive load changes for sparser graphs. Our experimental model can be applied to explore different complexity domains.

As future work, we would also like to explore other aspects of eye data. In addition to eye pupil diameter, we recorded eye movement data. We would like to explore the correlation between total eye movement and cognitive load. Perhaps the lack of significant findings with respect to pupil dilation was due to illumination effects. Pupil diameter is sensitive to light [58] and with denser graphs the screen would get darker.

One of the limitations of our study was that we used one task: finding a shortest path between two nodes. Nonetheless, we believe that this complex task is representative of a fundamental class of tasks and involves other 'sub-tasks', such as disambiguating edges, inspecting neighbours, remembering previously inspected nodes, browsing through paths, and so on. As future work, it would be possible to perform studies with other detail-oriented tasks and compare the results with ours. These would lead to the understanding of the effects of task complexity on cognitive load.

In continuation of the work discussed in Section 5.4, we intend to develop a thorough cognitive model that would include different factors of graph complexity.

5.7 Conclusion

In this chapter, we explore the effects of graph size (number of nodes and density) on cognitive load. We limit our exploration to one task: finding a shortest path between two nodes. Our results show that all measures of cognitive load indicate heavy load beyond graphs with more than 50 nodes. They also show that cognitive load increases significantly for graphs with a density higher than four.

We use one standard visualisation technique (force-directed node-link diagrams) to show the graphs to the participants. Thus minimising changes in visual complexity. Nonetheless, due to dependencies between graph structure and layout, we end up having different layout features between the stimuli. We explore the correlation of these visual factors with the intrinsic factors belonging to the data in Section 5.4. We also explore the correlations of these factors with the dependent variables of the user study.

Our results confirm the usefulness of our high-quality *Ultra-Compact Grid Layout*, which we presented in Chapter 3. We were initially worried that our method was not able to handle large networks, but now realise that diagrams of large networks are not suited for tasks that require a detailed understanding of the network. The work in this chapter can guide visualisation designers when creating visualisations that must scale to larger graph data (e.g. setting limits on neighbourhood size in overview-and-detail techniques using node-link diagrams for detail). We also hope this work stimulates development of new techniques that demonstrably scale to more complex networks, such as our summary representation discussed in the next chapter.

Chapter 6

Graph Thumbnails: Making Sense of Very Large Networks

"but now, from way up here,
it's crystal clear"

Sir Timothy Miles Bindon Rice,
A Whole New World

A song from Disney's animated film
Aladdin.

In this chapter we propose *Graph Thumbnails*, small icon-like visualisations of the high-level structure of network data. *Graph Thumbnails* are designed to be legible in small multiples to support rapid browsing within large graph corpora. Compared to existing graph-visualisation techniques our representation has several advantages: (1) the visualisation can be computed in linear time; (2) it is canonical in the sense that isomorphic graphs always have identical thumbnails; and (3) it provides precise information about the graph structure. We report the results of two user studies. The first study compares Graph Thumbnails to node-link and matrix views for identifying similar graphs. The second study investigates the comprehensibility of the different representations. We demonstrate the usefulness of this representation for summarising the evolution of protein-protein interaction networks across a range of species.

The work presented in this chapter was accomplished in collaboration with my supervisors Tim Dwyer, Karsten Klein, Kim Marriott, and Michael Wybrow. An article describing the work was published in *Transactions on Computer Graphics and Visualization*.

6.1 Introduction

Modern graphics computing power makes it possible and tempting to create visualisations that render individual marks for each node and link in large network data. For example, node-link diagrams are typically drawn with a circle or rectangle mark for each node—possibly with a label—and a line for each link. Further, it is a triumph of algorithm engineering that we now have reasonably efficient methods for computing unfolded

layouts of diagrams with tens or hundreds of thousands of nodes in seconds or minutes (e.g. [39, 165]). Such algorithms work quite well for sparse tree-like or mesh structures, but on networks that have a scale-free or small-world property the node-link diagrams cannot be untangled sufficiently to understand the connectivity—the infamous "hair-ball" problem (e.g. Figure 6.1(a)). Adjacency matrices are an alternative representation that sidesteps the problem of tangled edges by giving each edge a mark in its own matrix cell. However, matrices become unreadable when there are large numbers of nodes, as the rows and columns become too narrow (e.g. Figure 6.1(b)).

In addition to efficient methods that are able to produce diagrams for large networks, we propose a high-quality layout for small to medium networks, which is presented and discussed in Chapter 3. However, not every visualisation task requires a level of detail where every graph element is visible. Sometimes we just want an overview of the large-scale structure of the network. Some applications call for a quick comparative view of a large number of networks, for example, when comparing the structure of networks from different origins, or when trying to understand the evolution of a dynamically changing network. The results of the user study presented in Chapter 5, show that path finding tasks become significantly challenging beyond networks with more than 50 nodes and densities of four.

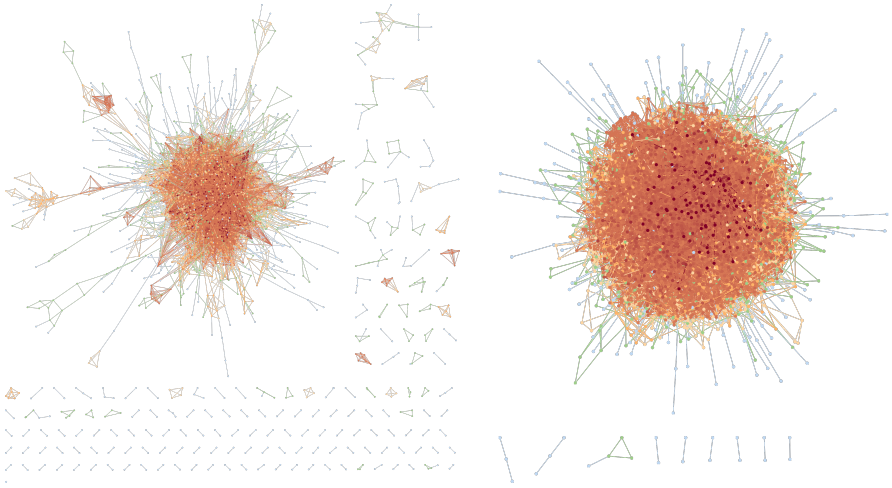
Another inspiration for the technique presented in this chapter is that force-directed layouts of large and reasonably dense networks tend to look like vaguely circular blobs. Maybe it is possible to see that the graph has several interlinked blobs. Maybe some denser cores are visible within the blobs. But, broadly speaking, if that is all you see after the effort of running a large-scale physics simulation to untangle the network, can we not find a more efficient way to decompose the network into some hierarchical structure and then visualise this hierarchy directly?

Most current decompositions and aggregations proposed for large graph analysis are challenging both in their computational complexity and interpretability, as—similar to the network layout algorithms mentioned above—they are typically intended to allow a detailed analysis of a single network. We are exploring how far we can simplify computation and visualisation while still retaining characteristic structural features that allow us to clearly distinguish different networks over a large set, or to detect high-level similarities.

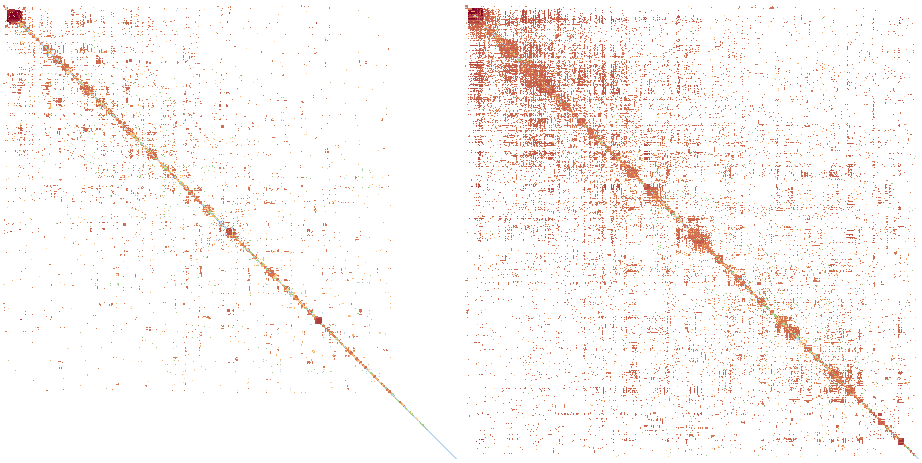
The *Graph Thumbnail* representation explored in this chapter offers quick comparison and intuitive representation. It maintains structural information and hierarchy and can act as an overview for complex and large networks. A typical application—as we will see in Section 6.7—would be to browse a set of large networks in order to identify patterns across networks, or to detect outlier networks. Another obvious use for thumbnail representations of graphs is the same way thumbnail representations are used in file explorers or menu exploration: as an iconic view of a graph file that can easily be found amongst a large collection.

Thus, to summarise, the visualisation tasks we consider are:

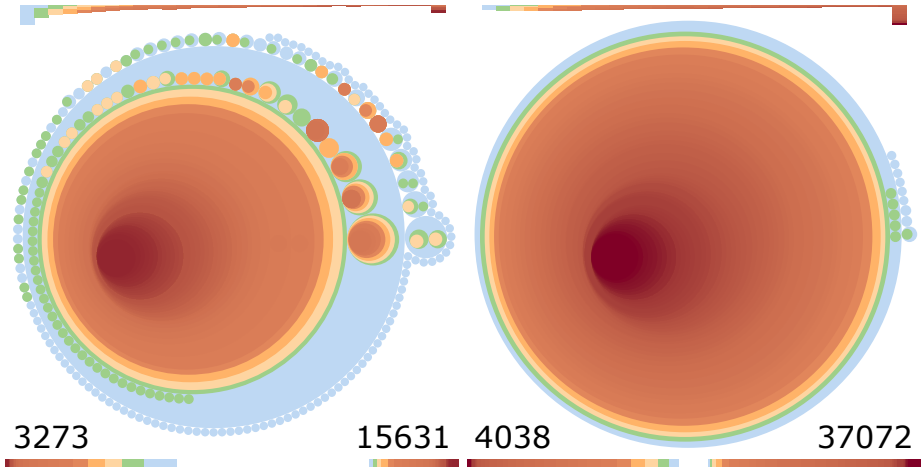
- **Identification**: Quick identification of a network from a collection, e.g. browsing through a directory full of graph files.
- **Comparison**: Finding similarly structured networks from a collection, or alternatively, finding outliers (i.e. dissimilar networks).
- **Overview**: Quickly ascertain key structural information about the overall network structure at a glance.



(a) Force-directed node-link diagrams



(b) Adjacency Matrices with overplotting



(c) Graph Thumbnails

Figure 6.1: Three ways to represent the 70% and 90% confidence protein-protein interaction networks for E. coli. The graph shown on the left of each subfigure has 3,273 nodes and 15,631 edges, while the graph on the right has 4,038 nodes and 37,072 edges.

The main contributions of this chapter are as follows:

1. We propose a novel *Thumbnail* representation for networks (Figure 6.1(c)) that:
   - takes linear time in the number of edges of the network;
   - always produces the same visual for a given input graph structure—and hence is canonical in the sense that isomorphic graphs will always have identical thumbnails; and
   - provides precise information about the graph structure in a readable way.
2. We detail the design considerations and evolution of our thumbnail design, via a user study, both in terms of the choice of decomposition used to obtain a representative hierarchy of the graph, and the visual elements of the thumbnails (Section 6.3).
3. We evaluate people's ability to quickly judge the similarity of different types of graph structure using thumbnails, node-link and matrix representations of large graphs (Section 6.5)—in support of **Identification** and **Comparison** tasks, via a second user study.
4. We evaluate people's ability to read structural details about graphs using these three representations (Section 6.6)—in support of **Overview** tasks.
5. We explore a detailed application of *Graph Thumbnails* for understanding a set of networks modelling protein-protein interaction across different organisms and their growth (due to biologists' growing understanding) over time (Section 6.7)—**Comparison** and **Overview** tasks.

6.2 Background and Related Work

Most existing work on network visualisation has focused on exploring detailed structure within a single network. However, comparison has been mentioned (briefly) by Lee *et al.* [212] in their task taxonomy for graph visualisation. Krzywinski *et al.* [205] introduce hive plots, which arrange nodes on radially oriented axes according to their characteristics, but suffer from the problem of scalability due to crossings of dense edge lines, and are computationally non-linear. Small-multiple visualisation has been used before when comparing timesteps in dynamic graphs, and where the set of nodes in the graph is wholly or partially the same in each graph (if nodes can appear and disappear). For example, Burch and Weiskopf [74] use a small-multiple 'edge splatting' technique where the node positions do not change, but high-level patterns in edge connectivity can be seen to change between graphs. Similarly, Bach *et al.* [42] explored small-multiple visualisation of dynamic brain activity networks, comparing experimental data collected from a number of patients across time. Each network was represented by a matrix view, which was found to support the task of weighted link comparison well. However, the networks were small, modelling the relationships between only 30 or so brain regions (nodes).

Thus, it seems most past work on network comparison has focused on detailed comparison of individual changes in edge weights or neighbourhoods. In general, we have found little past work on visual techniques for comparing high-level network structure

over large numbers of graphs and with arbitrary nodesets. Non-visual analytical techniques have been developed for classifying and determining structural similarity between sets of networks (e.g. [225]), or to visualise sets of structural statistics across a graph corpus (e.g. Kennedy *et al.* [189]—we use this 'graph landscape' information to classify the corpus of graphs used in Study 1, Section 6.5). Kairam *et al.* [185] developed a visual dashboard display of structural summary statistics for a network (such as centrality metrics). Similarly, Freire *et al.* [131] use a visual dashboard display to act as an analysis tool for multiple networks. However, understanding the display required a good understanding of the metrics, and the dashboard was proposed as a complementary view to a node-link diagram.

To make visualisation scale to large networks without creating overwhelming detail, some sort of aggregation is necessary. For node-link diagrams an obvious target for aggregation are the links, being potentially far more numerous than nodes and causing clutter through crossings. The link (edge) lines can be spatially [344] or structurally [43] grouped and drawn as bundles. Cliques and bi-partite cliques of nodes can be identified and their links implied through aggregate connections [103, 104, 270]. Alternately, the links can be omitted entirely and, instead, communities of highly-linked nodes represented in contiguous coloured regions [135]. However, all of these techniques require complex algorithms with running time greater than linear in the number of graph elements.

There are faster techniques for finding hierarchical decompositions of graphs. Major approaches include SPQR-trees for the decomposition into tri-connected components, in particular, for planar graph drawing [99], k-core decomposition [55], [144, 182] or even simple stochastic sampling [240, 307]. Seidman introduced the notion of a k-core [284] of a graph G, a maximal connected induced subgraph $H = (V', E')$ of $G = (V, E)$ where $V \subset V', E \subset E'$ such that, for the minimum degree $\delta(H)$ it holds $\delta(H) \geq k$. The concept was later extended to weighted graphs [145], and further generalised to so-called *nuclei*, which represent more complex connectivity structures, but are also harder to intuitively interpret [280]. Furthermore, k-core decompositions have been explored in the past as a basis for layout of nodes in large and dense networks into concentric circular 'shells' [26] or 2.5-dimensional levels [57], reflecting their coreness. In particular, Alvarez *et al.* [26] identified the utility of such drawings for so-called 'finger-printing' of large graphs, implying that the 'shells' of nodes formed a pattern that was distinctive enough for gross structure identification purposes. A number of works have used similar approaches, using various decomposition techniques to emphasise high-level structure in large-scale node-link visualisations. Archambault *et al.* [29] decompose large graphs as far as bi-connected components before choosing a layout algorithm most suitable for the structure of each of those components. In each of the above methods, however, the full set of nodes and edges is involved in the layout meaning that the visual clutter is too great for small-multiples comparison. Very recent work by Zhang *et al.* [340] has explored the use of a terrain metaphor to visualise attributes associated with the nodes and the edges of a graph. They consider various attributes in their examples, including k-core depth. Their 3D renderings could be considered a hybrid of our icicle and treemap representations.

6.3 Design

Our concept for a 'thumbnail' representation of a large graph that supports identification, comparison, and overview tasks, has two key elements: the hierarchical decomposition technique and the visual representation of this hierarchy. In most respects it is possible

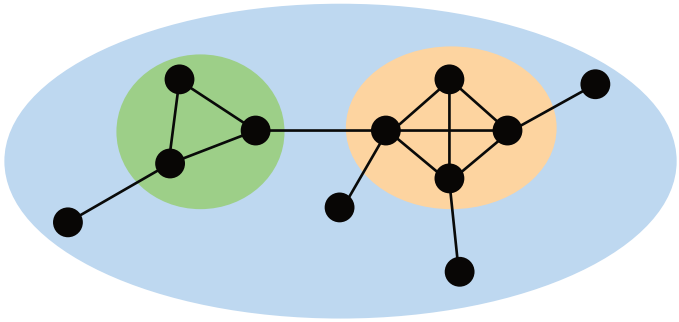


Figure 6.2: A network where all the vertices are connected to each other. The blue ellipse represents the 1-connected component. The green one represents the 2-connected component, while the orange ellipse represents the 3-core component. In this example, one 1-connected component contains all the vertices of the network. This network has two 2-connected components within the same 1-connected component. The orange ellipse is also 2-connected since our decomposition yields a hierarchical tree. The sole 3-core component of this network can be found by recursively removing all the vertices that have a degree of 0, 1, and 2 consecutively.

to separate these two concerns. As will be discussed, the semantics of the decomposition used in the examples in this chapter influence the colour scheme of our visual design, but otherwise, it is possible to create a thumbnail view from any hierarchical decomposition. As stated earlier, the particular decomposition explored here is chosen because of its linear running time and unambiguous (canonical) nature.

6.3.1 Hierarchical Graph Decomposition

We want each of the elements in our thumbnail network representation to say something very concrete about the structure of the graph. That is, a visual element of the thumbnail (such as a circle) should correspond to a substructure in the graph that is precisely definable. As a counter example, community detection methods fail this test since they typically seek to optimise a non-convex function over the sets of nodes assigned to clusters. For example, Girvan-Newman clustering [146] uses a greedy method to attempt to maximise the modularity score of each cluster. We cannot really say anything concrete about a cluster hierarchy obtained using this method because: a) it is likely only an approximation of the maximal modularity cluster decomposition; b) even if an optimum can be found, it is likely not unique; and c) a particular cluster's modularity is only defined relative to the rest of the graph structure. Furthermore, modularity-maximising clustering methods are not suitable for our purposes because even approximation methods for the (otherwise NP-hard) optimisation problem trade off running time with precision and even the fastest (and most imprecise) remain super-linear. Clusterings obtained using random walks (e.g., [254]) have also become popular in recent times, but these also trade off precision with running time.

An intuitive approach to finding dense substructures, which are components of significant functional relevance in many application areas like biology or the social sciences, would be to calculate a hierarchy of k-connected subgraphs in the network, i.e. subgraphs for which, between each pair of vertices, there are at least k vertex-disjoint paths. Figure 6.2 shows an example of a network where all the vertices belong to the same 1-connected component. The network also has two distinct 2-connected components, where

one contains three vertices, and the other, four. So called k-vertex-connected components are also easily defined (explained) using Menger's theorem as maximal components in which at least k nodes must be removed in order to split the component into 2 or more connected components. However, a decomposition of k-vertex-connected components (only) is not a practical approach for large graph browsing, for a number of reasons.

While k-vertex-connectivity of a graph for values of k up to 3 can be tested in linear time, and a graph can be decomposed into so-called tri-connected components in linear time ([98, 150]), these components are not necessarily 3-connected, and extracting 3-connected components from them is not straightforward. To the best of our knowledge, there is no practical linear time algorithm available to accomplish this.

In addition, the decomposition into k-connected components is not unique for $k \geq 4$, and thus, it is not well-defined as to what the set of k-connected components would be for general k. Consequently, there is no canonical hierarchical decomposition for k-connectivity known. A similar decomposition was described by Carmesin *et al.* [75], only for so-called k-blocks, as a concept based on the $(k-1)$-inseparability of the block within the original graph. This concept is more difficult to interpret in practical applications, e.g. the nodes do not even need to induce a connected subgraph, and there is also no linear-time algorithm known to compute the decomposition. In addition, it is not easy to make a comparison of similar structures over a set of graphs based on their k-block structure.

By contrast, k-cores for $k \geq 3$ are relatively straightforward, canonical, and a full k-core decomposition can be computed in time linear in the number of edges, due to an algorithm by Batagelj and Zavesnik [55]. A k-core of a graph is a component of the subgraph found by repeatedly removing vertices that have degrees less than k. Figure 6.2 shows an example network with a 3-core subgraph.

In light of these concerns, our final choice of decomposition (which we name a k-core-component clustering or KC^3 decomposition) for *Graph Thumbnails* is relatively simple:

- Level 1 is made up of singly-connected components.
- Level 2 consists of bi-connected components.
- Level 3 is 3-cores that are contained within a bi-connected component.
- Levels 4 and up are k-cores $(k > 3)$.

Clearly, all bi-connected components are contained in singly-connected components and all k-cores $(k > 3)$ are contained in $(k-1)$-cores. A challenge is that 3-cores are not strictly contained in a bi-connected component. To enforce a strict hierarchy we begin computing the core decomposition using bi-connected components as starting points.

For Study 1 we used an earlier version of KC^3 using tri-connected components (the R nodes of an SPQR decomposition) for Level 3 and beginning the search for k-cores for Levels 4 and up $(k > 3)$ within these tri-connected components. However, in Study 2 we wanted the participants to be able to understand the decomposition in order to make precise assessments of the graph structure. We found people were confused by the definition of tri-connectivity and so we opted for the simpler definition above.

Figure 6.3 shows an example of applying this decomposition on the popular network of Zachary's Karate Club [339].

In summary, KC^3 returns a canonical and hierarchical tree for the 1, 2-connected and k-core components of a network, which can be computed in linear time.

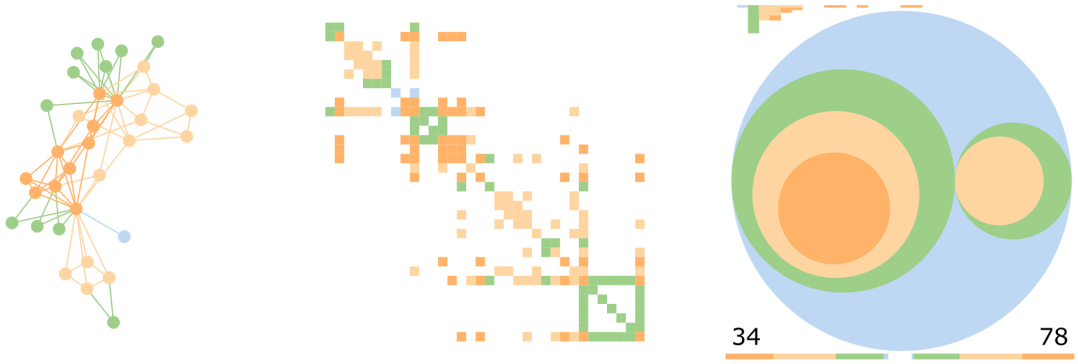


Figure 6.3: Zachary's Karate Club network [339], shown using node-link, matrix and *Graph Thumbnail* representations. The colours show the four levels of the KC^3 decomposition. This network is known for its two main communities: one around the administrator and a denser one around the instructor. These communities are respected in the KC^3 decomposition and can be clearly seen in the *Graph Thumbnail* representation.

6.3.2 Visual Design

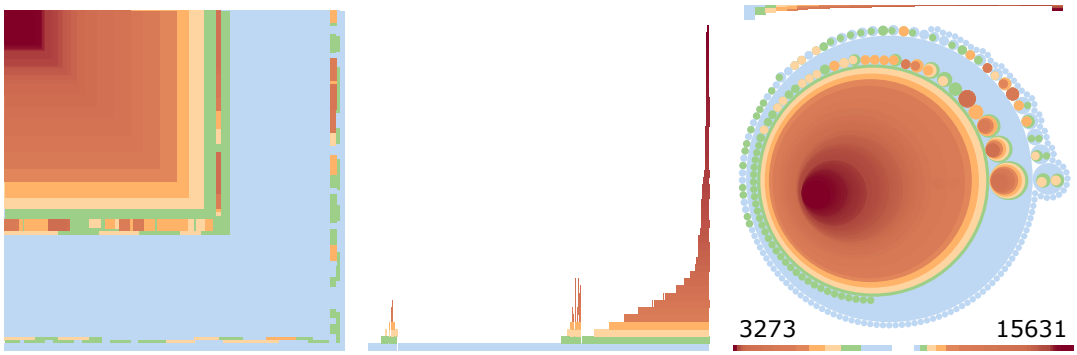
As previously mentioned, the inspiration for *Graph Thumbnails* was to have a small visual, showing a minimum amount of detail while still providing sufficient insight into the structure of the network; suggestive of a conventional large-scale network visualisation. We can expand upon these requirements for *Graph Thumbnail* visuals as follows:

- **R1** - suggestive of a standard large-scale network visualisation technique;
- **R2** - as readable as possible to support small-multiple comparisons;
- **R3** - the level in the cluster hierarchy of each visual element to be clear;
- **R4** - the visual elements accurately convey the relative size of the cluster to which they correspond; and
- **R5** - the representation should scale to arbitrarily large or dense networks.

Beginning by considering **R1**, force-directed layouts such as those in Figure 6.1(a) suggest an arrangement of circles. Adjacency matrices, such as those in Figure 6.1(b), suggest something more square, such as a treemap. If we attribute less significance to **R1**, a third option is to avoid the inaccuracy of area encoding and use a mapping where the length of the visual elements precisely encodes the size of clusters.

Treemaps

Treemaps over a cluster hierarchy have been considered before by Muelder and Ma as a layout strategy for large graphs [102, 231]. Their approach used the treemap slice-and-dice algorithm to arrange the nodes inside nested rectangular regions corresponding to the cluster hierarchy, then the full set of edges were overlaid on the node arrangement. In the resulting visualisation, the cluster hierarchy is still evident, but the clutter (due to edges) meant that the final rendering suffers from similar problems to large-scale force-directed layout (i.e. the 'hairball effect'). One possible thumbnail design, then, is simply to render the cluster-hierarchy treemap without the edges (see Figure 6.4(a)). Being space-filling,



(a) Treemap (b) Icicle plot (c) Graph Thumbnail

Figure 6.4: Comparison of the three visual designs explored for *Graph Thumbnails*.

treemaps arguably have the advantage that they maximise the 'data ink' and are, thus, efficient for small-multiple representations (**R2**). The number of nodes involved in each element of the cluster hierarchy is indicated by the area of the corresponding rectangular mark in the treemap (**R4**).

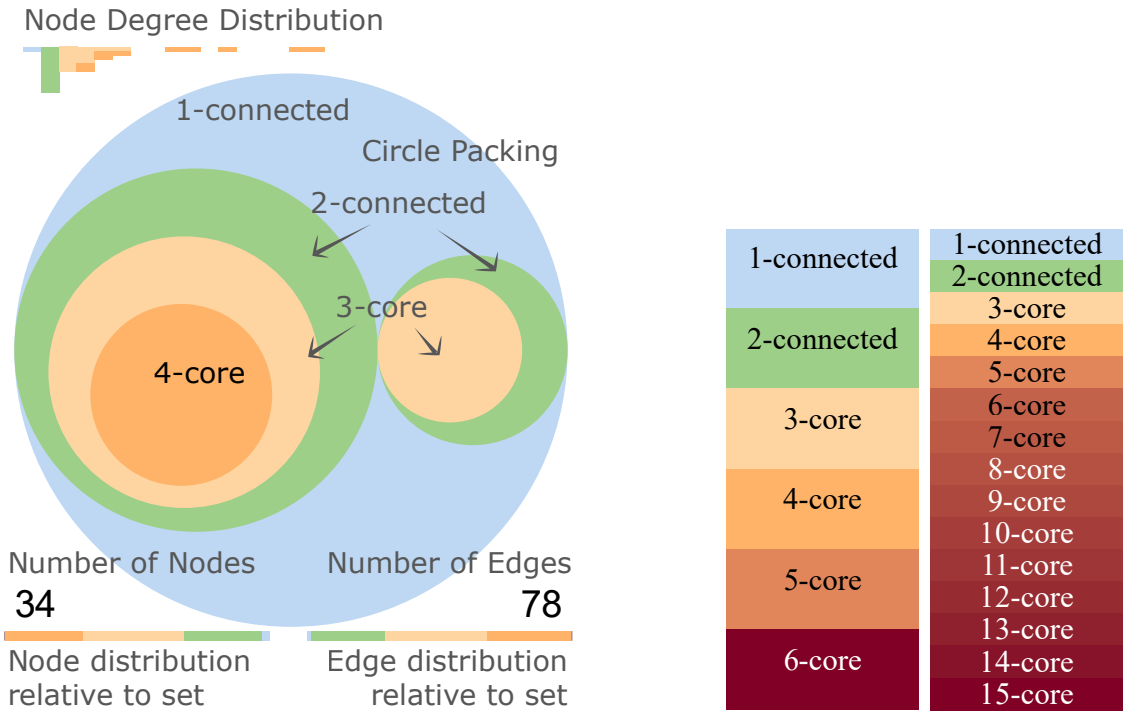
Icicle Plots

It is well known in psychology [292], and reinforced by experimental results in readability of information graphics [83], that encoding a scalar value with area causes people to underestimate the true value. Thus, we also considered icicle plots [204], which have the advantage over treemaps by using a length encoding of the marks to precisely indicate cluster size (**R5**). Also, depth of each cluster in the hierarchy is precisely readable from the vertical position of the marks (**R4**), while in the treemap, we rely on the colour encoding and rectangular containment to indicate hierarchy depth. Unfortunately, we found that the minute features of icicle plots for large graphs, such as can be seen in Figure 6.4(b), became unreadable when reduced to thumbnail size (**-R2**, **-R5**).

Circles

The third design option we considered, involved a nested 'circle packing'; employing the algorithm proposed by Wang *et al.* [316]. While not completely filling the available rectangular area, like a treemap or an icicle plot, a 'circle packing' still gives a good aspect ratio, a reasonable use of space (**R2**, **R5**), and uses circle area to indicate the size of clusters (**R4**). In our piloting for Study 1, we eventually settled on circle packing as the preferred thumbnail design over treemaps, as we felt they convey a stronger sense of hierarchy (**R3**). In addition, the empty areas at the corners turned out to be useful gaps where we could place additional adornments, described below. We preferred them over icicle plots, due to the aforementioned issue of scalability. The evaluation done by McGuffin and Robert [223] shows that leaf nodes in nested circles have a larger portion of the overall area than those in icicle plots. They also show that nested circles have a better distribution of area across the different levels of the tree than icicle plots.

Of the three thumbnail representations considered, the circle packing is also, arguably, the most strongly reminiscent of a conventional large-scale network visualisation, being suggestive of the 'blob' structure of a large-scale force-directed layout (**R1**).



(a) Layout, annotations and adornments.

(b) Adaptive colour scheme.

Figure 6.5: *Graph Thumbnail* visual design. We use the *Pack* function from *D3* [3], which uses nested circles to represent a hierarchy. The sizes of the circles are representative of the number of nodes contained within the respective component.

Colour Scheme

Our colour scheme evolved between Study 1 and Study 2. The original colour scheme used in Study 1 is shown in Figure 6.7. Essentially, we used a diverging palette with warm colours for the single-, bi- and tri-connected components, and cool colours for the *k*-cores. The mid-colour (white) was chosen to align with the special case of 4-core children of tri-connected components.

For the quick similarity task in Study 1, the participants only saw each pair of thumbnails for three seconds at a time. Thus, other than needing to provide good contrast, the precise choice of colour scheme was less significant for Study 1 than for Study 2; the latter requiring much closer inspection to interpret structural details. Also, as previously mentioned, for Study 2 we modified the KC^3 clustering to transition from bi-connected components to *k*-cores at level 3. Thus, while piloting for Study 2 we tested a new colour scheme that inverted the mapping of cool and warm colours. Further, we opted to have distinct hues for the sparser clusters. Thus, in our final colour scheme, singly connected components are blue, bi-connected components are green, 3-core children of bi-connected components are yellow and higher cores are interpolated from red to dark brown. Thus, there is a clear mapping of warmth of colour to density of connectivity within each cluster level.

One issue with colour selection is that there is a trade-off between clear readability for the higher-level (sparser) clusters and scalability, to very dense graphs with many levels of cores. It is also a trade-off between having a canonical colour scheme (the same across all graph sets) and maximising the discriminability of adjacent colours when the core hierarchy is very deep. Our solution is to fix the colours for the first five levels, and

then interpolate from red to maroon in the LAB colour space for levels corresponding to k-cores where $k \geq 6$, as shown in Figure 6.5(b).

With the examples in this chapter, we also use this colour scheme for the elements of the node-link and matrix representations. That is, we colour the nodes by their deepest cluster level membership and the links by the deepest common cluster-level membership of the two nodes that each connects. These coloured node-link and matrix representations were additional conditions in Study 2.

Annotations and Adornments

In several examples shown in this chapter, as well as the use-case described in Section 6.7, the *Graph Thumbnails* were augmented with additional annotations. Examples of these annotations are detailed in Figure 6.5(a). Note that these annotations and adornments were not used in the studies where we were concerned with testing the readability of the basic thumbnail design and comparing it with standard node-link and matrix representations (which typically do not use such adornments).

The size of a *Graph Thumbnail* is independent of the number of nodes and edges in the network. This allows for matching of networks with similar structural properties independent of size. However, the absolute numbers of nodes and edges are key identifying features of a network, so we choose to add these numbers as labels to the lower left and right corners of the thumbnail. When a large set of graphs are being compared with small-multiples, it can be useful to have a visual encoding of these numbers. Thus, we add bars below these numbers where the length of the bar double encodes the number of nodes/edges. Since *Graph Thumbnails* need to display graphs with a wide variety of sizes, we choose to make the lengths of these bars relative to the set of graphs displayed in a given small-multiples matrix, with the maximum length being just less than half the width of the thumbnail. This allows relative sizes of graphs within the set to be compared at a glance. Within the node and edge count bars, we further show the breakdown of these elements into the various clusters, with colour bands corresponding to the level-hierarchy colour scheme.

Node degree distribution is a method frequently used as a rough profile of graph structure. We display an inverted histogram across the top of the thumbnail where the height of each bar indicates the number of nodes of a given node degree, with lower degree nodes on the left. 0-degree nodes (if any) will be the left-most bar and coloured grey. Within each bar, the membership of nodes of different degrees to clusters is again indicated with coloured bands. The right-most bar is a cumulative count of nodes of degree > 30. This maximum prevents the bars from becoming too thin in very large and dense graphs.

6.4 Algorithm Scalability

The algorithm to create *Graph Thumbnails* has three stages.

1. **Hierarchical Decomposition**. This returns the different connected and core components of the network as described in Section 6.3.1. This takes linear time in the number of edges.

2. **Canonical Encoding**. In this stage of the algorithm, we encode the nodes of the tree, which is acquired by the first stage of the algorithm, in order to achieve a

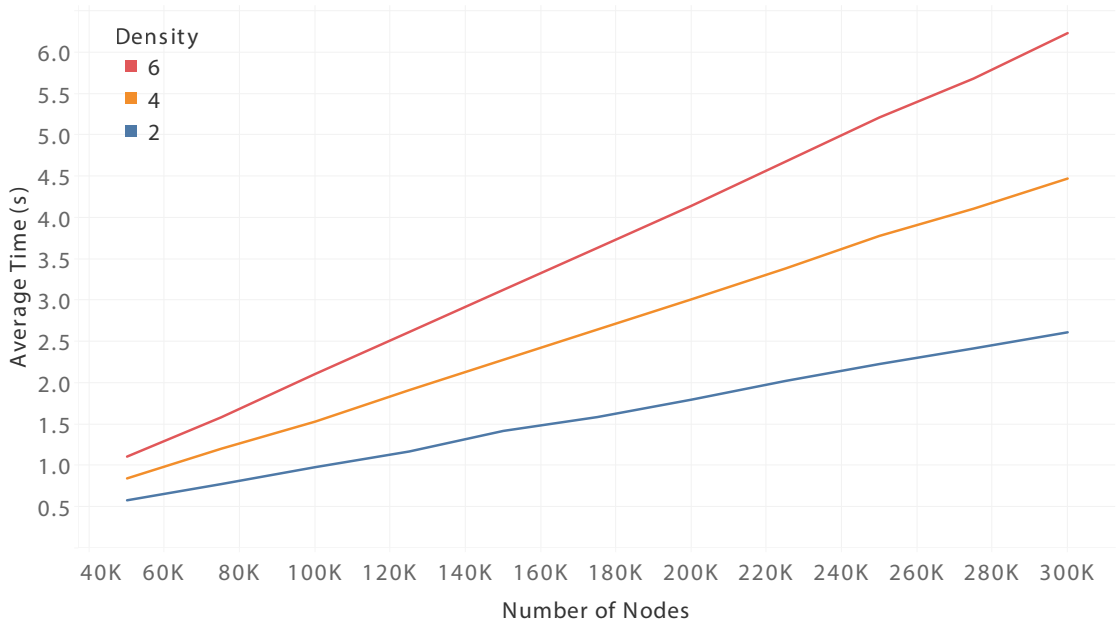


Figure 6.6: The average running time of creating *Graph Thumbnails* for large graphs of different sizes. We were able to represent graphs with 300,000 nodes and 1,800,000 edges in approximately six seconds.

canonical ordering. The canonical encoding of a tree can be computed in linear time [162, Chapter 3].

3. **Circle Packing**. We use the *D3* [3] implementation of the circle packing algorithm proposed by Wang *et al.* [316]. The positions of the circles are based on the canonical ordering achieved in the previous stage. We use the standard JavaScript sort which requires $O(m \log m)$ time, where m is the number of components in the tree returned by the Hierarchical Decomposition. However we could use the *lexicographic sort of strings of varying length* instead, which takes $O(m)$ time [162, Chapter 3].

We conducted an experimental evaluation to verify that, in practice, our algorithm for creating *Graph Thumbnails* takes linear time in the number of edges.

We used a *NetworkX* [154] random graph generator based on the *Watts-Strogatz* model to generate 25 small-world graph instances of 11 different orders (number of nodes) and three different densities. The order of the graphs ranged from 50,000 to 300,000 nodes, with 25,000 nodes added at each step. To obtain different graph densities, we set the number of edges to be a multiple of the number of nodes, thus in Figure 6.6 the number of edges increases linearly with respect to the number of nodes. We used 2, 4 and 6 times the number of nodes to decide the number of edges. We ended up with a total of 825 graphs.

The times reported in Figure 6.6 are the complete times required to create a *Graph Thumbnail* representation for each graph size. The results clearly show the linear trend in running time with number of edges. We were able to create a *Graph Thumbnail* representation for graphs with 50,000 nodes and 100,000 edges in 582 milliseconds, graphs with 225,000 nodes and 1,350,000 edges in under five seconds, and graphs of 300,000 nodes and 1,800,000 edges in 6.24 seconds.

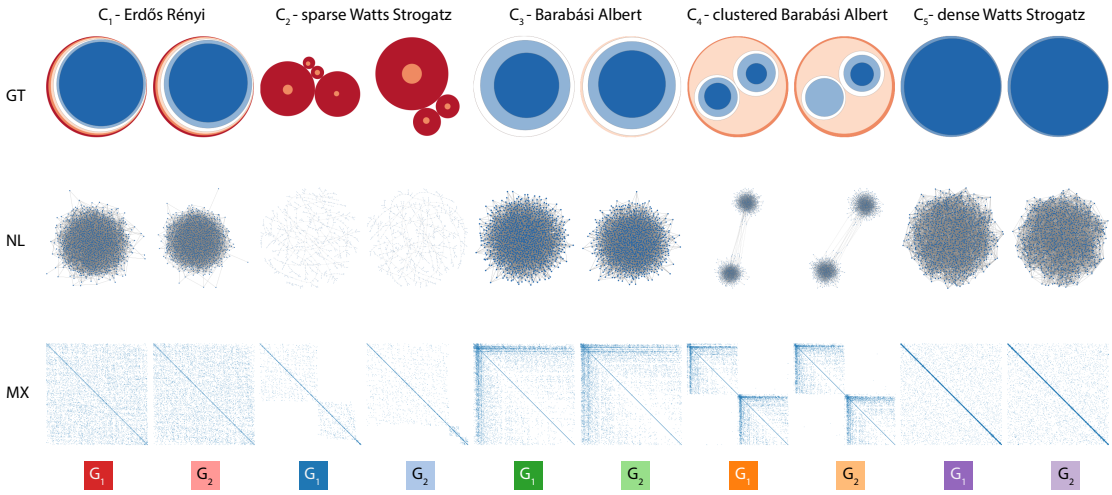


Figure 6.7: Study 1 stimuli. Ten graphs from five graph classes, shown in three visual representations: *Graph Thumbnail*, node-link and matrix.

6.5 Study 1: Identifying Similarity

Our first study evaluated people's ability to differentiate large graphs with different structural properties using *Graph Thumbnail* (GT), node-link (NL) and matrix (MX) representations. The study aimed to show how much the three visual representations aided the **Identification** and **Comparison** tasks. We chose to compare against matrix and node-link views due to their wide use in practice, which is not shared by the more novel techniques for aggregated views identified in Sections 6.2 and 2.5.

We came up with five different graph classes based on different generating techniques. In general, we focused on generators that produce small-world and scale-free graphs, as these arise frequently in important application areas, such as Biology and Social Sciences.

The first C_1, second C_2 and fifth C_5 classes were based on two *NetworkX* [154] generators: Watts-Strogatz and ErdősRényi [283].

The Watts-Strogatz generator returns graphs with small-world properties, such as short average path lengths and high clustering, whereas the ErdősRényi generator returns binomial graphs, which have low clustering coefficients and do not have many hubs.

The parameters of Watts-Strogatz were changed to yield sparse and dense graphs, giving the second and fifth classes respectively.

The third class C_3 was based on the BarabásiAlbert model [51]. The BarabásiAlbert generator uses preferential attachment to return scale-free graphs.

The fourth class C_4 used a hybrid generator for clustered graphs. It employed the BarabásiAlbert model to generate two subgraphs and merged them by randomly adding edges between the low-degree nodes of each. We generated two graphs from each of the five graph classes, leading to the ten graphs shown in Figure 6.7, each with 1,000 nodes.

The NL stimuli were created using the force layout of D3.js [3]. MX stimuli were created and ordered using Reorder.js with barycenter reordering [128]. At first, we found that all but the diagonal portions of the matrices were unreadable, since the individual cells were smaller than a pixel when scaled to thumbnail size. Thus, we increased the cell-size, allowing each filled cell to overplot its adjacent eight cells. We feel this is a necessary modification for any large-scale use of matrices in order for patterns and outliers to remain visible.

6.5.1 Procedure

The study had 21 participants: 16 male, 5 female. The participants were shown an explanatory statement. After agreeing to participate, participants were asked questions about their background: how often they saw network diagrams and how familiar they were with the terms 'cluster' and 'connected components'. 7 participants often came across network diagrams, 11 occasionally did, while 3 never did. 7 of the participants were familiar with the terms 'cluster' and 'connected components'. In preparation for the task, participants were shown a set of diagrams generated using the same five generators used to produce the study stimuli. This was done to give them an idea of the range of similarities or differences in the diagrams of all three visual representations they were about to be shown. There was no additional explanation or training questions—we wanted participants to consider purely visual similarities, rather than have them try to interpret the visualisations.

We used a within-subjects design where participants were asked to rate similarity of diagrams within all three visual representations: GT, NL, and MX. Participants were shown pairs of diagrams from the same visual representation side-by-side at the thumbnail size of 300×300 pixels. Each pair was displayed for three seconds before they were hidden, and the participant was asked to rate their similarity on a five-point scale from 'Similar' to 'Different'. Participants were shown each pair of diagrams twice (both left-right orderings). The overall presentation order cycled between the three different representations, in random order.

The study was performed on site with a facilitator present. The study was presented to the participant using a custom website rendered by Google Chrome in full-screen mode on a 21-inch monitor with 1680×1050 pixel resolution. Participants were entered into a random draw for an AUD$50 voucher.

6.5.2 Hypotheses

- **H1** - We hypothesised that large graphs shown using MX would be more distinguishable than NL, even though the relationships between nodes are more direct in NL than MX. However, the fact that in MX they are assigned a non-overlapping pixel helps the discrimination task.

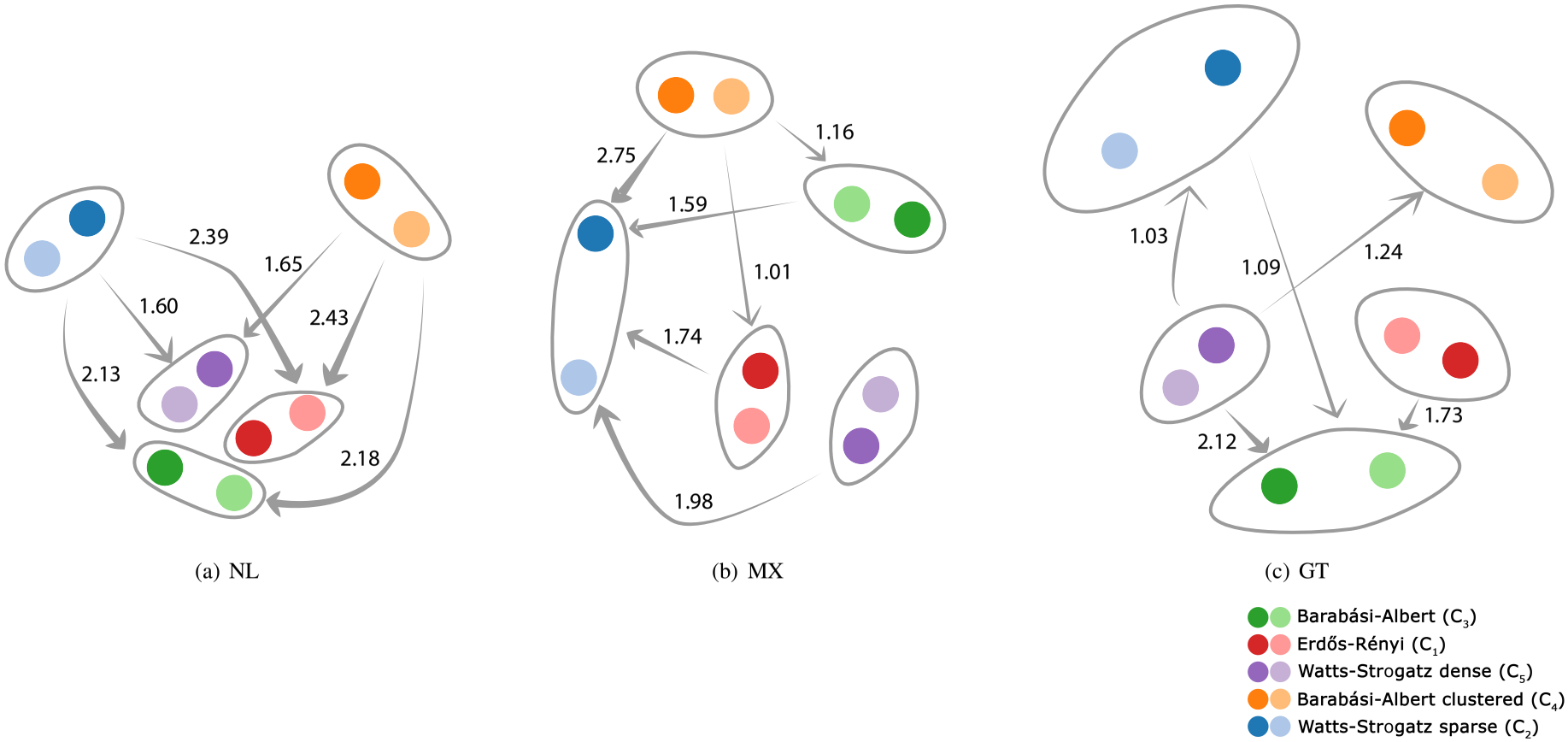
- **H2** - Similarly, we hypothesised that GT would allow large graphs to become more distinguishable than when using NL or MX, since GT is designed to show an overview and will have distinct, non-overlapping and well packed circles.

6.5.3 Results

We collected 270 similarity ratings for each user; 90 for each visual representation from all possible pairings of the 10 graphs excluding comparison of a graph with itself [1]. The graphs are divided into 5 classes, therefore, the full matrix of scores is denoted $S(C_iG_k, C_jG_l)$ $i, j \in \{1, \ldots, 5\}$ $k, l \in \{1, 2\}$.

Figure 6.8 shows multidimensional scaling plots of the mean similarity ratings between graphs for each visual representation. The distances inversely represent similarity, so pairs of graphs that participants considered more similar are closer together. The plots

[1] The full results are published online at vahany.com/gt-study1-results.html



(a) NL (b) MX (c) GT

Figure 6.8: Multidimensional scaling plot of the results of the first study. The distances inversely represent the mean of the similarity rating for each pair of graphs. The arrows show statistically significant differences in discriminability between graph classes. The direction goes from higher discriminability to lower.

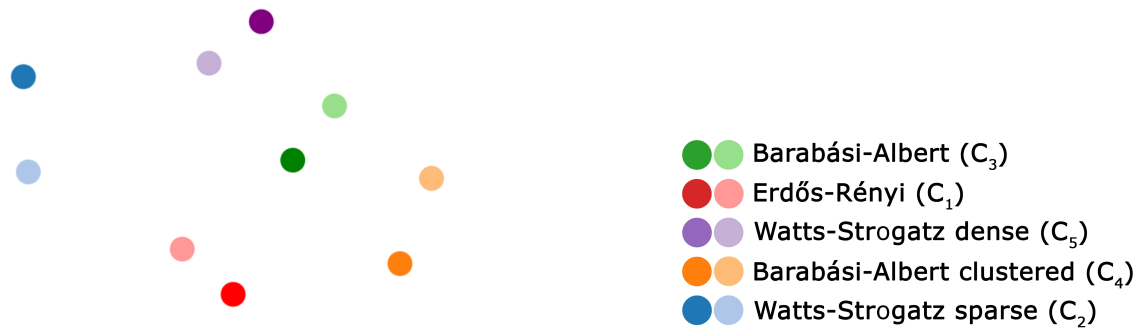


Figure 6.9: A multidimensional scaling plot with distances representing dissimilarity of 17 graph properties, computed using the *Graph Landscape* method [189]. We can clearly identify the pairs of graphs generated with the same methods, and the particularly large distance of the sparse Watts-Strogatz graphs to the rest. Hue represents the graph class.

suggest that GT and MX helped the participants identify the differences between graphs from different classes, since the dots with different hues are placed far from each other. Similarly, we notice that GT and MX helped identify the similarities of graphs that belong to the same class, since the dots with similar hues are placed close to each other. However, with NL, the participants could not differentiate between the six graphs that belonged to three different classes: C_3, C_5, and C_1.

The significant differences in discriminability between the classes, as indicated on the MDS plots by arrows, were determined as follows. For each participant and visual representation, and a given class C_i, we consider the rating provided by the participant of C_iG_1 and C_iG_2 to $selfsimilarity_{12}(C_i)$. i.e.,

$$selfsimilarity_{12}(C_i) = S(C_iG_1, C_iG_2) \tag{6.1}$$

For each class C_i we consider each participant's ability to differentiate graphs in C_i from all graphs not in C_i, while using each visual representation, with a *discriminability* score:

$$\begin{aligned} discriminability_{12}(C_i) = \\ \sum_{j=1, j\neq i}^{5} \Big(\sum_{k=1}^{2} \sum_{l=1}^{2} (selfsimilarity_{12}(C_i) - S(C_iG_k, C_jG_l)) \\ + (selfsimilarity_{12}(C_i) - S(C_jG_k, C_iG_l)) \Big) \end{aligned} \tag{6.2}$$

Ranging over k and l considers the pairs of graphs from classes other than C_i, shown in both normal and reverse order. For each class C_i we have a second similarity rating $selfsimilarity_{21}$ for the graphs within C_i, presented in reverse order, which gives us a second discriminability measure $discriminability_{21}(C_i)$. The possible range is $-128 \leq discriminability(C_i) \leq 128$, where 128 would occur if a participant gave the pair within C_i a rating of 5 (the highest) and gave all comparisons to other classes ($C_{j\neq i}$) a rating of 1 (the lowest). In the reverse case, $discriminability(C_i)$ would be -128, but in practice no participant had a negative score for any representation.

Overall, we had two scores for five pairs, three visual representations, and 21 participants. We analysed the results using the Kruskal-Wallis test, which showed that GT

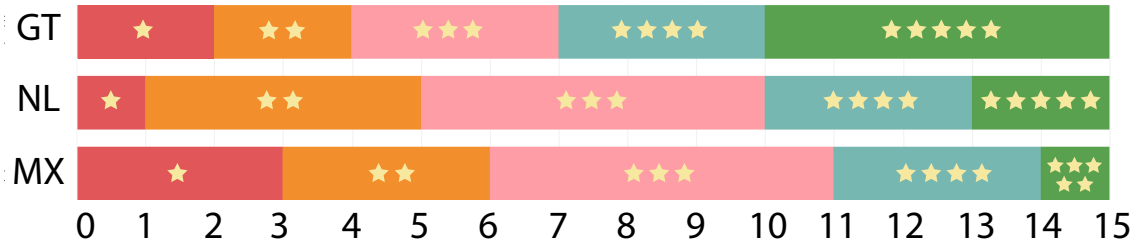


Figure 6.10: 15 participants submitted a 5-star rating of how easy it was to do the similarity tasks, given each visual representation. More than 50% of the participants rated GT on the easy side, while only 33.33%, and less than 30% of the participants, thought MX and NL were easy respectively.

and MX had significantly higher scores than NL (p-value $= 0.001716$). There was no significant difference between GT and MX.

We also checked if any particular pair of graphs had a significantly higher discriminability across the three visual representations. To do this, we filtered the set of measures by different visual representations into three subsets. We ran the Kruskal-Wallis test to check for significant differences between the measures across the five similar pairs.

The arrows in Figure 6.8 show the statistically significant differences in discriminability between the five classes. The direction goes from higher discriminability to lower. The label indicates ratio of observed over critical difference found using the multiple comparison test between different classes with $p < .001$ using the Kruskal-Wallis test.

While using NL, the participants were able to discriminate the class of sparse Watts-Strogatz graphs C_2 and the clustered graphs C_4 more easily than the other three classes: C_1, C_3, and C_5.

When using MX, the participants were able to differentiate all classes better than the class of sparse Watts-Strogatz graphs C_2. They also had higher discriminability scores for the class of clustered graphs C_4, over all other classes, except the class of dense Watts-Strogatz graphs C_5.

With GT, the participants could identify the class of dense Watts-Strogatz graphs C_5 better than all classes, except the class of ErdősRényi graphs C_1. They could also differentiate the class of sparse Watts-Strogatz graphs C_2 and the class of ErdősRényi graphs C_1 better than the class of BarabásiAlbert graphs C_3.

To check for any learning effect, we further partitioned the results into three equal sets, for the position of pairs within the entire set shown to each participant. We did not find any significant differences in the results across different stages of the study.

We asked participants to provide post-study feedback. They were asked to use a 5-star rating to express how easy it was to perform the task using each visual representation. 15 participants submitted feedback. Figure 6.10 shows the results of the 5-star rating of the post-study survey. More than 50% of respondents rated GT as being on the easy side, versus only 33.33% for MX, and less than 20% for NL.

Participants were also asked to describe how each visual representation helped them complete the similarity tasks. For NL, three participants mentioned that size and shape helped, while four mentioned density could be used to compare similarity. For MX, five participants said they used the position, location or distribution of dots, and 8 mentioned using the density or concentration for determining similarity. For GT, five participants said they used the circles, with five participants stating the circle nesting, hierarchy or pattern helped with the task. Several participants also mentioned either colour or sizes as

a beneficial feature. Five participants specifically stated the GT representation made the task easy.

6.5.4 Discussion

Analysis of the Study 1 results showed that both GT and MX allowed significantly better discrimination of graphs from different classes over NL, thus supporting **H1** and partially supporting **H2**. Even though the study showed no significant differences between the results of GT and MX, the feedback from the participants favoured GT over the two other visual representations.

While single graph properties, like diameter or average degree, are not well suited to discriminate between classes of graphs, combinations of such properties can be used for characterisation and comparison of graphs. However, due to the diversity of properties and the complexity of their interplay, it is hard to support a good comparison at a glance. The *graph landscape* is a concept for the visual analysis of graph structure that uses a large set of such graph properties for in-depth analysis of graph set characteristics and overlap [189]. An MDS map may be used to spot clusters and outliers, as shown in Figure 6.9 for the graphs of our study. The ground truth here is that pairs of graphs that are generated with the same method can be clearly identified, and the sparse Watts-Strogatz graphs have the largest distance to the other graphs. We would hope to see a similar landscape emerge from the similarity ratings observed by participants. Indeed, MDS plots of our analysis of participant ratings of similarity in Figure 6.8 do show a similar landscape with the same features.

Limitations: The design of GT has evolved somewhat since Study 1. At the time, it had a different colour scheme and a different structure as discussed in Section 6.3. However, since the rapid similarity ranking task was based on very high-level features, we expect the results to be repeatable with the new design.

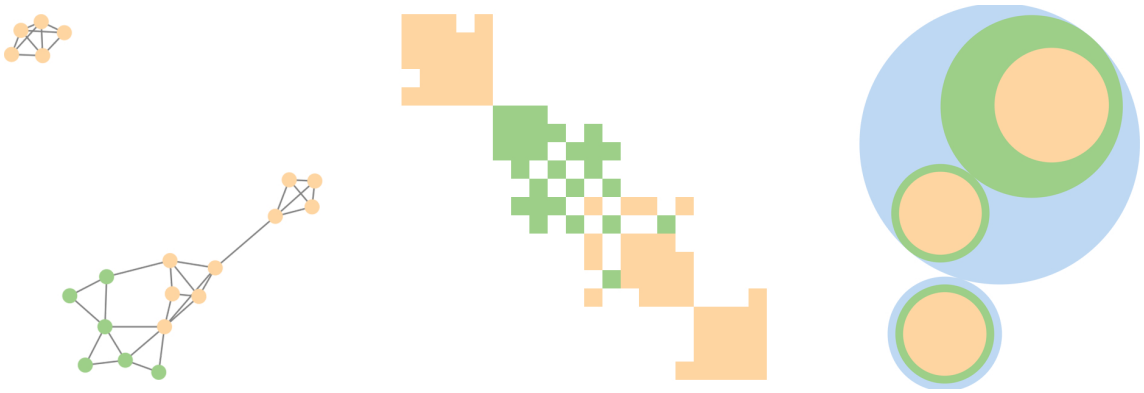
The networks used in the study were each 1,000 nodes. This size was chosen as we found larger graphs unreadable with MX or NL. The image size used in this study was 300×300 pixels. This could be considered large for a 'thumbnail' image. We believe that at smaller scales, GT would still remain usable, while NL and MX might not.

6.6 Study 2: Understanding Structure

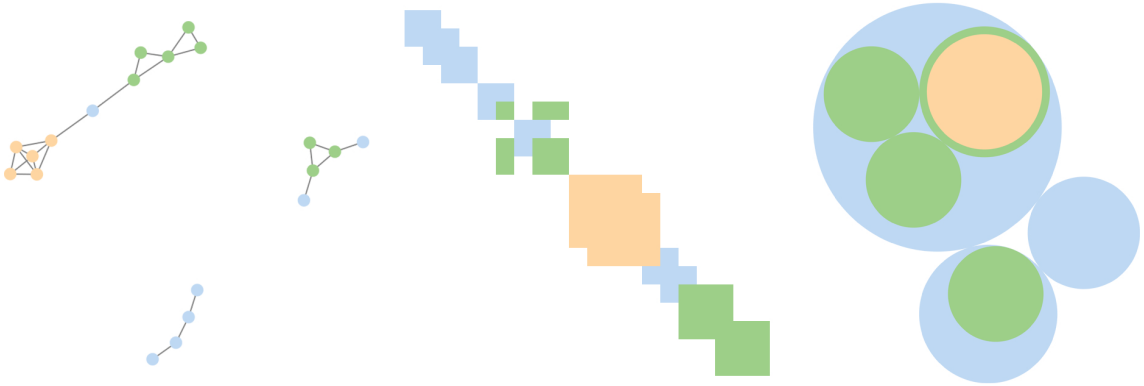
Study 1 did not require participants to interpret or understand the visual representations, only to notice visual differences supporting **Identification** and **Comparison**. We therefore performed a second study to evaluate whether GT, NL and MX convey structural properties sufficiently to support **Overview** tasks.

The study had four tasks using 12 different 20-node graphs, randomly picked from the Rome Graphs corpus [10]. The graphs were used unmodified for tasks 3 and 4. The first two tasks involved finding 1- and 2-connected components. For these tasks, the number of components in each graph were controlled by randomly adding and removing links.

As in Study 1, we used GT, NL and MX to show the graphs. For GT we used the final design colour scheme discussed in Section 6.3. We had two variations of NL and MX: grey (NLGrey, MXGrey) and coloured versions (NLColour, MXColour) using a similar colour scheme as GT. For MXColour, if the nodes connected by an edge belonged to different components of different levels, the cell representing the edge would be coloured according to the higher component.



(a) Task 1: 1-connected components (correct answer 2)



(b) Task 2: 2-connected components (correct answer 4)

Figure 6.11: Sample stimuli from Study 2. Answers were multiple choice (0…6 or unsure).

6.6.1 Procedure

The participants were shown an explanatory statement. After agreeing to this, they were asked questions on their background: how often they saw network diagrams and how familiar they were with the terms 'cluster' and 'connected components'. Participants then worked through two tutorials [12], which explained the basics of connectivity and presented the different visual representations. Each task had several training questions that had to be answered correctly before the participants could proceed.

Task 1

The participants were asked to count the 1-connected components of a network. The networks were shown using NL, GT, MX, MXColour, NLColour. In addition to the examples in Figure 6.11(a), the task included MXGrey and NLGrey diagrams. Piloting revealed that extra training was required for MX. Participants were instructed that in order for two nodes or blocks to be part of the same 1-connected component, they needed to overlap with at least one cell.

- **H1.1** We hypothesised that participants would be faster, more accurate and require less eye movement while using GT than with MX.
- **H1.2** Similarly, we expected NL to be better than MX.

- **H1.3** We also expected colour encoding to help MX and NL, thus we hypothesised NLColour would be better than NLGrey.
- **H1.4** Similarly, we expected MXColour to be better than MXGrey.

Task 2

Counting 2-connected components. Again, extra training was required for MX: participants were told that in order for two nodes or blocks to be part of the same 2-connected component in MX, they needed to overlap by at least two cells (see Figure 6.11(b)). The example also shows that the colouring alone is not enough, since there might exist a bridge node between two 2-connected components.

- **H2.1** We hypothesised that GT would be better than MX.
- **H2.2** Similarly, we expected that GT would be better than NL
- **H2.3** We also hypothesised that NL would be better than MX.
- **H2.4** Again, we expected NLColour to be better than NLGrey.
- **H2.5** Similarly, we expected MXColour to be better than MXGrey.

Task 3

Shown a pair of networks, participants had to pick the one with more links. The networks were shown using all five representations.

- **H3.1** We hypothesised MX would be better than NL.
- **H3.2** Similarly, we expected MX to be better than GT. As mentioned previously, GT did not include annotations for this study (an edge count annotation would have made this task trivial).

Task 4

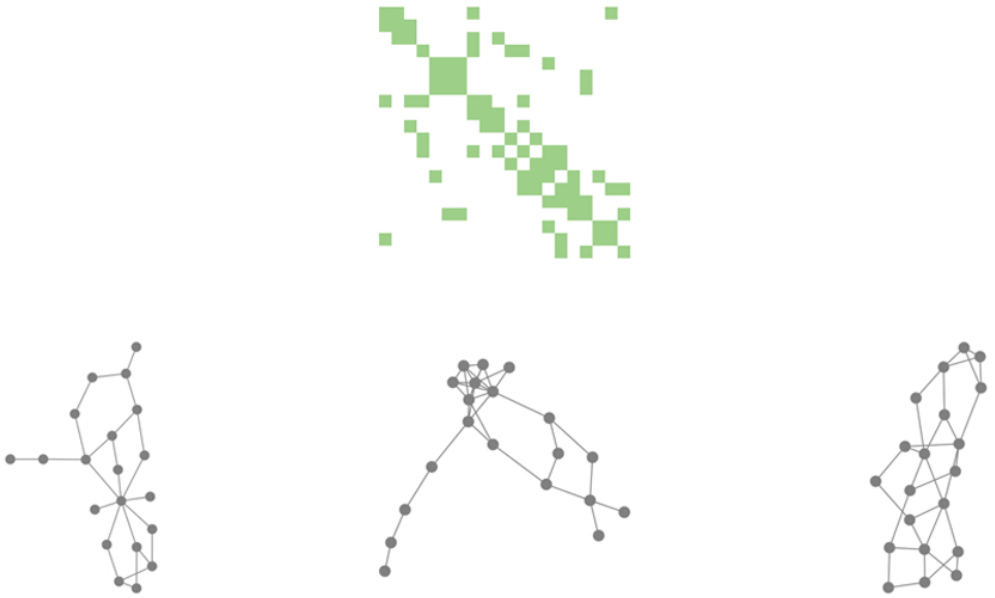
In the final task, the participants were asked to match a given reference graph shown using GT, MXGrey or MXColour to one of three NLGrey graphs (Figure 6.12).

- **H4** Since GT elides detail, we hypothesised that MX would outperform GT.

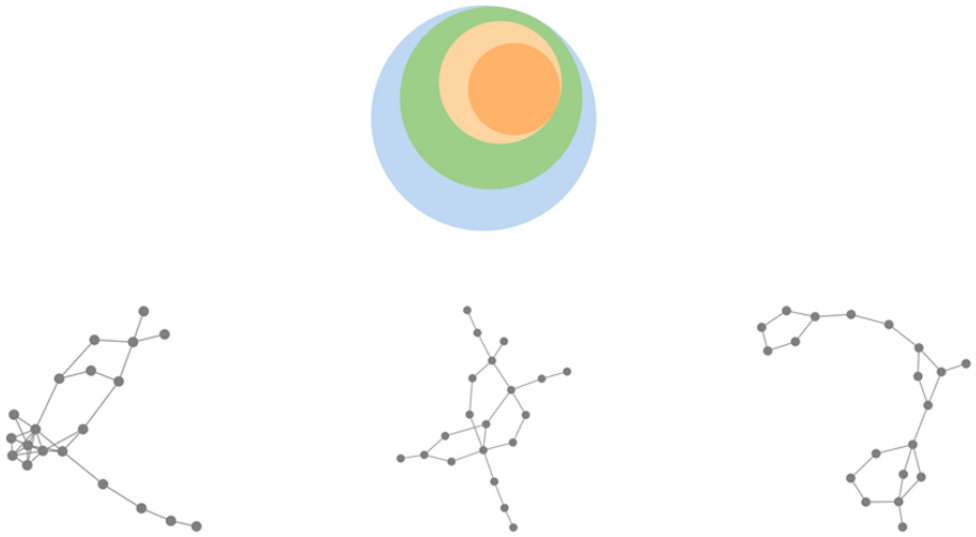
6.6.2 Setup

Tasks 1 and 2 had 10 training and 40 timed questions each. Task 3 had 3 training and 50 timed questions. Task 4: 6 training and 30 timed. A timeout of 10 seconds was applied for Tasks 1–3, while Task 4 had a timeout of 30 seconds; after which the graphs were hidden. The number of trials and the timeouts were decided upon through pilot studies. All visual stimuli were shown an equal number of times and their order was decided by latin square.

A Tobii pro x3-120 eye tracker was used throughout the study. The study was run on a Windows 10 Dell Latitude E7440 laptop, equipped with 2.7 GHz i7 processor and 8 GB



(a) MXColour reference graph (correct answer third NLGrey)



(b) GT reference graph (correct answer first NLGrey)

Figure 6.12: Sample stimuli from Task 4 of Study 2. From the selection of three NLGrey graphs, participants were required to select the one that best matched the reference.

RAM. It was shown on a 22-inch HP monitor, using a Mozilla Firefox 47.0 browser. The eye tracker was linked to the laptop through a Tobii pro external processing unit.

We had 26 participants (eight of whom also participated in the previous study): 19 male, 7 female; 22 aged 20–30, two aged 30–40, and two 40-50. Six participants stated that they often saw network diagrams, 18 participants said that they occasionally saw network diagrams, and two said that they had never seen network diagrams before. 9 participants were not familiar with the terms 'cluster' and 'connected components', while 17 were.

6.6.3 Results

Tasks 1 and 2 each had 8 trials $\times$ 5 techniques $\times$ 26 participants = 1,040 trials. For Task 3, we had 10 trials $\times$ 5 techniques $\times$ 26 participants = 1,300 trials, and for Task 4 there were 10 trials $\times$ 3 techniques $\times$ 26 participants = 780 trials. In total we had 4,160 trials across all four tasks. On average, the participants answered 81.5% of all questions correctly, 15.7% incorrectly, and 2.8% were answered as 'unsure'. These 'unsure' answers were counted as incorrect in the analysis. Unless specified otherwise, response times are for correct answers. Eye movement was measured by calculating the distance between two consecutive gaze points, then summing them for each visual stimuli, task and participant. Full results are shown in Figure 6.13.

The following results are statistically significant using the Kruskal-Wallis test with $p < .001$.

In Task 1, results strongly support H1.1: participants had higher speed and accuracy, and less eye movement when using GT over MX. Results also strongly support H1.2: participants had higher speed and accuracy, and less eye movement when using NL over MX, except that the results did not show any significant differences in accuracy between NL and MXColour. We did not find any significant differences to support H1.3 and H1.4—colour highlighting of component hierarchy in NL and MX were of little benefit.

The results of Task 2 strongly support H2.1 (GT over MX) and H2.3 favouring NL over MX; as well as H2.2 (GT over NL) for speed and eye movement, but not for accuracy.

We did not find any support for H3.1 (MX over NL) and H3.2 (MX over GT). On the contrary, the results showed that the participants took less time overall when using GT and NLGrey over MX, and NLColour over MXGrey. They also took less time to perform correctly when using NLGrey over MXGrey.

The results did not support H4 (MX over GT). They did show that GT outperformed MXGrey in both speed and accuracy. They also showed that colour enhanced the speed and accuracy of participants for MX.

To check for any learning effect we partitioned the trials into three equal sets, for each task and each participant. We did not find any significant differences in the results across the three sets.

6.6.4 Discussion

The results show participants performed equally well using GT and NL on 1- and 2-connected component detection in Tasks 1 and 2, while they struggled in accuracy, speed and eye movement when using MX. In Task 2, participants performed slower and moved their eyes more while finding 2-connected components using NL than GT.

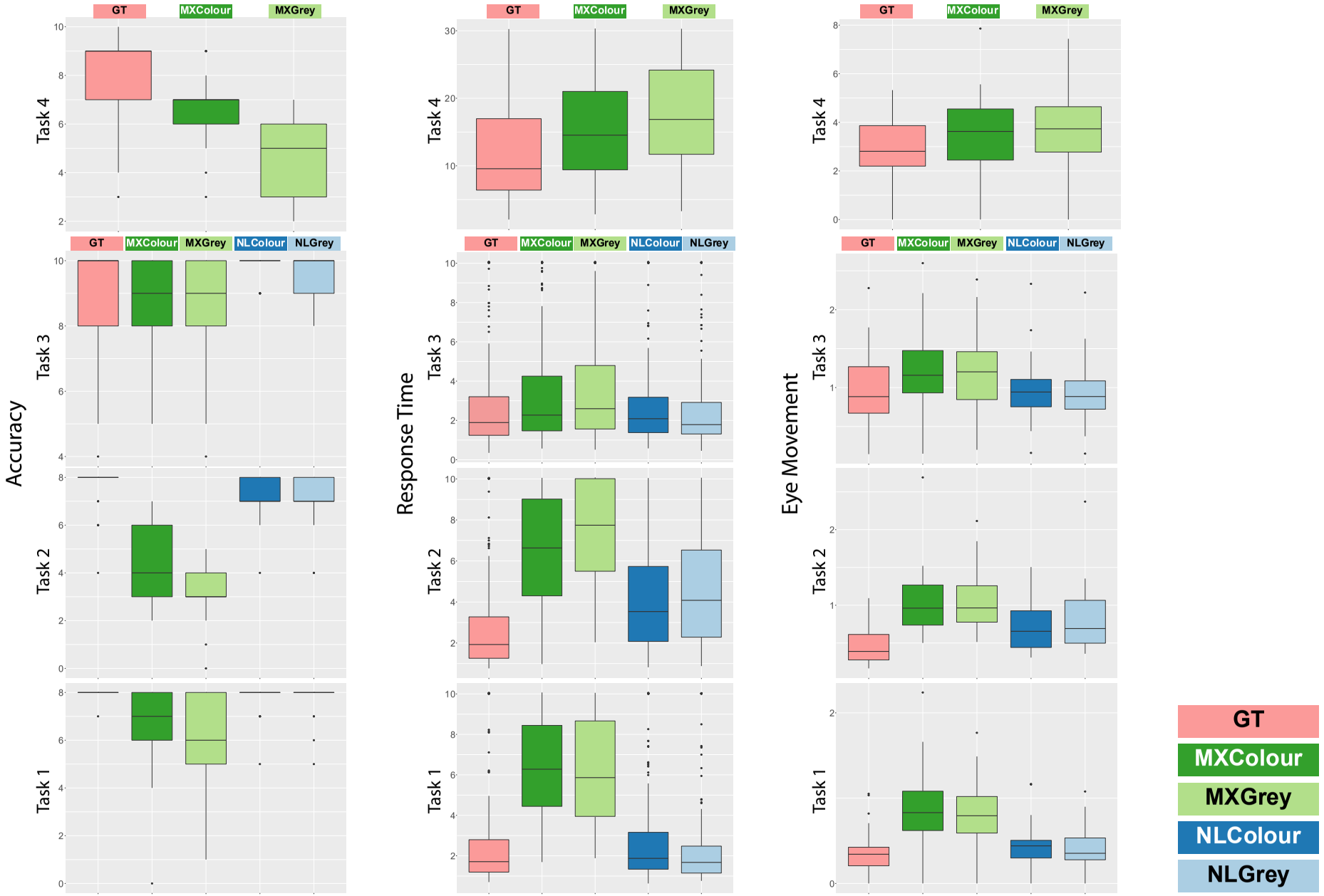


Figure 6.13: Study 2 results. All three measures: accuracy on the left, response time (in seconds) in the middle, and total eye movement (in 0.1 megapixels) on the right, show a similar trend where the participants performed better on *Graph Thumbnails* than matrix representations. They also performed better on node-link diagrams than matrix representations.

There were no significant differences in accuracy across the different stimuli for edge density estimation in Task 3. However, as mentioned earlier, if we included the adornments of GT, it would have been a simple task of reading and comparing the edge count directly from the adornments. Furthermore, we were surprised to find that both GT and NL aided the speed of the participants over MX, where edge-density should directly correspond to cell-density. Another surprising result was that of GT outperforming MXGrey in matching with NLGrey in Task 4.

Many participants said that the colour confused them when applied to NL and MX, which was born out in our results in all tasks except the matching in Task 4. Participants performed Task 4 better when using MXColour over MXGrey.

Generally, MX was outperformed by GT across all tasks. 10 participants expressed difficulties completing the tasks using MX, and the same number of participants said that performing the tasks using GT was easy. Overall, we attribute this to GT removing the detail of MX that was not directly necessary to complete each task.

Limitations: The study used only small graphs, with the size chosen through piloting to make the tasks possible in limited time using all three representations. The limiting factor was the scalability of NL and MX representations, while GT scales with less clutter, as seen in Figure 6.1 and Figure 6.14.

6.7 Use Case

We apply our approach to the analysis of protein-protein interaction (PPI) data. Analysis of PPI is an important step to better understand the complex mechanisms of life and diseases. Information about suspected and confirmed PPI is collected in a number of databases. Some of these only store manually curated data, while others also store automatically derived information, e.g. from literature mining. These databases grow over time, changing due to new evidence arising from experiments. Many PPI databases allow download of historical releases for comparison and to ensure that an analysis takes into account the properties of the corresponding release.

PPI networks model proteins as nodes and interactions as edges. There has long been debate on the structural properties of PPI networks, including argument of scale-freeness, i.e. connectivity distributed according to a power law [82,301], small-worldness, i.e. most nodes can be reached by most other nodes with only a few hops, and relatively high local clustering compared to random networks. As evidence for PPI can be derived from a multitude of sources, including experiments or transfer from other species, interactions can be highly dynamic. Hence, entries (edges) in PPI databases are usually assigned probability or confidence scores, due to the high rate of false negatives and false positives [312].

PPI networks are often represented as hairball node-link diagrams (Figure 6.1(a)), sometimes with the goal of showing complexity rather than structure. Common force-directed layout methods are usually unable to untangle such networks properly due to their small diameter. Biologists have a need to compare networks (e.g. between individuals, species, or conditions, and to investigate evolutionary changes), to gain an overview on the structural characteristics, and to deduce molecular function. Thus, all three of the visualisation tasks that we aim to support with *Graph Thumbnails* are relevant for PPI analysis.

We downloaded both the most reliable core and full data sets from the Database of Interacting Proteins (DIP) [277]. This includes manually curated PPI data for the years

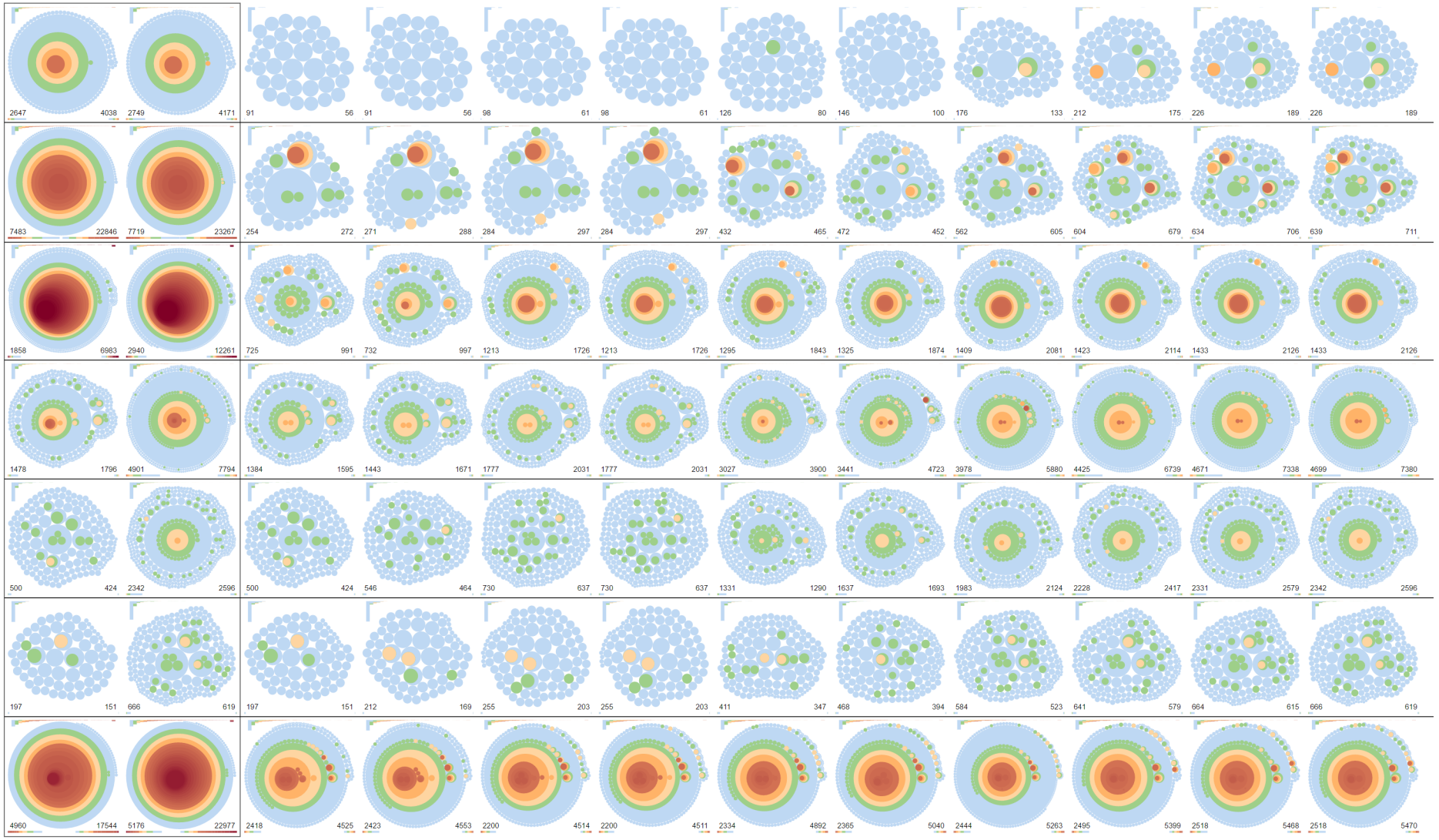


Figure 6.14: Evolution of DIP database structure for seven organisms (*C. elegans*, *D. melanogaster*, *E. coli*, *H. sapiens*, *M. musculus*, *R. norvegicus*, *S. cerevisiae*), each row shows data for one organism (from top to bottom in the listed order). The two leftmost columns show the full DIP dataset for the years 2008 and 2017, respectively. The remaining columns show the high-confidence core dataset (the most reliable subset of the interactions) for the years 2008–2017 (Note that at the time of retrieval, the full and the core data set for mouse and rat were the same for the years 2008 and 2017, while the sizes given on the DIP web page differed).

2008–2017 for seven species (last release for each year), including several model organisms, which have long been the focus of scientific research. Comparing those networks, both across species and across years, can be supported by a representation with a focus on the connectivity and hierarchical density, as those features show both the basic structural characteristics of PPI and the change in the coverage of those interactions by the database. We can compare the required numbers of years and species using small-multiples (**R2**), where each thumbnail will indicate a hierarchical structure (or its absence due to the lack of confirmed evidence) that biologists could also glean from force-directed layouts (**R1**). While ultimately some 'drill-down' facility is required to obtain information on the properties of individual proteins and interactions, the depth of the hierarchy and its components, as well as their size, allows analysts to judge differences in the PPI networks due either to: database updates over the years; differences in these updates depending on the species; or general differences in the species' networks (**R3,R4**). As PPI knowledge is continually growing, a solution that supports such a comparison also needs to scale to the large networks stored in PPI databases (**R5**).

There are a wealth of publications using topological features like degree distribution, clustering coefficient, and average shortest path for characterisation and comparison of PPI networks. These numbers alone can, however, vary for biologically-similar networks or be very close for biologically-differing networks. In addition, it is a tedious process to understand their interplay and to interpret the impact of their combination for comparison purposes, in particular for larger sets of networks. Proteins can also work together in complexes, which might overlap and can be hierarchically organised. These features will be difficult to detect in a standard node-link diagram. We thus investigated the applicability of the *Graph Thumbnail* representation for analysis of the evolution of PPI networks for a range of species.

The *Graph Thumbnail* depiction of the DIP data is shown in Figure 6.14. It clearly shows the development of the connectivity in the core set for most organisms, in contrast to the small changes in the connectivity for the full set, where often a monolithic structure is present from the earlier years on. We can also distinguish the different levels of connectivity occurring in the different species in the early years, where for the most interesting model organisms (in particular *S. cerevisiae*), large structures with deep cores are already visible, due to the state of research at the time. In addition, we can clearly spot differences in development of different organisms, depending on the amount of research put into the detection of PPI over time, with strong activity for *E. coli*, *H. sapiens*, *M. musculus*, and with *R. norvegicus* having very low coherence. In contrast, *R. norvegicus*' core set comprises nearly the full dataset.

6.8 Limitations

One limitation that arises from the canonical characteristic of *Graph Thumbnails*, is the fact that graphs with similar structure but different content will result in similar *Graph Thumbnail* representations. We feel that for graph browsing applications, canonicity is the more significant goal, since differences between similar representations can be resolved through a more detailed inspection—which, as mentioned previously, we are not trying to replace. By contrast, differing representations of similar graphs might lead to these similarities being missed entirely.

We base our studies on one representation, although there exist other possibilities (some of which we mention in Section 6.3), however, we chose to test *Graph Thumbnails*

based on our iterative design process. We also use one clustering algorithm, while there exist other possible ways. Nonetheless, as we point out, the one we chose has properties that made it favourable over many others - in particular, canonicity and linear running time.

6.9 Conclusion

The *Graph Thumbnail* representation is not a replacement for node-link and matrix representations, which are clearly required for detailed inspection of local structure. However, our results show that in Identification, Comparison and Overview related tasks, detail can be confusing and may be better served by a structural overview, such as provided by *Graph Thumbnails*. Generally, visual comparison of high-level graph structure is understudied. The approach of Study 1 is a step to fill this void. Our results indicate that the *Graph Thumbnail* representation can allow humans to identify graphs with various structures that are also differentiated by the graph landscape metrics. Our first study showed that *Graph Thumbnails* are at least as indicative of the network structure as matrices, and significantly better than node-link diagrams. With a second study, we further evaluated *Graph Thumbnails* by asking the participants to do certain Overview tasks, related to connectivity, density and matching. The results showed that in most cases *Graph Thumbnails* outperformed both matrices and node-link diagrams.

Chapter 7

Conclusion

«Ցտեսութիւն ընկերներ,
Ահա օրն է մթներ,
Յաջորդ մեր հանդիպումին,
Հարցեր ունինք պակաւին:»

Վարուժան Արք. Հերկելեան՝
Կենդանիներու Ժողով

An Armenian hymn for children:
Animals' Meeting,
written by Varoujan Hergelian.

Translates into "Until we meet again friends, this meeting is concluded, at our next encounter, we still have open questions."

Networks are sets of interconnected things, which are better understood when visualised [317]. Node-link diagrams are one of the most popular methods for visualising networks, however, there are limits to the number of nodes and links that can be shown in node-link diagrams before they begin to overlap. Quality of layout is another major challenge in network visualisation. There are many possible ways to draw the nodes and the links, thus, researchers in the field have spent decades exploring aesthetics and features that enhance the readability of node-link diagrams. Moreover, producing high-quality, clear diagrams becomes more difficult as networks become bigger, denser and more complex.

Most existing layout techniques for networks deal with the computational complexity of drawing networks by using heuristics and multi-stage methods, which are difficult to refine and modify (**CH-1** in Section 1.1). Many of these techniques lead to layouts with unwanted features due to trade-offs between features introduced at different stages (e.g. long edges to avoid crossings), and thus lead to poorer quality, when compared to human generated layouts [191] (**CH-2** in Section 1.1). We proposed a novel layout model called *Ultra-Compact Grid Layout* that favours compactness and is motivated by grid-based typographical layouts.

The main reason why many of the existing methods use heuristics to find reasonable approximations to desired layout aesthetics, is scalability (**CH-2** in Section 1.1). Finding layouts that enhance aesthetics to optimality is computationally expensive. We decided to

pay the computational price and achieve an optimal layout with respect to our proposed aesthetics. Our optimal methods were able to handle small and medium sized networks, producing aesthetically pleasing and compact layouts.

We subsequently extended the *Ultra-Compact Grid Layout* approach to handle larger networks. We employed *Large Neighbourhood Search* meta-heuristics and developed an iterative method that would focus on small neighbourhoods of the network at one time and produce 'near-optimal' (locally optimal) layouts according to some additional restrictions. This method allowed us to create diagrams for networks based on the *Ultra-Compact Grid Layout*, with up to 100 nodes in under five minutes.

Ultra-Compact Grid Layout: The first major contribution of this thesis is a novel grid-based layout model (Chapter 3).

- We showed the adaptivity of the *Ultra-Compact Grid Layout* model and its usefulness to show hierarchical, grouped, directed and other types of networks, via usage cases.
- We compared the performance of three generic optimisation solvers when solving the network layout problem to achieve an optimal *Ultra-Compact Grid Layout*.
  - The SAT solver performed the best and was able to find optimal layouts for networks with around 60 nodes in five minutes.
- We proposed a heuristic version of the *Ultra-Compact Grid Layout* model based on the *Large Neighbourhood Search* meta-heuristics (LNS), that can produce near-optimal layouts for networks with 100 nodes in under five minutes.
- Force-based grid layouts had poorer quality than layouts created using our LNS approach, which in turn, had poorer quality compared the optimal *Ultra-Compact Grid Layout*.

An obvious question arising from exploring the scalability of our approach for creating optimal and near-optimal visualisations for networks based on the *Ultra-Compact Grid Layout* was, what is a reasonable size for networks that need to or can be visually represented in detail?

We conducted a survey on empirical studies that have used node-link diagrams, in the hopes of finding limits that previous researchers have encountered in terms of network size. We also looked for other factors that might affect the complexity of the visualisations and limit their comprehension. We discovered, through the survey, that tasks which required a detailed understanding of the network are often performed on networks with 100 nodes or less, thus, showing that the use of our *Ultra-Compact Grid Layout* is indeed practical. Another important discovery through the survey was that large networks are mostly used when interactive techniques were involved in the evaluation, and the tasks did not require a detailed understanding of the network. The survey is presented in detail in Chapter 4.

The studies examined in the survey did not reveal limits of network size that hinder human comprehension. Only a few studies explored these limits explicitly, but they also did not report a threshold beyond which detailed node-link diagrams become too difficult to understand (**CH-3** in Section 1.1). Thus, we conducted our own study. We showed the participants node-link diagrams of networks with different sizes and asked them to find the

shortest path between two highlighted nodes. In addition to performance and subjective measures, we used physiological measures such as electrical activity of the brain, pupil dilation and heart rate variability, to study the effects of network size on cognitive load. The results showed that node-link diagrams become difficult and inefficient for networks beyond 50 nodes and a density of 4 when performing path-finding tasks. The user study and our findings are discussed in details in Chapter 5.

The second major contribution of the thesis is the exploration of how network size relates to the complexity of network visualisation.

- We identified four categories of network size with respect to number of nodes.
  - **Small** networks have 20 nodes or less.
  - **Medium** networks have more than 20 nodes and 50 nodes or less.
  - **Large** networks have more than 50 nodes and 200 nodes or less.
  - **Very large** networks have more than 200 nodes.
- We identified three categories of networks with respect to density.
  - **Sparse** networks have densities of more than 1 and densities of 2 or less.
  - **Dense** networks have densities of more than 2 and densities of 4 or less.
  - **Very dense** networks have densities of more than 4.
- Tasks that require a detailed understanding of the network are often performed on small networks.
- Tasks that are often performed on large networks require only a high-level understanding of the overall network structure.
- We collated 152 studies into a survey and provided on-line, interactive visualisations to explore them.
- We discovered, through a user study, that the efficiency of node-link diagrams significantly deteriorates when finding shortest paths in scale-free networks with more than 50 nodes and densities of more than 4.
- We identified the effects of visual factors, such as number of crossings in node-link diagrams on performance and cognition, when finding shortest paths.
- We identified the effects of intrinsic factors, such as the number of nodes and density of networks on performance and cognition, when finding shortest paths using node-link diagrams.

If our *Ultra-Compact Grid Layout* can be used for small networks, a summary representation that provides high-level insight into the structure of the network is needed for large networks (**CH-4** in Section 1.1). We proposed a decomposition of the network that produces a hierarchical tree with many fewer elements than the number of nodes. The decomposition is based on the connectivity structure of the network, thus a good prospect for a summary. We explored the design space to visualise this summary, ended up choosing packed circles, and called the representation *Graph Thumbnails*. We conducted user

studies to compare *Graph Thumbnails* to node-link diagrams and adjacency matrices, in terms of their efficiency, to support overview tasks. We also showed a usage case where *Graph Thumbnails* are used to browse through a set of large protein-protein interaction networks and show their evolution. We discussed these in more detail in Chapter 6.

Graph Thumbnails: As a third major contribution of this thesis we proposed a novel, small, icon-like summary representation of network structure (Chapter 6).

- We compared *Graph Thumbnails* to node-link diagrams and adjacency matrices with respect to their efficiency to support overview tasks.
  - *Graph Thumbnails* were significantly more accurate in identifying similarities and differences between networks compared to node-link diagrams.
  - Subjective feedback showed a preference for *Graph Thumbnails* over node-link diagrams and adjacency matrices.
  - Adjacency matrices were significantly more accurate in identifying similarities and differences between networks compared to node-link diagrams.
  - We discovered that *Graph Thumbnails* are more intuitive and can reveal structural information that relates to node-link diagrams, compared to adjacency matrices (Section 6.6).
- We showed, through a usage case, that *Graph Thumbnails* are useful in displaying summaries of large networks and providing insight into the evolution of large networks, such as protein-protein interaction networks (Section 6.7).

7.1 Future Work

A continuation of the work discussed in this thesis could be to combine the optimal representation based on the *Ultra-Compact Grid Layout* with the *Graph Thumbnail* summary representation. The *Graph Thumbnail* summary representation could act as an overview, whereas the detailed diagram (possibly based on the *Ultra-Compact Grid Layout*), could be revealed through interactive navigation performed via the overview. For example, hovering over a particular cluster in one thumbnail could highlight the locations of similar nodes in thumbnails for other graphs in the set, or to use thumbnails to provide an overview in an interactive graph-browsing scenario, similar to one presented by Dwyer *et al.* [107]. A smaller neighbourhood of interest could be shown in a detailed high-quality view, while the cluster memberships of the nodes in the neighbourhood could be indicated through highlights in the thumbnail overview. The choice between the optimal and heuristic layout models could be decided based on the size of the sub-network being explored, and could be switched interactively. Figure 7.1 shows a basic sketch of a sub-network revealed on a *Graph Thumbnail*, and which could develop into an interactive network visualisation tool. This combination would help make the optimisation model for the *Ultra-Compact Grid Layout* more constrained and might make it easier to be solved, since the placement model will be restricted by the clusters of the *Graph Thumbnail*.

Even though visual representations that are created based on the *Ultra-Compact Grid Layout* are optimal with respect to compactness and features included in the objective function, the routing is not optimal. Nonetheless, since routing highly depends on the placement of the nodes, the resulting layouts are of high quality. Figure 3.12 shows an

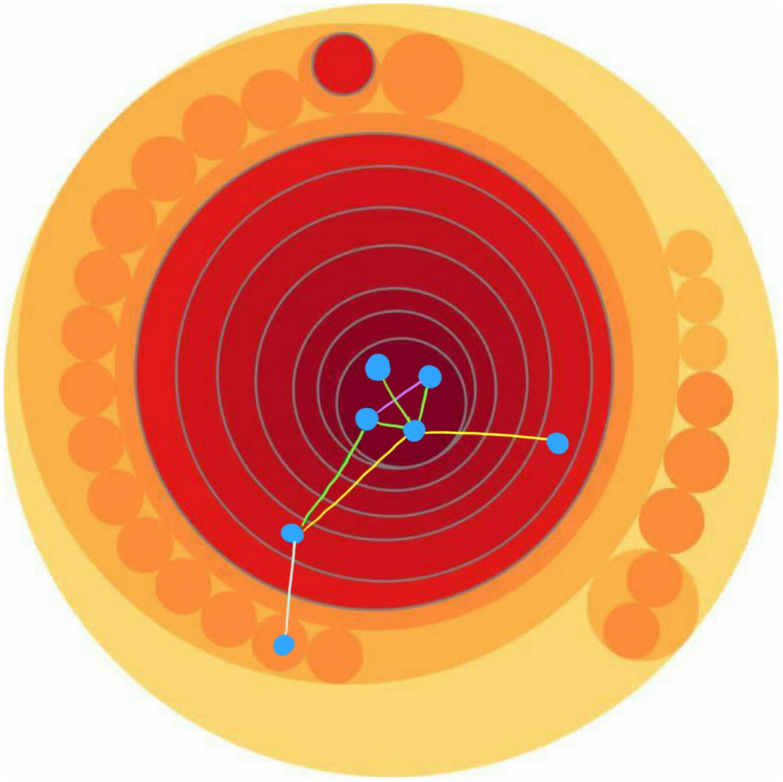


Figure 7.1: A sketch of a possible application of a *Graph Thumbnail* representation with detailed network information.

example in comparing some aesthetics in a diagram based on a force-based layout, the near-optimal *Ultra-Compact Grid Layout* with LNS, and the optimal *Ultra-Compact Grid Layout*. Even though, the *Ultra-Compact Grid Layout* model does not explicitly minimise the number of crossings, the resulting layouts have very few crossings. As future work, it would be interesting to incorporate decision variables for routing in the layout model, and evaluate the resulting layouts.

An interesting direction for our heuristic approach, discussed in Chapter 3, might be to vary the neighbourhood selection for relaxation. We believe that the neighbourhood selection has an important effect on the level of improvement and better selection criteria could lead to layouts with higher quality, while maintaining the same computational complexity. Interviews with experts could identify possible criteria that could guide the neighbourhood selection. Another possibility is to allow the user to interactively select areas of interest as neighbourhoods, thus, involving a human in the loop. This could be useful in diagram creation or in editing scenarios.

In our investigation of the effects of network size on cognitive load in Chapter 5, we limited ourselves to using only one type of task. It would be interesting to investigate the effects of network size on cognitive load with respect to other tasks also. It would also be interesting to find a lower bound on cognitive load, below which tasks become too repetitive, and as a result, might lead to decreased accuracy due to difficulty maintaining focus.

A possible future direction for the *Graph Thumbnail* representation is to experiment with circle packing algorithms, such that the layout can convey other aspects of graph structure. Some possibilities for adjusting, like scale and rotation of components due to

metrics, are relatively easy. More sophisticated packing, such as trying to bring components closer together based on their inter-connectivity, is also possible, but the problem quickly devolves into a similar level of complexity to force-directed layout. Another possibility would be to obtain more fitted non-circular cluster boundaries that more accurately convey the total cluster size with area. Voronoi treemaps [50] might be a starting point for this.

7.2 Closing Remarks

The motivation behind this work was to visualise both small and large networks using neat and aesthetically pleasing diagrams. I achieved this by creating diagrams for small networks based on the *Ultra-Compact Grid Layout* and *Graph Thumbnails* for large networks. I also explored and identified what size of networks should be identified as small versus large and sparse versus dense.

In general, the ease of experimenting with different layouts through simple edits to the declarative model described in Section 3.4, opens up a world of possibilities that should be explored before embarking on engineering faster heuristics. A significant open challenge is a layout model that somehow incorporates routing in a way that is efficiently solvable to optimality.

There are many applications, such as the *yEd Graph Editor* and the *Graph Visualization Software* (*Graphviz*), and libraries and packages, such as *ogdf*, *networkx*, *d3.js*, *cola.js*, and *Cytoscape.js*, for creating network diagrams, however, it is still difficult to find many of the state-of-the-art layout methods in one place or sometimes even at all. I believe, there is a need in the network visualisation community to create a common platform and encourage researchers to upload their proposed layout methods onto this platform, in addition to publishing in well known venues.

I hope that my research paves the way for future research in identifying guidelines for useful sizes of networks for different tasks. These guidelines could inform the design of future network visualisation methods. Clustering algorithms for network data could also be designed to reduce networks to fewer elements based on these guidelines.

List of Publications

Publications arising from this thesis include:

Yoghourdjian, V., Dwyer, T., Gange, G., Kieffer, S., Klein, K., & Marriott, K. (2016). High-Quality Ultra-Compact Grid Layout of Grouped Networks. IEEE Transactions on Visualization and Computer Graphics, 22(1), 339-348.

Yoghourdjian, V., Dwyer, T., Klein, K., Marriott, K., & Wybrow, M. (2018). Graph Thumbnails: Identifying and Comparing Multiple Graphs at a Glance. IEEE Transactions on Visualization and Computer Graphics.

Yoghourdjian, V., Archambault, D., Diehl, S., Dwyer, T., Klein, K., Purchase, H. C., & Wu, H. Y. Exploring the Limits of Complexity: A Survey of Empirical Studies on Graph Visualisation. Information Visualization. *Submitted*.

Yoghourdjian, V., Dwyer, T., Marriott, K., & Wybrow, M. Scalability of Graph Visualisation from a Cognitive Load Perspective. IEEE Transactions on Visualization and Computer Graphics. *Submitted*.

Appendix A

CP Model

```
include "globals.mzn";

   int: nv; // number of vertices
   set of int: vertices = 1..nv;
   constraint assert(nv >= 2, "need at least two vertice");
   array[vertices] of 0..maxwidth : swd; // specified width &
      height
   array[vertices] of 0..maxheight : sht;
   array[vertices] of var 1..maxwidth: wd; // calculated width
      & height
   array[vertices] of var 1..maxheight: ht;
   constraint (forall (u in vertices where swd[u] != 0)(wd[u] =
      swd[u]));
   constraint (forall (u in vertices where sht[u] != 0)(ht[u] =
      sht[u]));
   // desired (Manhattan) distance and weight between each pair
      of vertices
   array[vertices,vertices] of 0..(maxwidth+maxheight): dd;
   array[vertices,vertices] of int: ddw;
   constraint
   assert(forall(u,v in vertices where u!=v)(dd[u,v] >= 0), "dd
      should be >=0");
   constraint
   assert(forall(u,v in vertices where u!=v)(ddw[u,v] >= 0),
      "ddw should be >=0");
   // containment matrix for each pair of vertices
   array[vertices,vertices] of bool : containment;
   // non-overlap matrix for each pair of vertices : 0 if they
      should not overlap and 1 if they are allowed
   array[vertices,vertices] of bool : noverlap;
   int: maxwidth; // maximum starting width of grid
   int: maxheight; // maximum starting height of grid
   var int : maxX;
   var int : maxY;
   constraint 0 <= maxX /\ maxX <= maxwidth;
   constraint 0 <= maxY /\ maxY <= maxheight;
   // core decision variables
```

```
// vertex position
array[vertices] of var 0..maxwidth: xs;
array[vertices] of var 0..maxwidth: xf;
array[vertices] of var 0..maxheight: ys;
array[vertices] of var 0..maxheight: yf;
constraint forall(u in vertices)(xf[u] = xs[u]+wd[u]);
constraint forall(u in vertices)(yf[u] = ys[u]+ht[u]);
// symmetry constraints
int : smallu = min([u | u in vertices where exists(v in
   vertices)(noverlap[u,v])]);
int : smallv = min([v | v in vertices where exists(u in
   vertices)(noverlap[u,v])]);
constraint xs[smallu] <= xs[smallv] /\ ys[smallu] <=
   ys[smallv];
// some vertices should not overlap
constraint
forall(u,v in vertices where u < v /\
   noverlap[u,v])(nonOverlap(u,v));
predicate nonOverlap(vertices: u, vertices:v) =
((xf[u] <= xs[v]) \/ (xf[v] <= xs[u]) \/ (yf[u] <= ys[v]) \/
   (yf[v] <= ys[u]));
// all vertices contained in their parent container
constraint
forall(u in vertices where exists(v in
   vertices)(containment[u,v])) (
xs[u] = min([xs[v]|v in vertices where containment[u,v]]) /\
xf[u] = max([xf[v]|v in vertices where containment[u,v]]) /\
ys[u] = min([ys[v]|v in vertices where containment[u,v]]) /\
yf[u] = max([yf[v]|v in vertices where containment[u,v]]));
// all vertices are in the drawing area
constraint
forall(v in vertices)(
0 <= xs[v] /\ xf[v] <= maxX);
constraint
forall(v in vertices)(
0 <= ys[v] /\ yf[v] <= maxY);
// distance between vertices
array[vertices,vertices] of var 0..maxwidth: xAbsDist;
array[vertices,vertices] of var 0..maxheight: yAbsDist;
constraint
forall(u,v in vertices where u < v) (
xabsdist[u,v]=max([0,xs[v]-xf[u]+1,xs[u]-xf[v]+1]) /\
yAbsDist[u,v]= max([0,ys[v]-yf[u]+1,ys[u]-yf[v]+1]));
// Manhattan stress
array[vertices,vertices] of var 0..(2*(maxwidth+maxheight)):
   absdist;
constraint
forall(u,v in vertices where u < v) (
absdist[u,v]=abs(xAbsDist[u,v]+yAbsDist[u,v]-dd[u,v]));
var int stress=sum(u,v in
   vertices)((ddw[u,v])*(absdist[u,v]));
```

```
var int size=sum(u in vertices)(wd[u]+ht[u]);
var int m=(4*stress)+(1*size)+(2*maxX)+(2*maxY);
solve
minimize m;
```

References

[1] 2014 Graph Drawing Conference contest: http://graphdrawing.de/contest2014/topic2.html.

[2] CPLEX: https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer.

[3] d3.js: http://d3js.org.

[4] g.tec: http://gtec.at.

[5] Heart Rate Variability Logger: https://www.marcoaltini.com/blog/heart-rate-variability-logger-app-details.

[6] https://github.com/Vahany/HqUcGl-of-Grouped-Networks.

[7] ILOG Concert Technology: http://www.ibm.com/software/commerce/ optimization/interfaces.

[8] LiveJournal: https://www.livejournal.com/.

[9] MATLAB: https://au.mathworks.com/products/matlab.html.

[10] Rome Graphs: http://www.graphdrawing.org/data/.

[11] Tobii Pro: http://tobiipro.com.

[12] Tutorial slides for Study 2: http://www.vahany.com/gt/study2/tutorial/.

[13] webcola: http://marvl.infotech.monash.edu/webcola.

[14] yEd Graph Editor: https://www.yworks.com/products/yed.

[15] yFiles automatic network layout software: http://www.yworks.com/en/products/yfiles/.

[16] J. Abello, F. Van Ham, and N. Krishnan. Ask-graphview: A large scale graph visualization system. *IEEE transactions on visualization and computer graphics*, 12(5):669–676, 2006.

[17] A. Abuthawabeh, F. Beck, D. Zeckzer, and S. Diehl. Finding structures in multi-type code couplings with node-link and matrix visualizations. In *Software Visualization (VISSOFT), 2013 First IEEE Working Conference on*, pp. 1–10. IEEE, 2013.

[18] J.-w. Ahn, C. Plaisant, and B. Shneiderman. A task taxonomy for network evolution analysis. *IEEE transactions on visualization and computer graphics*, 20(3):365–376, 2014.

[19] R. K. Ahuja, Ö. Ergun, J. B. Orlin, and A. P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1-3):75–102, 2002.

[20] R. Albert. Scale-free networks in cell biology. *Journal of cell science*, 118(21):4947–4957, 2005.

[21] C. F. Alexander. *Hymns for little children*. Masters, 1887.

[22] 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, pp. 483–492. ACM, New York, NY, USA, 2013. doi: 10.1145/2470654.2470724

[23] B. Alper, T. Höllerer, J. Kuchera-Morin, and A. Forbes. Stereoscopic highlighting: 2d graph visualization on stereo displays. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2325–2333, December 2011.

[24] B. E. Alper, N. Henry Riche, and T. Hollerer. Structuring the space: A study on enriching node-link diagrams with visual references. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pp. 1825–1834. ACM, New York, NY, USA, 2014. doi: 10.1145/2556288.2557112

[25] J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani. k-core decomposition: A tool for the visualization of large scale networks. *arXiv preprint cs/0504107*, 2005.

[26] J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani. Large scale networks fingerprinting and visualization using the k-core decomposition. *Advances in neural information processing systems*, 18:41, 2006.

[27] R. Amar, J. Eagan, and J. Stasko. Low-level components of analytic activity in information visualization. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, pp. 111–117. IEEE, 2005.

[28] E. W. Anderson, K. C. Potter, L. E. Matzen, J. F. Shepherd, G. A. Preston, and C. T. Silva. A user study of visualization effectiveness using eeg and cognitive load. In *Computer Graphics Forum*, vol. 30, pp. 791–800. Wiley Online Library, 2011.

[29] D. Archambault, T. Munzner, and D. Auber. Topolayout: Multilevel graph layout by topological features. *IEEE transactions on visualization and computer graphics*, 13(2), 2007.

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

[31] D. Archambault and H. C. Purchase. The mental map and memorability in dynamic graphs. In *Visualization Symposium (PacificVis), 2012 IEEE Pacific*, pp. 89–96, 2012. doi: 10.1109/PacificVis.2012.6183578

[32] D. Archambault and H. C. Purchase. Mental map preservation helps user orientation in dynamic graphs. In *Graph Drawing - 20th International Symposium, GD 2012, Redmond, WA, USA, September 19-21, 2012, Revised Selected Papers*, pp. 475–486, 2012. doi: 10.1007/978-3-642-36763-2_42

[33] D. Archambault and H. C. Purchase. The "map" in the mental map: Experimental results in dynamic graph drawing. *Int. Journal of Human-Computer Studies*, 71(11):1044–1055, 2013. doi: 10.1016/j.ijhcs.2013.08.004

[34] D. Archambault and H. C. Purchase. Can animation support the visualisation of dynamic graphs? *Information Sciences*, 330:495–509, 2016. SI Visual Info Communication. doi: 10.1016/j.ins.2015.04.017

[35] D. Archambault and H. C. Purchase. On the effective visualisation of dynamic attribute cascades. *Information Visualization*, 15(1):51–63, 2016. doi: 10.1177/1473871615576758

[36] D. Archambault, H. C. Purchase, and B. Pinaud. Difference map readability for dynamic graphs. In *Graph Drawing - 18th International Symposium, GD 2010, Konstanz, Germany, September 21-24, 2010. Revised Selected Papers*, pp. 50–61, 2010. doi: 10.1007/978-3-642-18469-7_5

[37] D. Archambault, H. C. Purchase, and B. Pinaud. The readability of path-preserving clusterings of graphs. *Comput. Graph. Forum*, 29(3):1173–1182, 2010. doi: 10.1111/j.1467-8659.2009.01683.x

[38] D. Archambault, H. C. Purchase, and B. Pinaud. Animation, small multiples, and the effect of mental map preservation in dynamic graphs. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):539–552, April 2011. doi: 10.1109/TVCG.2010.78

[39] A. Arleo, W. Didimo, G. Liotta, and F. Montecchiani. *A Distributed Multilevel Force-Directed Algorithm*, pp. 3–17. Springer International Publishing, Cham, 2016. doi: 10.1007/978-3-319-50106-2_1

[40] R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. Cardinality networks: a theoretical and empirical study. *Constraints*, 16(2):195–221, 2011.

[41] B. Bach, P. Dragicevic, D. Archambault, C. Hurter, and S. Carpendale. A descriptive framework for temporal data visualizations based on generalized space-time cubes. *Computer Graphics Forum*, 36(6):36–61, 2017. doi: 10.1111/cgf.12804

[42] B. Bach, N. Henry Riche, T. Dwyer, T. Madhyastha, J.-D. Fekete, and T. Grabowski. Small MultiPiles: Piling time to explore temporal patterns in dynamic networks. In *Computer Graphics Forum*, vol. 34, pp. 31–40. Wiley Online Library, 2015.

[43] B. Bach, N. Henry Riche, C. Hurter, K. Marriott, and T. Dwyer. Towards unambiguous edge bundling: Investigating confluent drawings for network visualization. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):541–550, 2017.

[44] B. Bach, E. Pietriga, and J.-D. Fekete. Graphdiaries: Animated transitions and temporal navigation for dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 20(5):740–754, November 2013.

[45] B. Bach, N. H. Riche, C. Hurter, K. Marriott, and T. Dwyer. Towards unambiguous edge bundling: Investigating confluent drawings for network visualization. *IEEE transactions on visualization and computer graphics*, 23(1):541–550, 2017.

[46] A. Baddeley. Working memory. *Science*, 255(5044):556–559, 1992.

[47] J. Bae, T. Helldin, and M. Riveiro. Understanding indirect causal relationships in node-link graphs. In *Computer Graphics Forum*, vol. 36, pp. 411–421. Wiley Online Library, 2017.

[48] J. Bae and B. Watson. Developing and evaluating quilts for the depiction of large layered graphs. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2268–2275, December 2011.

[49] K. Ballweg, M. Pohl, G. Wallner, and T. von Landesberger. Visual similarity perception of directed acyclic graphs: A study on influencing factors. *arXiv preprint arXiv:1709.01007*, 2017.

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

[51] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

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

[53] L. Barth, A. Gemsa, B. Niedermann, and M. Nöllenburg. On the readability of boundary labeling. In *Graph Drawing and Network Visualization - 23rd International Symposium, GD 2015, Los Angeles, CA, USA, September 24-26, 2015, Revised Selected Papers*, pp. 515–527, 2015. doi: 10.1007/978-3-319-27261-0_42

[54] V. Batagelj and M. Zaveršnik. Generalized cores. *arXiv preprint cs/0202039*, 2002.

[55] V. Batagelj and M. Zaversnik. An O(m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*, 2003.

[56] C. Batini, E. Nardelli, and R. Tamassia. A layout algorithm for data flow diagrams. *Software Engineering, IEEE Transactions on*, (4):538–546, 1986.

[57] M. Baur, U. Brandes, M. Gaertler, and D. Wagner. Drawing the as graph in 2.5 dimensions. In *International Symposium on Graph Drawing*, pp. 43–48. Springer, 2004.

[58] J. Beatty, B. Lucero-Wagoner, et al. The pupillary system. *Handbook of psychophysiology*, 2:142–162, 2000.

[59] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. A taxonomy and survey of dynamic graph visualization. *Computer Graphics Forum*, 36(1):133–159, 2017. doi: 10.1111/cgf.12791

[60] M. Behrisch, B. Bach, N. Henry Riche, T. Schreck, and J.-D. Fekete. Matrix reordering methods for table and network visualization. In *Computer Graphics Forum*, vol. 35, pp. 693–716. Wiley Online Library, 2016.

[61] N. Beldiceanu, M. Carlsson, S. Demassey, and T. Petit. Global constraint catalogue: Past, present and future. *Constraints*, 12(1):21–62, 2007.

[62] P. Bertolazzi, G. Di Battista, and W. Didimo. Computing orthogonal drawings with the minimum number of bends. *IEEE Transactions on Computers*, 49(8):826–840, 2000.

[63] G. Betz, C. Doll, A. Gemsa, I. Rutter, and D. Wagner. Column-based graph layouts. In *International Symposium on Graph Drawing*, pp. 236–247. Springer, 2012.

[64] G. Betz, A. Gemsa, C. Mathies, I. Rutter, and D. Wagner. Column-based graph layouts. *Journal of Graph Algorithms and Applications*, 18(5):677–708, 2014. doi: 10.7155/jgaa.00341

[65] A. Bezerianos, F. Chevalier, P. Dragicevic, N. Elmqvist, and J.-D. Fekete. Graphdice: A system for exploring multivariate social networks. In *Computer Graphics Forum*, vol. 29, pp. 863–872. Wiley Online Library, 2010.

[66] T. Biedl, T. Bläsius, B. Niedermann, M. Nöllenburg, R. Prutkin, and I. Rutter. Using ilp/sat to determine pathwidth, visibility representations, and other grid-based graph drawings. In *Graph Drawing*, pp. 460–471. Springer, 2013.

[67] J. Blythe, C. McGrath, and D. Krackhardt. The effect of graph layout on inference from social network data. In *Graph Drawing, Symposium on Graph Drawing, GD '95, Passau, Germany, September 20-22, 1995, Proceedings*, pp. 40–51, 1995. doi: 10.1007/BFb0021789

[68] M. A. Borkin, C. S. Yeh, M. Boyd, P. Macko, K. Z. Gajos, M. Seltzer, and H. Pfister. Evaluation of filesystem provenance visualization tools. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2476–2485, December 2013.

[69] O. Bratfisch et al. Perceived item-difficulty in three tests of intellectual performance capacity. 1972.

[70] S. S. Bridgeman and R. Tamassia. A user study in similarity measures for graph drawing. In *Graph Drawing, 8th International Symposium, GD 2000, Colonial Williamsburg, VA, USA, September 20-23, 2000, Proceedings*, pp. 19–30, 2000. doi: 10.1007/3-540-44541-2_3

[71] M. Burch, G. Andrienko, N. Andrienko, M. Höferlin, M. Raschke, and D. Weiskopf. Visual task solution strategies in tree diagrams. In *Proceedings of the 6th IEEE Pacific Visualization Symposium (PacificVis 2013)*, pp. 169–176, 2013. doi: 10.1109/PacificVis.2013.6596142

[72] M. Burch, J. Heinrich, N. Konevtsova, M. Höferlin, and D. Weiskopf. Evaluation of traditional, orthogonal, and radial tree diagrams by an eye tracking study. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2440–2448, December 2011.

[73] M. Burch, C. Vehlow, N. Konevtsova, and D. Weiskopf. Evaluating partially drawn links for directed graph edges. In *Graph Drawing - 19th International Symposium, GD 2011, Eindhoven, The Netherlands, September 21-23, 2011, Revised Selected Papers*, pp. 226–237, 2011. doi: 10.1007/978-3-642-25878-7_22

[74] M. Burch and D. Weiskopf. A flip-book of edge-splatted small multiples for visualizing dynamic graphs. In *Proceedings of the 7th International Symposium on Visual Information Communication and Interaction*, p. 29. ACM, 2014.

[75] J. Carmesin, R. Diestel, M. Hamann, and F. Hundertmark. k-blocks: A connectivity invariant for graphs. *SIAM Journal on Discrete Mathematics*, 28(4):1876–1891, 2014. doi: 10.1137/130923646

[76] P. Chandler and J. Sweller. Cognitive load theory and the format of instruction. *Cognition and instruction*, 8(4):293–332, 1991.

[77] C. Chang, B. Bach, T. Dwyer, and K. Marriott. Evaluating perceptually complementary views for network exploration tasks. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 1397–1407. ACM, 2017.

[78] S. Chen, L. Lin, and X. Yuan. Social media visual analytics. *Computer Graphics Forum*, 36(3):563–587, 2017. doi: 10.1111/cgf.13211

[79] M. Chimani, C. Gutwenger, M. Jünger, G. W. Klau, K. Klein, and P. Mutzel. The Open Graph Drawing Framework (OGDF). *Handbook of Graph Drawing and Visualization*, 2011:543–569, 2013.

[80] D. Chivers and P. Rodgers. Octilinear force-directed layout with mental map preservation for schematic diagrams. In *Diagrammatic Representation and Inference - 8th International Conference, Diagrams 2014, Melbourne, VIC, Australia, July 28 - August 1, 2014. Proceedings*, pp. 1–8, 2014. doi: 10.1007/978-3-662-44043-8_1

[81] A. Clauset, M. E. Newman, and C. Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[82] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Rev.*, 51(4):661–703, Nov. 2009. doi: 10.1137/070710111

[83] W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American statistical association*, 79(387):531–554, 1984.

[84] M. Codish, V. Lagoon, and P. J. Stuckey. Logic programming with satisfiability. *Theory and Practice of Logic Programming*, 8(01):121–128, 2008.

[85] M. Codish and M. Zazon-Ivry. Pairwise cardinality networks. In *16th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, vol. 6355 of *LNCS*, pp. 154–172. Springer, 2010.

[86] A. CohÂt'e, B. Liutkus, G. Bailly, J. Eagan, and E. Lecolinet. Schemelens: A content-aware vector-based fisheye technique for navigating large systems diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):330–338, August 2015.

[87] M. X. Cohen. *Analyzing neural time series data: theory and practice*. MIT press, 2014.

[88] M. Cordeil, T. Dwyer, K. Klein, B. Laha, K. Marriott, and B. H. Thomas. Immersive collaborative analysis of network connectivity: Cave-style or head-mounted display? *IEEE Transactions on Visualization and Computer Graphics*, 23(1):441–450, 2017.

[89] M. Coscia, F. Giannotti, and D. Pedreschi. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 4(5):512–546, 2011.

[90] M. Cypko, J. Wojdziak, M. Stoehr, B. Kirchner, B. Preim, A. Dietz, H. Lemke, and S. Oeltze-Jafra. Visual verification of cancer staging for therapy decision support. In *Computer Graphics Forum*, vol. 36, pp. 109–120. Wiley Online Library, 2017.

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

[92] J. Q. Dawson, T. Munzner, and J. McGrenere. A search-set model of path tracing in graphs. *Information Visualization*, 14(4):308–338, 2015. doi: 10.1177/1473871614550536

[93] D. De Waard. *The measurement of drivers' mental workload*. Groningen University, Traffic Research Center Netherlands, 1996.

[94] K. E. DeLeeuw and R. E. Mayer. A comparison of three measures of cognitive load: Evidence for separable measures of intrinsic, extraneous, and germane load. *Journal of educational psychology*, 100(1):223, 2008.

[95] F. Della Croce. *Mixed Integer Linear Programming Models for Combinatorial Optimization Problems*, pp. 101–133. John Wiley & Sons, Inc., 2014. doi: 10.1002/9781119005216.ch5

[96] A. Delorme and S. Makeig. Eeglab: an open source toolbox for analysis of single-trial eeg dynamics including independent component analysis. *Journal of neuroscience methods*, 134(1):9–21, 2004.

[97] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry*, 4(5):235–282, 1994.

[98] G. Di Battista and R. Tamassia. Incremental planarity testing. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pp. 436–441. IEEE, 1989.

[99] G. Di Battista and R. Tamassia. On-line maintenance of triconnected components with spqr-trees. *Algorithmica*, 15(4):302–318, 1996.

[100] E. Di Giacomo, W. Didimo, G. Liotta, F. Montecchiani, and I. G. Tollis. Exploring complex drawings via edge stratification. In *International Symposium on Graph Drawing*, pp. 304–315. Springer, 2013.

[101] W. Didimo, E. M. Kornaropoulos, F. Montecchiani, and I. G. Tollis. A visualization framework and user studies for overloaded orthogonal drawings. In *Computer Graphics Forum*, vol. 37, pp. 288–300. Wiley Online Library, 2018.

[102] W. Didimo and F. Montecchiani. Fast layout computation of clustered networks: Algorithmic advances and experimental analysis. *Information Sciences*, 260:185–199, 2014.

[103] 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, pp. 3247–3256. ACM, New York, NY, USA, 2013. doi: 10.1145/2470654.2466444

[104] T. Dwyer, N. Henry Riche, K. Marriott, and C. Mears. Edge compression techniques for visualization of dense directed graphs. *IEEE transactions on visualization and computer graphics*, 19(12):2596–2605, 2013.

[105] T. Dwyer, Y. Koren, and K. Marriott. Ipsep-cola: An incremental procedure for separation constraint layout of graphs. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):821–828, 2006.

[106] T. Dwyer, B. Lee, D. Fisher, K. I. Quinn, P. Isenberg, G. Robertson, and C. North. A comparison of user-generated and automatic graph layouts. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):961–968, November/December 2009.

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

[108] T. Dwyer, K. Marriott, and P. J. Stuckey. Fast node overlap removal. In *Graph Drawing*, vol. 3843 of *LNCS*, pp. 153–164. Springer, 2006.

[109] T. Dwyer, C. Mears, K. Morgan, T. Niven, K. Marriott, and M. Wallace. Improved optimal and approximate power graph compression for clearer visualisation of dense graphs. In *Visualization Symposium (PacificVis), 2014 IEEE Pacific*, pp. 105–112. IEEE, 2014.

[110] T. Dwyer, N. H. Riche, K. Marriott, and C. Mears. Edge compression techniques for visualization of dense directed graphs. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2596–2605, December 2013. doi: 10.1109/TVCG.2013.151

[111] P. Eades. A heuristic for graph drawing. *Congressus numerantium*, 42:149–160, 1984.

[112] P. Eades, S. Hong, A. Nguyen, and K. Klein. Shape-based quality metrics for large graph visualization. *Journal of Graph Algorithms and Applications*, 21(1):29–53, 2017. doi: 10.7155/jgaa.00405

[113] P. Eades and M. L. Huang. Navigating clustered graphs using force-directed methods. In *Graph Algorithms And Applications 2*, pp. 191–215. World Scientific, 2004.

[114] P. Eades, B. D. McKay, and N. C. Wormald. On an edge crossing problem. In *Proc. 9th Australian Computer Science Conference*, vol. 327, p. 334, 1986.

[115] N. Eén and N. Sörensson. Translating pseudo-boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:1–26, 2006.

[116] S. G. Eick and A. F. Karr. Visual scalability. *Journal of Computational and Graphical Statistics*, 11(1):22–43, 2002.

[117] M. Eiglsperger, M. Siebenhaller, and M. Kaufmann. An efficient implementation of sugiyamaâĂŹs algorithm for layered graph drawing. In *International Symposium on Graph Drawing*, pp. 155–166. Springer, 2004.

[118] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull. GraphvizâĂŤopen source graph drawing tools. In *International Symposium on Graph Drawing*, pp. 483–484. Springer, 2001.

[119] N. Elmqvist and J.-D. Fekete. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Trans. on Visualization and Computer Graphics*, 16(3):439–454, 2010.

[120] R. W. Engle. Working memory capacity as executive attention. *Current directions in psychological science*, 11(1):19–23, 2002.

[121] O. Ersoy, C. Hurter, F. Paulovich, G. Cantareiro, and A. Telea. Skeleton-based edge bundling for graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2364–2373, 2011.

[122] L. Euler. Solutio problematis ad geometriam situs pertinensis. *Comm. Acad. Sci. Imper. Petropol.*, 8:128–140, 1736.

[123] H. Ezaiza, S. R. Humayoun, R. AlTarawneh, and A. Ebert. Person-vis: Visualizing personal social networks (ego networks). In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pp. 1222–1228. ACM, 2016.

[124] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM computer communication review*, vol. 29, pp. 251–262. ACM, 1999.

[125] I. Farkas, I. Derényi, H. Jeong, Z. Neda, Z. Oltvai, E. Ravasz, A. Schubert, A.-L. Barabási, and T. Vicsek. Networks in life: Scaling properties and eigenvalue spectra. *Physica A: Statistical Mechanics and its Applications*, 314(1-4):25–34, 2002.

[126] M. Farrugia and A. Quigley. Effective temporal graph layout: A comparative study of animation versus static display methods. *Information Visualization*, 10(1):47–64, 2011. doi: 10.1057/ivs.2010.10

[127] P. Federico and S. Miksch. Evaluation of two interaction techniques for visualization of dynamic graphs. In *Graph Drawing and Network Visualization - 24th International Symposium, GD 2016, Athens, Greece, September 19-21, 2016, Revised Selected Papers*, pp. 557–571, 2016. doi: 10.1007/978-3-319-50106-2_43

[128] J.-D. Fekete. Reorder.js: A JavaScript library to reorder tables and networks. In *IEEE VIS 2015*, 2015.

[129] T. Feydy, A. Schutt, and P. Stuckey. Semantic learning for lazy clause generation. In *TRICS workshop, held alongside CP*, 2013.

[130] C. J. Fisk and D. Isett. âĂIJaccelâĂİ automated circuit card etching layout. In *Proceedings of the SHARE design automation project*, pp. 9–1. ACM, 1965.

[131] M. Freire, C. Plaisant, B. Shneiderman, and J. Golbeck. Manynets: an interface for multiple network analysis and visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 213–222. ACM, 2010.

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

[133] G. Gange, P. J. Stuckey, and K. Marriott. Optimal k-level planarization and crossing minimization. In *International Symposium on Graph Drawing*, pp. 238–249. Springer, 2010.

[134] E. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In *Graph Drawing*, vol. 3383 of *LNCS*, pp. 239–250. Springer, 2005. doi: 10.1007/978-3-540-31843-9_25

[135] E. R. Gansner, Y. Hu, and S. Kobourov. Gmap: Visualizing graphs and clusters as maps. In *Visualization Symposium (PacificVis), 2010 IEEE Pacific*, pp. 201–208. IEEE, 2010.

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

[137] M. Gendreau and J.-Y. Potvin. *Handbook of metaheuristics*, vol. 2. Springer, 2010.

[138] S. Ghani, N. Elmqvist, and J. S. Yi. Perception of animated node-link diagrams for dynamic graphs. *Computer Graphics Forum*, 31(3):1205–1214, 2012. doi: 10.1111/j.1467-8659.2012.03113.x

[139] S. Ghani, N. H. Riche, and N. Elmqvist. Dynamic insets for context-aware graph navigation. *Computer Graphics Forum*, 30(3):861–870, 2011. doi: 10.1111/j.1467-8659.2011.01935.x

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

[141] M. Ghoniem, J.-D. Fekete, and P. Castagliola. On the readability of graphs using node-link and matrix-based representations: A controlled experiment and statistical analysis. *Information Visualization*, 4(2):114–135, 2005. doi: 10.1057/palgrave.ivs.9500092

[142] E. D. Giacomo, W. Didimo, L. Grilli, and G. Liotta. Graph visualization techniques for web clustering engines. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):294–304, March/April 2007. doi: 10.1109/TVCG.2007.40

[143] E. D. Giacomo, W. Didimo, G. Liotta, F. Montecchiani, and I. G. Tollis. Exploring complex drawings via edge stratification. In *Graph Drawing - 21st International Symposium, GD 2013, Bordeaux, France, September 23-25, 2013, Revised Selected Papers*, pp. 304–315, 2013. doi: 10.1007/978-3-319-03841-4_27

[144] C. Giatsidis, F. D. Malliaros, N. Tziortziotis, C. Dhanjal, E. Kiagias, D. M. Thilikos, and M. Vazirgiannis. A k-core decomposition framework for graph clustering. *arXiv preprint arXiv:1607.02096*, 2016.

[145] C. Giatsidis, D. M. Thilikos, and M. Vazirgiannis. Evaluating cooperation in communities with the k-core structure. In *2011 International Conference on Advances in Social Networks Analysis and Mining*, pp. 87–93, July 2011. doi: 10.1109/ASONAM.2011.65

[146] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[147] N. Greffard, F. Picarougne, and P. Kuntz. Visual community detection: An evaluation of 2d, 3d perspective and 3d stereoscopic displays. In *Graph Drawing - 19th International Symposium, GD 2011, Eindhoven, The Netherlands, September 21-23, 2011, Revised Selected Papers*, pp. 215–225, 2011. doi: 10.1007/978-3-642-25878-7_21

[148] J. L. Gross, J. Yellen, and P. Zhang. *Handbook of graph theory*. Chapman and Hall/CRC, 2013.

[149] H. Guo, J. Huang, and D. H. Laidlaw. Representing uncertainty in graph edges: An evaluation of paired visual variables. *IEEE Transactions on Visualization and Computer Graphics*, 21(10):1173–1186, October 2015. doi: 10.1109/TVCG.2015.2424872

[150] C. Gutwenger and P. Mutzel. *A Linear Time Implementation of SPQR-Trees*, pp. 77–90. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. doi: 10.1007/3-540-44541-2_8

[151] S. Hachul and M. Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In *International Symposium on Graph Drawing*, pp. 285–295. Springer, 2004.

[152] R. Hadany and D. Harel. A multi-scale algorithm for drawing graphs nicely. *Discrete Applied Mathematics*, 113(1):3–21, 2001.

[153] S. Hadlak, H. Schumann, and H.-J. Schulz. A survey of multi-faceted graph visualization. In *Eurographics Conference on Visualization (EuroVis) - STARs*, 2015.

[154] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pp. 11–15. Pasadena, CA USA, Aug. 2008.

[155] D. Harel and M. Sardas. Randomized graph drawing with heavy-duty preprocessing. *Journal of Visual Languages & Computing*, 6(3):233–253, 1995.

[156] N. Henry, A. Bezerianos, and J. Fekete. Improving the readability of clustered social networks using node duplication. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1317–1324, November/December 2008. doi: 10.1109/TVCG.2008.141

[157] I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Trans. on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[158] M. Himsolt. Comparing and evaluating layout algorithms within graphed. *Journal of Visual Languages and Computing*, 6(3):255–273, 1995.

[159] M. Hlawatsch, M. Burch, and D. Weiskopf. Visual adjacency lists for dynamic graphs. *IEEE Transactions on Visualization and Computer Graphics*, 20(11):1590–1603, November 2014.

[160] D. Holten, P. Isenberg, J. J. van Wijk, and J.-D. Fekete. An extended evaluation of the readability of tapered, animated, and textured directed-edge representations in node-link graphs. In *Visualization Symposium (PacificVis), 2011 IEEE Pacific*, pp. 195–202, 2011. doi: 10.1109/PACIFICVIS.2011.5742390

[161] D. Holten and J. J. van Wijk. A user study on visualizing directed edges in graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2299–2308. ACM, 2009.

[162] J. E. Hopcroft, J. D. Ullman, and A. Aho. The design and analysis of computer algorithms, 1975.

[163] E. Hoque and G. Carenini. Interactive topic hierarchy revision for exploring a collection of online conversations. *Information Visualization*, p. 1473871618757228, 2018.

[164] Y. Hu. Efficient, high-quality force-directed graph drawing. *Mathematica Journal*, 10(1):37–71, 2005.

[165] Y. Hu and L. Shi. Visualizing large graphs. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(2):115–136, 2015.

[166] Y. Hu, L. Shi, and Q. Liu. A coloring algorithm for disambiguating graph and map drawings. *IEEE Transactions on Visualization and Computer Graphics*, 2018.

[167] W. Huang. Using eye tracking to investigate graph layout effects. In *Visualization, 2007. APVIS'07. 2007 6th International Asia-Pacific Symposium on*, pp. 97–100. IEEE, 2007.

[168] W. Huang, P. Eades, and S.-H. Hong. A graph reading behavior: Geodesic-path tendency. In *Proceedings of the 2th IEEE Pacific Visualization Symposium (PacificVis 2009)*, pp. 137–144, 2009. doi: 10.1109/PACIFICVIS.2009.4906848

[169] W. Huang, P. Eades, and S.-H. Hong. Measuring effectiveness of graph visualizations: A cognitive load perspective. *Information Visualization*, 8(3):139–152, 2009. doi: 10.1057/ivs.2009.10

[170] W. Huang, P. Eades, and S.-H. Hong. Measuring effectiveness of graph visualizations: A cognitive load perspective. *Information Visualization*, 8(3):139–152, 2009.

[171] W. Huang, P. Eades, S.-H. Hong, and H. B.-L. Duh. Effects of curves on graph perception. In *Pacific Visualization Symposium (PacificVis), 2016 IEEE*, pp. 199–203. IEEE, 2016.

[172] W. Huang, P. Eades, S.-H. Hong, and C.-C. Lin. Improving force-directed graph drawings by making compromises between aesthetics. In *Visual Languages and Human-Centric Computing (VL/HCC), 2010 IEEE Symposium on*, pp. 176–183. IEEE, 2010.

[173] W. Huang, S.-H. Hong, and P. Eades. Layout effects on sociogram perception. In *Graph Drawing, 13th International Symposium, GD 2005, Limerick, Ireland, September 12-14, 2005, Revised Papers*, pp. 262–273, 2005. doi: 10.1007/11618058_24

[174] W. Huang, S.-H. Hong, and P. Eades. Effects of crossing angles. In *Visualization Symposium, 2008. PacificVIS '08. IEEE Pacific*, pp. 41–46, 2008. doi: 10.1109/PACIFICVIS.2008.4475457

[175] W. Huang and M. L. Huang. Exploring the relative importance of crossing number and crossing angle. In *2010 International Symposium on Visual Information Communication, VINCI '10, Beijing, China - September 28 - 29, 2010*, p. 10, 2010. doi: 10.1145/1865841.1865854

[176] X. Huang, Y. Zhao, C. Ma, J. Yang, X. Ye, and C. Zhang. Trajgraph: A graph-based visual analytics approach to studying urban network centralities using taxi trajectory data. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):160–169, January 2016. doi: 10.1109/TVCG.2015.2467771

[177] V. Hugo. *Les misérables*. First Avenue Editions, 2015.

[178] T. Jankun-Kelly, T. Dwyer, D. Holten, C. Hurter, M. Nöllenburg, C. Weaver, and K. Xu. Scalability considerations for multivariate graph visualization. In A. Kerren, H. C. Purchase, and M. O. Ward, eds., *Multivariate Network Visualization LNCS 8380*, chap. 10, pp. 207–235. Springer, 2014.

[179] T. Jankun-Kelly, T. Dwyer, D. Holten, C. Hurter, M. Nöllenburg, C. Weaver, and K. Xu. Scalability considerations for multivariate graph visualization. In *Multivariate Network Visualization*, pp. 207–235. Springer, 2014.

[180] T. Jankun-Kelly and K.-L. Ma. Moiregraphs: Radial focus+ context visualization and interaction for graphs with visual nodes. In *Information Visualization, 2003. INFOVIS 2003. IEEE Symposium on*, pp. 59–66. IEEE, 2003.

[181] R. Jianu, A. Rusu, Y. Hu, and D. Taggart. How to display group information on node-link diagrams: An evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 20(11):1530–1541, November 2014.

[182] H. Kabir and K. Madduri. Parallel k-core decomposition on multicore platforms. In *Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2017 IEEE International*, pp. 1482–1491. IEEE, 2017.

[183] N. R. Kadaba, P. P. Irani, and J. Leboe. Visualizing causal semantics using animations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1254–1261, November/December 2007.

[184] D. Kahneman. *Attention and effort*, vol. 1063. Prentice-Hall Englewood Cliffs, NJ, 1973.

[185] S. Kairam, D. MacLean, M. Savva, and J. Heer. GraphPrism: compact visualization of network structure. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pp. 498–505. ACM, 2012.

[186] T. Kamada, S. Kawai, et al. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15, 1989.

[187] 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*, 5(1):62–76, 2006. doi: 10.1057/palgrave.ivs.9500116

[188] A. Kennedy, K. Klein, and A. Nguyen. The graph landscape – a concept for the visual analysis of graph set properties. In Submission.

[189] A. Kennedy, K. Klein, A. Nguyen, and F. Y. Wang. The graph landscape: using visual analytics for graph set analysis. *Journal of Visualization*, pp. 1–16, 2016. doi: 10.1007/s12650-016-0374-6

[190] S. Kieffer, T. Dwyer, K. Marriott, and M. Wybrow. Incremental grid-like layout using soft and hard constraints. In *International Symposium on Graph Drawing*, pp. 448–459. Springer, 2013.

[191] S. Kieffer, T. Dwyer, K. Marriott, and M. Wybrow. Hola: Human-like orthogonal network layout. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):349–358, August 2015.

[192] S. A. KIEFFER. A Human-Centred Approach to Network Layout Algorithm Design. 5 2017. doi: 10.4225/03/59261996e5d2b

[193] Y. Kim and H. Jeong. Map equation for link communities. *Physical Review E*, 84(2):026110, 2011.

[194] P. Kindermann, W. Meulemans, and A. Schulz. Experimental analysis of the accessibility of drawings with few segments. *arXiv preprint arXiv:1708.09815*, 2017.

[195] G. Klau and P. Mutzel. Optimal compaction of orthogonal grid drawings. *Integer Programming and Combinatorial Optimization*, pp. 304–319, 1999.

[196] W. Klimesch. Eeg alpha and theta oscillations reflect cognitive and memory performance: a review and analysis. *Brain research reviews*, 29(2-3):169–195, 1999.

[197] S. G. Kobourov. Force-directed drawing algorithms. *Handbook of Graph Drawing and Visualization*, pp. 383–408, 2013.

[198] S. G. Kobourov, S. Pupyrev, and B. Saket. Are crossings important for drawing large graphs? In *Graph Drawing - 22nd International Symposium, GD 2014, Würzburg, Germany, September 24-26, 2014, Revised Selected Papers*, pp. 234–245, 2014. doi: 10.1007/978-3-662-45803-7_20

[199] K. Kojima, M. Nagasaki, E. Jeong, M. Kato, and S. Miyano. An efficient grid layout algorithm for biological networks utilizing various biological attributes. *BMC bioinformatics*, 8(1):76, 2007.

[200] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[201] R. E. Korf, M. D. Moffitt, and M. E. Pollack. Optimal rectangle packing. *Annals of Operations Research*, 179(1):261–295, 2010. doi: 10.1007/s10479-008-0463-6

[202] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos. VOG: summarizing and understanding large graphs. In *Proceedings of the 2014 SIAM international conference on data mining*, pp. 91–99. SIAM, 2014.

[203] C. Koylu and D. Guo. Design and evaluation of line symbolizations for origin–destination flow maps. *Information Visualization*, 16(4):309–331, 2017.

[204] J. B. Kruskal and J. M. Landwehr. Icicle plots: Better displays for hierarchical clustering. *The American Statistician*, 37(2):162–168, 1983.

[205] M. Krzywinski, I. Birol, S. J. Jones, and M. A. Marra. Hive plotsâĂŤrational approach to visualizing networks. *Briefings in bioinformatics*, 13(5):627–644, 2011.

[206] O. Kwon, C. Muelder, K. Lee, and K. Ma. A study of layout, rendering, and interaction methods for immersive graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(7):1802–1815, July 2016. doi: 10.1109/TVCG.2016.2520921

[207] O.-H. Kwon, T. Crnovrsanin, and K.-L. Ma. What would a graph look like in this layout? a machine learning approach to large graph visualization. *IEEE transactions on visualization and computer graphics*, 24(1):478–488, 2018.

[208] A. Lancichinetti and S. Fortunato. Community detection algorithms: a comparative analysis. *Physical review E*, 80(5):056117, 2009.

[209] C. Le Pape. *Constraint Programming*, pp. 325–338. John Wiley & Sons, Inc., 2014. doi: 10.1002/9781119005216.ch11

[210] A. Lee and D. Archambault. Communities found by users – not algorithms: Comparing human and algorithmically generated communities. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pp. 2396–2400. ACM, New York, NY, USA, 2016. doi: 10.1145/2858036.2858071

[211] B. Lee, C. S. Parr, C. Plaisant, B. B. Bederson, V. D. Veksler, W. D. Gray, and C. Kotfila. Treeplus: Interactive exploration of networks with enhanced tree layouts. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1414–1426, November/December 2006. doi: 10.1109/TVCG.2006.106

[212] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry Riche. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, pp. 1–5. ACM, 2006.

[213] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[214] A. Lhuillier, C. Hurter, and A. Telea. State of the art in edge and trail bundling techniques. *Computer Graphics Forum*, 36(3):619–645, 2017. doi: 10.1111/cgf.13213

[215] W. Li and H. Kurata. Visualizing global properties of large complex networks. *PLoS One*, 3(7):e2541, 2008.

[216] R. M. Lima and I. E. Grossmann. Computational advances in solving mixed integer linear programming problems. 2011.

[217] D. Liu, F. Guo, B. Deng, H. Qu, and Y. Wu. egocomp: A node-link-based technique for visual comparison of ego-networks. *Information Visualization*, 16(3):179–189, 2017.

[218] G. L. Lohse. The role of working memory on graphical information processing. *Behaviour & Information Technology*, 16(6):297–308, 1997.

[219] P. Lüders, R. Ernst, and S. Stille. An approach to automatic display layout using combinatorial optimization algorithms. *Software: Practice and Experience*, 25(11):1183–1202, 1995.

[220] E. Mäkinen and H. Siirtola. The barycenter heuristic and the reorderable matrix. *Informatica (Slovenia)*, 29(3):357–364, 2005.

[221] M. R. Marner, R. T. Smith, B. H. Thomas, K. Klein, P. Eades, and S. Hong. GION: interactively untangling large graphs on wall-sized displays. In *Graph Drawing - 22nd International Symposium, GD 2014, Würzburg, Germany, September 24-26, 2014, Revised Selected Papers*, pp. 113–124, 2014. doi: 10.1007/978-3-662-45803-7_10

[222] K. Marriott, H. Purchase, M. Wybrow, and C. Goncu. Memorability of visual features in network diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2477–2485, December 2012.

[223] M. J. McGuffin and J.-M. Robert. Quantifying the space-efficiency of 2d graphical representations of trees. *Information Visualization*, 9(2):115–140, 2010.

[224] G. Melancon. Just how dense are dense graphs in the real world?: a methodological note. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, pp. 1–7. ACM, 2006.

[225] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pp. 117–128. IEEE, 2002.

[226] A. Metodi and M. Codish. Compiling finite domain constraints to sat with bee. *Theory and Practice of Logic Programming*, 12(4-5):465–483, 2012.

[227] W. Meulemans and A. Schulz. A tale of two communities: Assessing homophily in node-link diagrams. In *Graph Drawing and Network Visualization - 23rd International Symposium, GD 2015, Los Angeles, CA, USA, September 24-26, 2015, Revised Selected Papers*, pp. 489–501, 2015. doi: 10.1007/978-3-319-27261-0_40

[228] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.

[229] R. Ming, R. VanBuren, C. M. Wai, H. Tang, M. C. Schatz, J. E. Bowers, E. Lyons, M.-L. Wang, J. Chen, E. Biggers, et al. The pineapple genome and the evolution of cam photosynthesis. *Nature genetics*, 47(12):1435, 2015.

[230] T. Moscovich, F. Chevalier, N. Henry, E. Pietriga, and J.-D. Fekete. Topology-aware navigation in large networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pp. 2319–2328. ACM, New York, NY, USA, 2009. doi: 10.1145/1518701.1519056

[231] C. Muelder and K.-L. Ma. A treemap based method for rapid layout of large graphs. In *Visualization Symposium, 2008. PacificVIS'08. IEEE Pacific*, pp. 231–238. IEEE, 2008.

[232] J. Muller-Brockmann. *Grid systems in graphic design: A visual communication manual for graphic designers, typographers and three dimensional designers*. Niederteufen, 1981.

[233] T. Munzner. *Visualization analysis and design*. CRC press, 2014.

[234] D. Nekrasovski, A. Bodnar, J. McGrenere, F. Guimbretière, and T. Munzner. An evaluation of pan & zoom and rubber sheet navigation with and without an overview. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 11–20. ACM, 2006.

[235] B. M. Neta, G. H. Araújo, F. G. Guimarães, R. C. Mesquita, and P. Y. Ekel. A fuzzy genetic algorithm for automatic orthogonal graph drawing. *Applied Soft Computing*, 12(4):1379–1389, 2012.

[236] N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, and G. Tack. Minizinc: Towards a standard cp modelling language. In *International Conference on Principles and Practice of Constraint Programming*, pp. 529–543. Springer, 2007.

[237] R. Netzel, M. Burch, and D. Weiskopf. Comparative eye tracking study on node-link visualizations of trajectories. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2221–2230, December 2014.

[238] M. E. Newman. Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133, 2004.

[239] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[240] Q.-H. Nguyen, S. Hong, P. Eades, and A. Meidiana. Proxy graph: Visual quality metrics of big graph sampling. *Transactions on Visualization and Computer Graphics*, 2017.

[241] M. Nöllenburg. An improved algorithm for the metro-line crossing minimization problem. In *Graph Drawing*, pp. 381–392. Springer, 2010.

[242] M. Nöllenburg and A. Wolff. A mixed-integer program for drawing high-quality metro maps. In *Graph Drawing*, vol. 3843 of *LNCS*, pp. 321–333. Springer, 2006.

[243] C. North, P. Saraiya, and K. Duca. A comparison of benchmark task and insight evaluation methods for information visualization. *Information Visualization*, 10(3):162–181, 2011. doi: 10.1177/1473871611415989

[244] L. R. Novick and K. M. Catley. Interpreting hierarchical structure: Evidence from cladograms in biology. In *Diagrammatic Representation and Inference, 4th International Conference, Diagrams 2006, Stanford, CA, USA, June 28-30, 2006, Proceedings*, pp. 176–180, 2006. doi: 10.1007/11783183_24

[245] M. Okoe, S. S. Alam, and R. Jianu. A gaze-enabled graph visualization to improve graph reading tasks. *Computer Graphics Forum*, 33(3):251–260, 2014. doi: 10.1111/cgf.12381

[246] M. Okoe and R. Jianu. Graphunit: Evaluating interactive graph visualizations using crowdsourcing. *Computer Graphics Forum*, 34(3):451–460, 2015. doi: 10.1111/cgf.12657

[247] M. Okoe, R. Jianu, and S. Kobourov. Revisited network representations. In *25th Symposium on Graph Drawing (GD)*, 2017.

[248] F. Paas, J. E. Tuovinen, H. Tabbers, and P. W. Van Gerven. Cognitive load measurement as a means to advance cognitive load theory. *Educational psychologist*, 38(1):63–71, 2003.

[249] F. G. Paas and J. J. Van Merriënboer. The efficiency of instructional conditions: An approach to combine mental effort and performance measures. *Human factors*, 35(4):737–743, 1993.

[250] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *nature*, 435(7043):814, 2005.

[251] E. M. M. Peck, B. F. Yuksel, A. Ottley, R. J. Jacob, and R. Chang. Using fnirs brain sensing to evaluate information visualization interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 473–482. ACM, 2013.

[252] 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*, 24(1):215–225, 2018.

[253] D. Pisinger and S. Ropke. Large neighborhood search. In *Handbook of metaheuristics*, pp. 399–419. Springer, 2010.

[254] P. Pons and M. Latapy. Computing communities in large networks using random walks. In *International Symposium on Computer and Information Sciences*, pp. 284–293. Springer, 2005.

[255] S. Pupyrev, L. Nachmanson, and M. Kaufmann. Improving layered graph layouts with edge bundling. In *International Symposium on Graph Drawing*, pp. 329–340. Springer, 2010.

[256] H. C. Purchase. Which aesthetic has the greatest effect on human understanding? In *Graph Drawing, 5th International Symposium, GD '97, Rome, Italy, September 18-20, 1997, Proceedings*, pp. 248–261, 1997. doi: 10.1007/3-540-63938-1_67

[257] H. C. Purchase. Performance of layout algorithms: Comprehension, not computation. *Journal of Visual Languages & Computing*, 9(6):647–657, 1998.

[258] H. C. Purchase. Metrics for graph drawing aesthetics. *Journal of Visual Languages & Computing*, 13(5):501–516, 2002.

[259] H. C. Purchase, J. Allder, and D. A. Carrington. User preference of graph layout aesthetics: A UML study. In *Graph Drawing, 8th International Symposium, GD 2000, Colonial Williamsburg, VA, USA, September 20-23, 2000, Proceedings*, pp. 5–18, 2000. doi: 10.1007/3-540-44541-2_2

[260] H. C. Purchase, R. F. Cohen, and M. I. James. Validating graph drawing aesthetics. In *Graph Drawing, Symposium on Graph Drawing, GD '95, Passau, Germany, September 20-22, 1995, Proceedings*, pp. 435–446, 1995. doi: 10.1007/BFb0021827

[261] H. C. Purchase, J. Hamer, M. Nöllenburg, and S. G. Kobourov. On the usability of lombardi graph drawings. In *Graph Drawing - 20th International Symposium, GD 2012, Redmond, WA, USA, September 19-21, 2012, Revised Selected Papers*, pp. 451–462, 2012. doi: 10.1007/978-3-642-36763-2_40

[262] H. C. Purchase, E. E. Hoggan, and C. Görg. How important is the "mental map"? - an empirical investigation of a dynamic graph layout algorithm. In *Graph Drawing, 14th International Symposium, GD 2006, Karlsruhe, Germany, September 18-20, 2006. Revised Papers*, pp. 184–195, 2006. doi: 10.1007/978-3-540-70904-6_19

[263] H. C. Purchase, C. Pilcher, and B. Plimmer. Graph drawing aesthetics - created by users, not algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 18(1):81–92, January 2012. doi: 10.1109/TVCG.2010.269

[264] H. C. Purchase and A. Samra. Extremes are better: Investigating mental map preservation in dynamic graphs. In *Diagrammatic Representation and Inference, 5th International Conference, Diagrams 2008, Herrsching, Germany, September 19-21, 2008. Proceedings*, pp. 60–73, 2008. doi: 10.1007/978-3-540-87730-1_9

[265] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[266] G. Robertson, K. Cameron, M. Czerwinski, and D. Robbins. Animated visualization of multiple intersecting hierarchies. *Information Visualization*, 1(1):50–65, 2002. doi: 10.1057/palgrave.ivs.9500002

[267] P. Rodgers. A survey of euler diagrams. *Journal of Visual Languages & Computing*, 25(3):134–155, 2014.

[268] M. Rohrschneider, C. Heine, A. Reichenbach, A. Kerren, and G. Scheuermann. A novel grid-based visualization approach for metabolic networks with advanced focus&context view. In *Graph Drawing*, pp. 268–279, 2009. doi: 10.1007/978-3-642-11805-0_26

[269] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[270] L. Royer, M. Reimann, B. Andreopoulos, and M. Schroeder. Unraveling protein networks with power graph analysis. *PLoS Comput Biol*, 4(7):e1000108, 2008.

[271] S. Rufiange and M. J. McGuffin. Diffani: Visualizing dynamic graphs with a hybrid of difference maps and animation. *IEEE Transactions on Visualization and Computer Graphics*, 9(12):2556–2565, December 2013.

[272] S. Rufiange, M. J. McGuffin, and C. P. Fuhrman. Treematrix: A hybrid visualization of compound graphs. In *Computer Graphics Forum*, vol. 31, pp. 89–101. Wiley Online Library, 2012.

[273] K. Saito, T. Yamada, and K. Kazama. Extracting communities from complex networks by the k-dense method. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 91(11):3304–3311, 2008.

[274] B. Saket, C. Scheidegger, and S. G. Kobourov. Comparing node-link and node-link-group visualizations from an enjoyment perspective. *Computer Graphics Forum*, 35(3):41–50, 2016. doi: 10.1111/cgf.12880

[275] B. Saket, C. Scheidegger, S. G. Kobourov, and K. Börner. Map-based visualizations increase recall accuracy of data. *Computer Graphics Forum*, 34(3):441–450, 2015. doi: 10.1111/cgf.12656

[276] B. Saket, P. Simonetto, S. Kobourov, and K. Börner. Node, node-link, and node-link-group diagrams: An evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2231–2240, December 2014.

[277] L. Salwinski, C. S. Miller, A. J. Smith, F. K. Pettit, J. U. Bowie, and D. Eisenberg. The database of interacting proteins: 2004 update. *Nucleic acids research*, 32(suppl 1):D449–D451, 2004.

[278] G. Sander. Layout of directed hypergraphs with orthogonal hyperedges. In *Graph Drawing*, vol. 2912 of *LNCS*, pp. 381–386. Springer, 2004. doi: 10.1007/978-3-540-24595-7_35

[279] A. E. Sariyüce and A. Pinar. Fast hierarchy construction for dense subgraphs. *Proceedings of the VLDB Endowment*, 10(3):97–108, 2016.

[280] A. E. Sariyuce, C. Seshadhri, A. Pinar, and U. V. Catalyurek. Finding the hierarchy of dense subgraphs using nucleus decompositions. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pp. 927–937. ACM, 2015. doi: 10.1145/2736277.2741640

[281] R. Saxena, S. Kaur, D. Dash, and V. Bhatnagar. Leveraging structural hierarchy for scalable network comparison. In *International Conference on Database and Expert Systems Applications*, pp. 287–302. Springer, 2016.

[282] 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*, pp. 2292–2298. ACM, 2016.

[283] D. A. Schult and P. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conferences (SciPy 2008)*, vol. 2008, pp. 11–16, 2008.

[284] S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269 – 287, 1983. doi: 10.1016/0378-8733(83)90028-X

[285] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[286] L. Shi, C. Wang, Z. Wen, H. Qu, C. Lin, and Q. Liao. 1.5d egocentric dynamic network visualization. *IEEE Transactions on Visualization and Computer Graphics*, 21(5):624–637, May 2015. doi: 10.1109/TVCG.2014.2383380

[287] H. Shin, G. Park, and J. Han. Tablorer - an interactive tree visualization system for tablet pcs. *Computer Graphics Forum*, 30(3):1131–1140, 2011. doi: 10.1111/j.1467-8659.2011.01962.x

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

[289] H. Song, B. Kim, B. Lee, and J. Seo. A comparative evaluation on tree visualization methods for hierarchical structures with large fan-outs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 223–232. ACM, 2010.

[290] A. Srinivasan and J. Stasko. Orko: Facilitating multimodal interaction for visual exploration and analysis of networks. *IEEE transactions on visualization and computer graphics*, 24(1):511–521, 2018.

[291] J. Stasko, C. Görg, and Z. Liu. Jigsaw: supporting investigative analysis through interactive visualization. *Information visualization*, 7(2):118–132, 2008.

[292] S. S. Stevens. On the psychophysical law. *Psychological Review*, 64(3):153–181, 1957.

[293] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.

[294] J. Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive science*, 12(2):257–285, 1988.

[295] J. Sweller, J. J. Van Merrienboer, and F. G. Paas. Cognitive architecture and instructional design. *Educational psychology review*, 10(3):251–296, 1998.

[296] R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.

[297] R. Tamassia. *Handbook of graph drawing and visualization*. CRC press, 2013.

[298] D. S. Tan, G. Smith, B. Lee, and G. G. Robertson. Adaptivitree: Adaptive tree visualization for tournament-style brackets. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1113–1120, November/December 2007. doi: 10.1109/TVCG.2007.70537

[299] Y. Tanahashi, N. Leaf, and K. Ma. A study on designing effective introductory materials for information visualization. *Computer Graphics Forum*, 35(7):117–126, 2016. doi: 10.1111/cgf.13009

[300] M. Tennekes and E. de Jonge. Tree colors: Color schemes for tree-structured data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2072–2081, December 2014.

[301] A. Thomas, R. Cannings, N. Monk, and C. Cannings. On the structure of protein–protein interaction networks. *Biochemical Society Transactions*, 31(6):1491–1496, 2003. doi: 10.1042/bst0311491

[302] Y. Tu and H. Shen. Balloon focus: a seamless multi-focus+context method for treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1157–1164, November/December 2008. doi: 10.1109/TVCG.2008.114

[303] W. T. Tutte. How to draw a graph. *Proc. London Math. Soc*, 13(3):743–768, 1963.

[304] F. van Ham and A. Perer. "Search, show context, expand on demand": Supporting large graph exploration with degree-of-interest. *IEEE Trans. on Visualization and Computer Graphics*, 15(6):953–960, 2009.

[305] F. van Ham and B. E. Rogowitz. Perceptual organization in user-generated graph layouts. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1333–1339, November/December 2008. doi: 10.1109/TVCG.2008.155

[306] F. Van Ham and J. J. Van Wijk. Interactive visualization of small world graphs. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pp. 199–206. IEEE, 2004.

[307] W. van Heeswijk, G. H. Fletcher, and M. Pechenizkiy. On structure preserving sampling and approximate partitioning of graphs. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pp. 875–882. ACM, 2016.

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

[309] C. Vehlow, F. Beck, and D. Weiskopf. Visualizing group structures in graphs: A survey. In *Computer Graphics Forum*, vol. 36, pp. 201–225. Wiley Online Library, 2017.

[310] K. Verspoor, B. Ofoghi, and M. Robles Granda. Commviz: Visualization of semantic patterns in large social communication networks. *Information Visualization*, 17(1):66–88, 2018.

[311] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, and D. W. Fellner. Visual analysis of large graphs. *Computer Graphics Forum*, 30(6):1719–1749, 2011.

[312] C. Von Mering, R. Krause, B. Snel, M. Cornell, S. G. Oliver, S. Fields, and P. Bork. Comparative assessment of large-scale data sets of protein–protein interactions. *Nature*, 417(6887):399–403, 2002.

[313] S. Voß. Meta-heuristics: The state of the art. In *Local Search for Planning and Scheduling*, pp. 1–23. Springer, 2001.

[314] J. Walny, S. Huron, C. Perin, T. Wun, R. Pusch, and S. Carpendale. Active reading of visualizations. *IEEE transactions on visualization and computer graphics*, 24(1):770–780, 2018.

[315] C. Walshaw. A multilevel algorithm for force-directed graph drawing. In *International Symposium on Graph Drawing*, pp. 171–182. Springer, 2000.

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

[317] C. Ware. *Information visualization: perception for design*. Elsevier, 2012.

[318] C. Ware and R. Bobrow. Supporting visual queries on medium-sized node–link diagrams. *Information Visualization*, 4(1):49–58, 2005.

[319] C. Ware, A. T. Gilman, and R. J. Bobrow. Visual thinking with an interactive diagram. In *Diagrammatic Representation and Inference, 5th International Conference, Diagrams 2008, Herrsching, Germany, September 19-21, 2008. Proceedings*, pp. 118–126, 2008. doi: 10.1007/978-3-540-87730-1_13

[320] C. Ware, H. C. Purchase, L. Colpoys, and M. McGill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110, 2002. doi: 10.1057/palgrave.ivs.9500013

[321] D. J. Watts and S. H. Strogatz. Collective dynamics of âĂŸsmall-worldâĂŹnetworks. *nature*, 393(6684):440, 1998.

[322] C. D. Wickens, J. G. Hollands, S. Banbury, and R. Parasuraman. *Engineering psychology & human performance*. Psychology Press, 2015.

[323] H. P. Williams. *Model building in mathematical programming*. John Wiley & Sons, 2013.

[324] N. Wong, M. S. T. Carpendale, and S. Greenberg. Edgelens: An interactive method for managing edge congestion in graphs. In *9th IEEE Symposium on Information Visualization (InfoVis 2003), 20-21 October 2003, Seattle, WA, USA*, pp. 51–58, 2003. doi: 10.1109/INFVIS.2003.1249008

[325] P. C. Wong, H. Foote, G. C. Jr., P. Mackey, and K. Perrine. Graph signatures for visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1399–1413, November/December 2006. doi: 10.1109/TVCG.2006.92

[326] P. C. Wong, P. Mackey, K. Perrine, J. Eagan, H. Foote, and J. Thomas. Dynamic visualization of graphs with extended labels. In *IEEE Symposium on Information Visualization (InfoVis 2005), 23-25 October 2005, Minneapolis, MN, USA*, p. 10, 2005. doi: 10.1109/INFOVIS.2005.11

[327] F. Wu and B. A. Huberman. Finding communities in linear time: a physics approach. *The European Physical Journal B*, 38(2):331–338, 2004.

[328] Y. Wu, N. Cao, D. Archambault, Q. Shen, H. Qu, and W. Cui. Evaluation of graph sampling: A visualization perspective. *IEEE transactions on visualization and computer graphics*, 23(1):401–410, 2017.

[329] M. Wybrow, K. Marriott, and P. J. Stuckey. Orthogonal connector routing. In *Graph Drawing*, pp. 219–231. Springer, 2010.

[330] K. Xu, C. Rooney, P. Passmore, D.-H. Ham, and P. H. Nguyen. A user study on curved edges in graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2449–2456, December 2012.

[331] J. Yang, Y. Liu, X. Zhang, X. Yuan, Y. Zhao, S. Barlowe, and S. Liu. Piwi: Visually exploring graphs based on their community structure. *IEEE Transactions on Visualization and Computer Graphics*, 19(6):1034–1047, 2013.

[332] X. Yang, L. Shi, M. Daianu, H. Tong, Q. Liu, and P. Thompson. Blockwise human brain network visual comparison using nodetrix representation. *IEEE transactions on visualization and computer graphics*, 23(1):181–190, 2017.

[333] 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*, 23(1):411–420, 2017.

[334] J. S. Yi, Y. ah Kang, and J. Stasko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE transactions on visualization and computer graphics*, 13(6):1224–1231, 2007.

[335] V. Yoghourdjian, T. Dwyer, K. Klein, K. Marriott, and M. Wybrow. Graph thumbnails: Identifying and comparing multiple graphs at a glance. *IEEE Transactions on Visualization and Computer Graphics*, 2018.

[336] D. Yoon, N. H. Narayanan, S. Lee, and O. Kwon. Exploring the effect of animation and progressive revealing on diagrammatic problem solving. In *Diagrammatic Representation and Inference, 4th International Conference, Diagrams 2006, Stanford, CA, USA, June 28-30, 2006, Proceedings*, pp. 226–240, 2006. doi: 10.1007/11783183_31

[337] B. Yost and C. North. The perceptual scalability of visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):837–844, Sept 2006. doi: 10.1109/TVCG.2006.184

[338] X. Yuan, L. Che, Y. Hu, and X. Zhang. Intelligent graph layout using many usersâĂŹ input. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2699–2708, December 2012.

[339] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, pp. 452–473, 1977.

[340] Y. Zhang, Y. Wang, and S. Parthasarathy. Visualizing attributed graphs via terrain metaphor. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1325–1334. ACM, 2017.

[341] J. Zhao, M. Glueck, F. Chevalier, Y. Wu, and A. Khan. Egocentric analysis of dynamic networks with egolines. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pp. 5003–5014. ACM, New York, NY, USA, 2016. doi: 10.1145/2858036.2858488

[342] J. Zhao, M. Glueck, P. Isenberg, F. Chevalier, and A. Khan. Supporting handoff in asynchronous collaborative sensemaking using knowledge-transfer graphs. *IEEE transactions on visualization and computer graphics*, 24(1):340–350, 2018.

[343] J. Zhao, Z. Liu, M. Dontcheva, A. Hertzmann, and A. Wilson. Matrixwave: Visual comparison of event sequence data. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pp. 259–268. ACM, New York, NY, USA, 2015. doi: 10.1145/2702123.2702419

[344] H. Zhou, P. Xu, X. Yuan, and H. Qu. Edge bundling in information visualization. *Tsinghua Science and Technology*, 18(2):145–156, 2013.

[345] C. Ziemkiewicz and R. Kosara. The shaping of information by visual metaphors. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1269–1276, November/December 2008.

[346] C. Ziemkiewicz and R. Kosara. Preconceptions and individual differences in understanding visual metaphors. *Computer Graphics Forum*, 28(3):911–918, 2009. doi: 10.1111/j.1467-8659.2009.01442.x

[347] B. Zimmer and A. Kerren. Displaying user behavior in the collaborative graph visualization system ongrax. In *Graph Drawing and Network Visualization - 23rd International Symposium, GD 2015, Los Angeles, CA, USA, September 24-26, 2015, Revised Selected Papers*, pp. 247–259, 2015. doi: 10.1007/978-3-319-27261-0_21

[348] M. Zinsmaier, U. Brandes, O. Deussen, and H. Strobelt. Interactive level-of-detail rendering of large graphs. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2486–2495, 2012.