Graph Thumbnails: Identifying and Comparing Multiple Graphs at a Glance

Vahan Yoghourdjian, Tim Dwyer, Karsten Klein, Kim Marriott, and Michael Wybrow

Abstract—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 will 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.

Index Terms—network visualisation, circle packing, k-core decomposition, k-connected, network identification, large networks.

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., [6], [32]). 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 can not be untangled sufficiently to understand the connectivity—the infamous "hair-ball" problem (e.g., Fig. 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 is a large number of nodes as the rows and columns become too narrow (e.g., Fig. 1(b)).

Furthermore, 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 inspiration for the technique presented in this paper 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 if, broadly speaking, 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 paper 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 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.

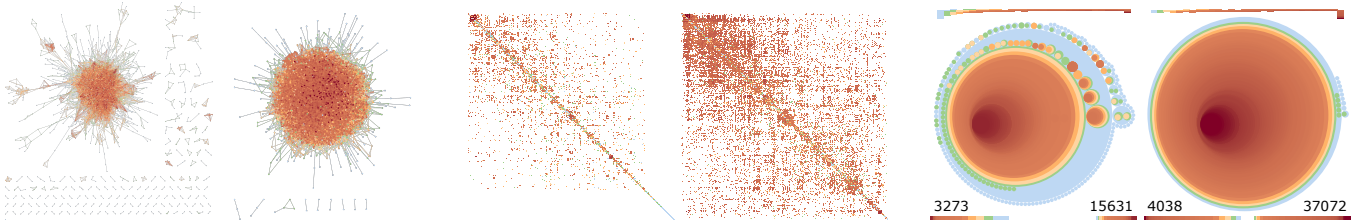
Thus, to summarise, the visualisation tasks we consider are:
Identification: Quick identification of a network from a collection, e.g., browsing through a directory full of graph files.
Comparison: Finding similarly structured networks from a collection, or alternatively finding outliers (i.e., dissimilar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

The main contributions of this paper are as follows:

1) We propose a novel *Thumbnail* representation for networks (Fig. 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, both in terms of the choice of decomposition

- *Vahan Yoghourdjian, Tim Dwyer, Kim Marriott, and Michael Wybrow are with Monash University.*
  E-mail: vahan.yoghourdjian@monash.edu, Tim.Dwyer@monash.edu, Kim.Marriott@Monash.edu, Michael.Wybrow@Monash.edu
- *Karsten Klein is with the University of Konstanz, and is an adjunct research fellow with Monash University.*
  E-mail: karsten.klein@uni-konstanz.de

Manuscript received August 26, 2017;



(a) Force-directed node-link diagrams (b) Adjacency Matrices with overplotting (c) Graph Thumbnails

Fig. 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.

used to obtain a representative hierarchy of the graph and the visual elements of the thumbnails (Section 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 5)—in support of **Identification** and **Comparison** tasks.
4) We evaluate people's ability to read structural details about graphs using these three representations (Section 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 7)—**Comparison** and **Overview** tasks.

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.* [38] in their task taxonomy for graph visualisation. Krzywinski *et al.* [37] 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, where the set of nodes in the graph is wholly or partially (if nodes can appear and disappear) the same in each graph. For example Burch and Weiskopf [13] 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.* [7] 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., [40]) or to visualise sets of structural statistics across a graph corpus (e.g., Kennedy *et al.* [35]—we use this "graph landscape" information to classify the corpus of graphs used in Study 1, Section 5). Kairam *et al.* [34] developed a visual dashboard display of structural summary statistics for a network (such as centrality metrics). Similarly Freire *et al.* [24] 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 [57] or structurally [8] grouped and drawn as bundles. Cliques and bi-partite cliques of nodes can be identified and their links implied through aggregate connections [20], [21], [44]. Alternately, the links can be omitted entirely and, instead, communities of highly-linked nodes represented in contiguous coloured regions [25]. 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 triconnected components, in particular for planar graph drawing [18], k-core decomposition [11], [26], [33] or even simple stochastic sampling [42], [51]. Seidman introduced the notion of a k-core [48] 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 later was extended to weighted graphs [27], and further generalised to so-called *nuclei*, which represent more complex connectivity structures, but are also harder to intuitively interpret [46]. 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" [4] or 2.5-dimensional levels [12] reflecting their coreness. In particular Alvarez *et al.* [4] 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.* [5] 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.* [56] 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

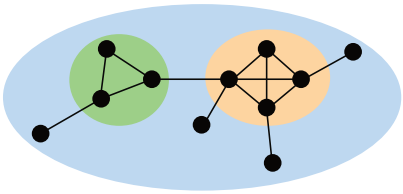


Fig. 2. A network where all the vertices are connected to each other. The blue ellipse represents the 1-connected component. The green one represents the 2-connected component. While the orange ellipse represents the 3-core component. In this example one 1-connected component contains all the vertices 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 (in steps, starting with 0 to k-1).

k-core depth. Their 3D renderings could be considered a hybrid of our icicle and treemap representations.

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 regards it is possible to separate these two concerns. As will be discussed, the semantics of the decomposition used in the examples in this paper 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 for its linear running time and unambiguous (canonical) nature.

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 would 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 [28] uses a greedy method to attempt to maximise the modularity score of each cluster. We can not 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; 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., [43]) 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. Fig. 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, while 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 into 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 triconnected components in linear time ([17], [29]), 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 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.* [14] only for so-called k-blocks, 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 [11]. A k-core of a graph is a component of the subgraph found by repeatedly removing vertices that have degrees less than k. Fig. 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 ($k \geq 3$).

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 triconnected 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 triconnected 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 triconnectivity and so we opted for the simpler definition above.

Figure 3 shows an example of applying this decomposition on the popular network of Zachary's Karate Club [55].

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

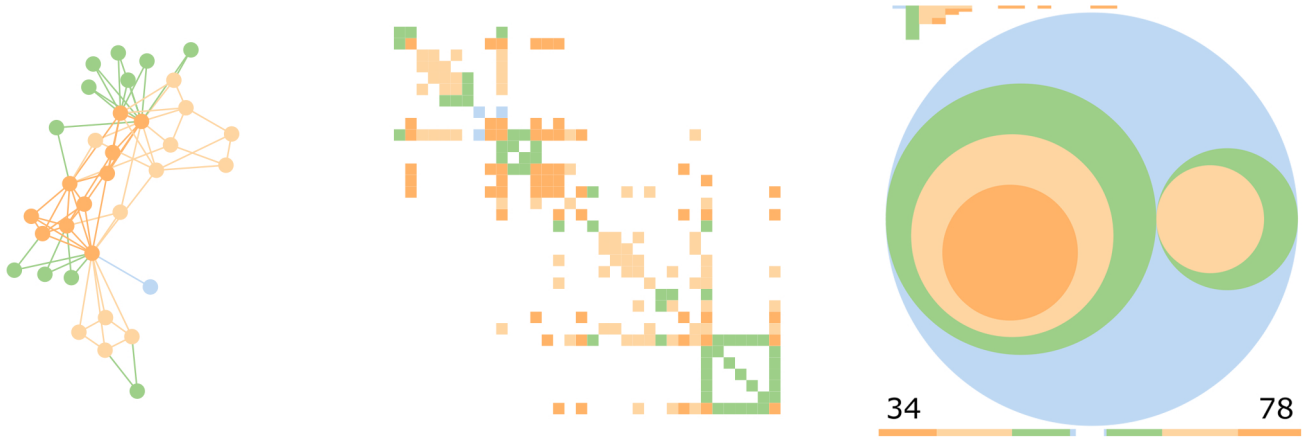


Fig. 3. Zachary's Karate Club network [55], shown using node-link, matrix and *Graph Thumbnail* representations. The colours show the four levels of the KC^3 decomposition. This network is known for its two main communities: one around the administrator and a denser one around the instructor. These communities are respected in the KC^3 decomposition and can be clearly seen in the *Graph Thumbnail* representation.

3.2 Visual Design

As already mentioned, the inspiration for *Graph Thumbnails* was to have a small visual, showing a minimum amount of detail but with great precision, that is suggestive of a conventional large-scale network visualisation. We can expand upon these requirements for *Graph Thumbnail* visuals as follows:

R1 they be suggestive of a standard large-scale network visualisation technique;

R2 they be as readable as possible to support small-multiple comparisons;

R3 the level in the cluster hierarchy of each visual element be clear;

R4 the visual elements accurately convey the relative size of the cluster to which they correspond;

R5 the representation should scale to arbitrarily large or dense networks.

Beginning by considering **R1**, force-directed layouts such as those in Fig. 1(a) suggest an arrangement of circles. Adjacency matrices such as those in Fig. 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.

3.2.1 Treemaps

Treemaps over a cluster hierarchy have been considered before by Muelder and Ma as a layout strategy for large graphs [19], [41]. 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 Fig. 4(a)). Being space-filling, 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**).

3.2.2 Icicle Plots

It is well known in psychology [49] and reinforced by experimental results in readability of information graphics [16], that encoding a scalar value with area causes people to underestimate the true value. Thus, we also considered icicle plots [36], which have the advantage over treemaps of 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 Fig. 4(b), became unreadable when reduced to thumbnail size (**-R2**,**-R5**).

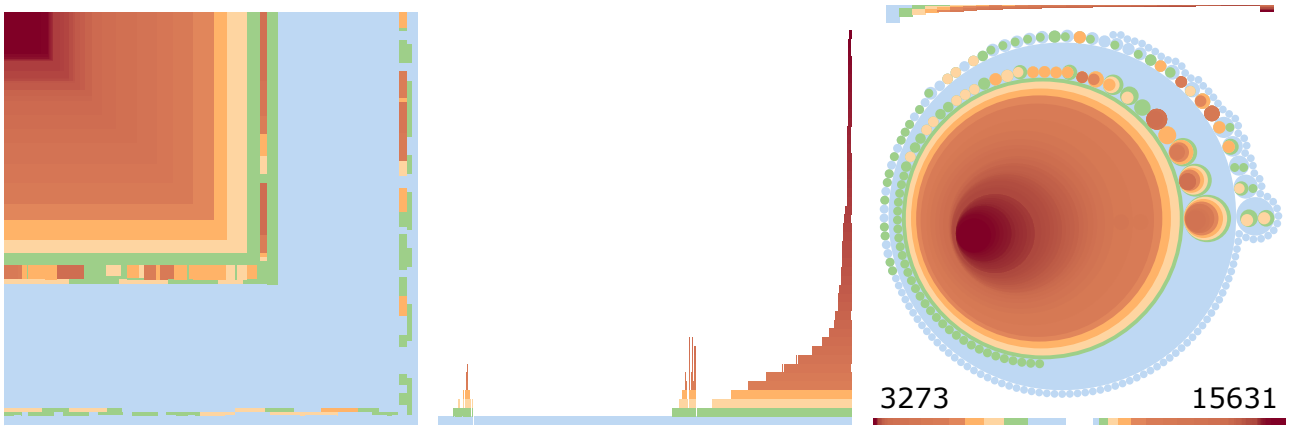
3.2.3 Circles

The third design option we considered involved a nested circle packing, employing the algorithm proposed by Wang *et al.* [53]. 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**) and also 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 [39] 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**).

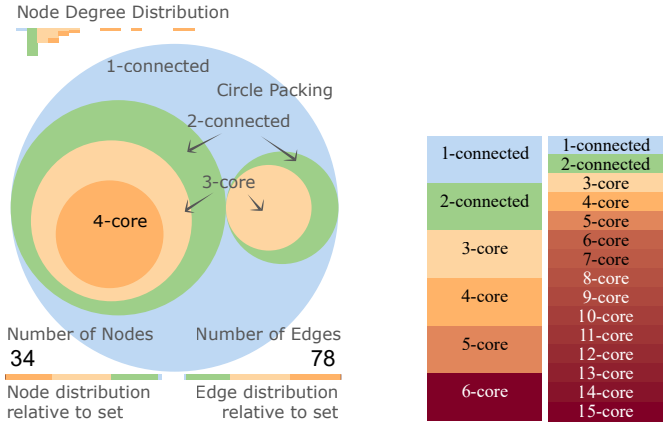
3.2.4 Colour Scheme

Our colour scheme evolved between Study 1 and Study 2. The original colour scheme used in Study 1 is shown in Fig. 7. Essentially, we used a diverging palette with warm colours for



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

Fig. 4. Comparison of the three visual designs explored for *Graph Thumbnails*.



(a) Layout, annotations and adornments. (b) Adaptive colour scheme.

Fig. 5. *Graph Thumbnail* visual design. We use the *Pack* function from *D3* [1], 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.

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 Fig. 5(b).

In the examples in this paper 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 each connects. These coloured node-link and matrix representations were additional conditions in Study 2.

3.2.5 Annotations and Adornments

In several examples shown in this paper as well as the use-case described in Section 7, the *Graph Thumbnails* were augmented with additional annotations. Examples of these annotations are detailed in Fig. 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 be 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 break-down 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.

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 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 canonical ordering. The canonical encoding of a tree can be computed in linear time [31, Chapter 3].
3) **Circle Packing**. We use the *D3* [1] implementation of the circle packing algorithm proposed by Wang *et al.* [53]. 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 [31, 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* [30] 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 Fig. 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 Fig. 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 *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.

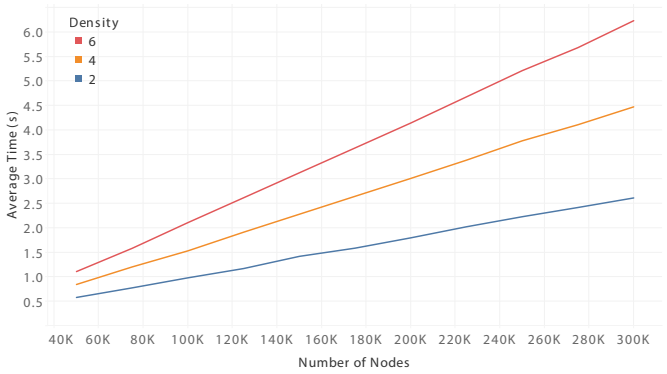


Fig. 6. The average running time of creating *Graph Thumbnails* for large graphs of different sizes. We were able to represent graphs with 300,000 nodes and 1,800,000 edges in approximately six seconds.

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 Section 2.

We came up with five different graph classes based on different generating techniques. In general we focused on generators which 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* [30] generators: Watts-Strogatz and ErdősRényi [47].

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 [10]. 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 Fig. 7, each with 1,000 nodes.

The NL stimuli were created using the force layout of D3.js. [1] MX stimuli were created and ordered using Reorder.js with barycenter reordering [23]. At first we found 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.

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

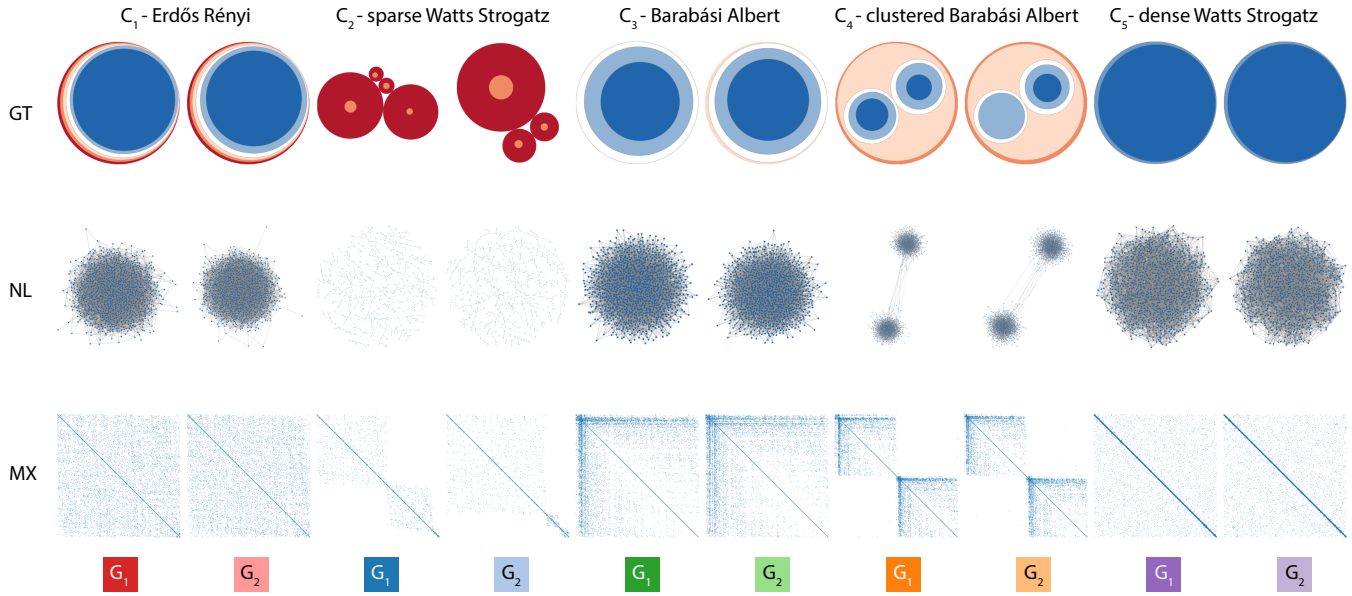


Fig. 7. Study 1 stimuli. Ten graphs from five graph classes, shown in three visual representations: *Graph Thumbnail*, node-link and matrix.

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.

5.2 Hypotheses

- **H1** We hypothesised that large graphs shown using MX would be more distinguishable than NL. Even though the relationships between nodes is 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.

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 $S(C_iG_k, C_jG_l)$ $i, j \in \{1,\ldots,5\}$ $k,l \in \{1,2\}$.

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

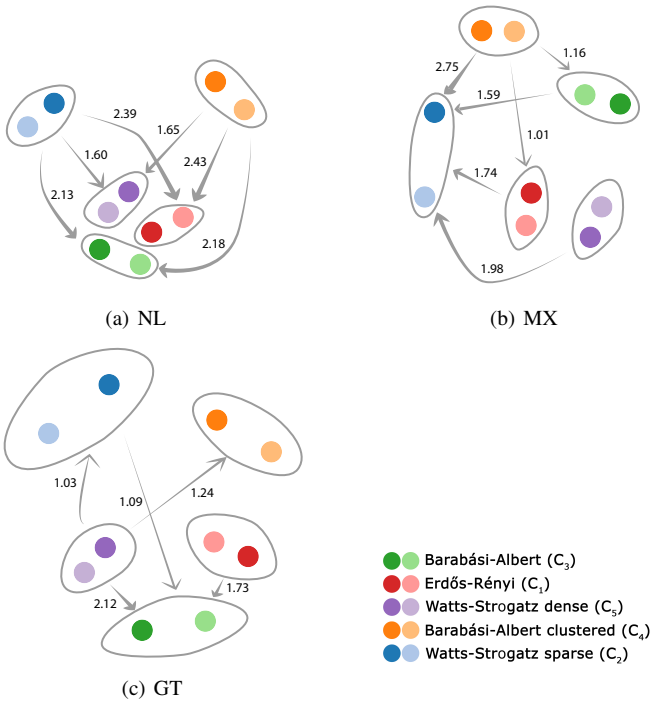


Fig. 8. Multi-dimensional scaling plot of the results of the first study. The distances inversely represent the mean of the similarity rating for each pair of graphs. The arrows show statistically significant differences in discriminability between graph classes. The direction goes from higher discriminability to lower.

Fig. 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 suggest that GT and MX helped the participants identify the

Fig. 9. A multi-dimensional scaling plot with distances representing dissimilarity of 17 graph properties, computed using the *Graph Landscape* method [35]. 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.

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.,

$$selfsimilarity_{12}(C_i) = S(C_iG_1, C_iG_2) \tag{1}$$

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:

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

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 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.

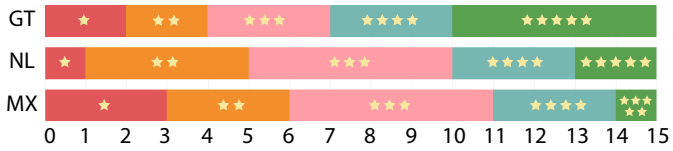


Fig. 10. 15 participants submitted a 5-star rating of how easy it was to do the similarity tasks given each visual representation. More than 50% of the participants rated GT on the easy side, while only 33.33% and less than 30% of the participants thought MX and NL were easy respectively.

The arrows in Fig. 8 show the statistically significant differences in discriminability between the five classes. The direction goes from higher discriminability to lower and 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. Fig. 10 shows the results of the 5-star rating of the post-study survey. More than 50% of respondents rated GT as 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.

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 [35]. An MDS map may be used to spot clusters and outliers, as shown in Fig. 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 Fig. 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 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 1000 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 Study 2: Understanding Structure

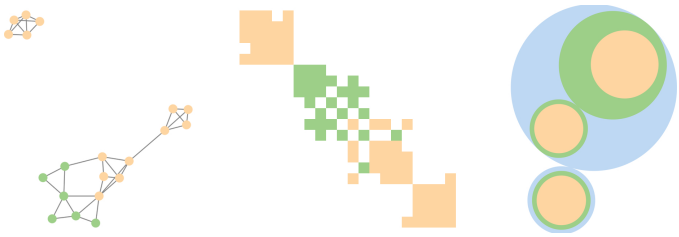
Study 1 did not require participants to interpret or understand the visual representations, only to notice visual differences supporting **Identification** and **Comparison**. We therefore performed a second study to evaluate whether GT, NL and MX convey structural properties sufficiently to support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

The study had four tasks using 12 different 20-node graphs, randomly picked from the Rome Graphs corpus [2]. 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 was controlled by randomly adding and removing links.

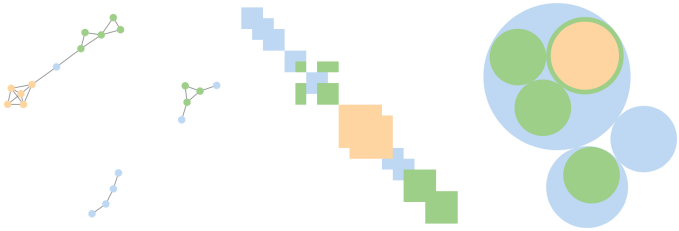
As in Study 1, we used GT, NL and MX to show the graphs. For GT we used the final design colour scheme discussed in Section 3. We had two variations of NL and MX: grey (NLGrey, MXGrey) and coloured versions (NLColour, MXColour) using a similar colour scheme as GT. For MXColour, if the nodes connected by an edge belonged to different components of different levels, the cell representing the edge would be coloured according to the higher component.

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 [3] which explained the basics of connectivity and presented the different visual representations. Each task had several training questions which had to be answered correctly before the participants could proceed.



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



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

Fig. 11. Sample stimuli from Study 2. Answers were multiple choice (0…6 or unsure).

6.1.1 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 Fig. 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.

6.1.2 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 Fig. 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.

6.1.3 Task 3

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

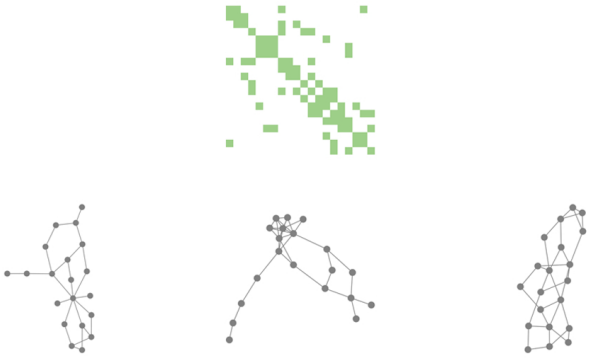
- **H3.1** We hypothesised MX would be better than NL.
- **H3.2** Similarly, we expected MX to be better than GT. As mentioned previously, GT did not include annotations for this

study (an edge count annotation would have made this task trivial).

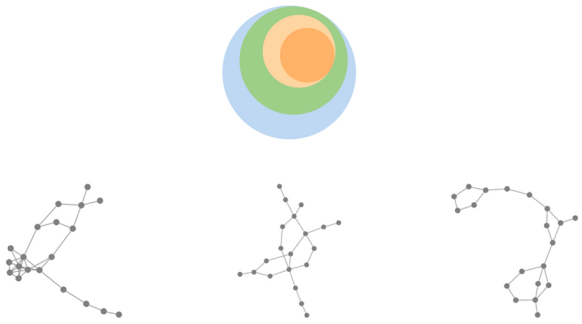
6.1.4 Task 4

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 (Fig. 12).

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



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



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

Fig. 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.

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 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.

6.3 Results

Tasks 1 and 2 each had 8 trials $\times$ 5 techniques $\times$ 26 participants = 1040 trials. For Task 3, we had 10 trials $\times$ 5 techniques $\times$ 26 participants = 1300 trials, and for Task 4 there were 10 trials $\times$ 3 techniques $\times$ 26 participants = 780 trials. In total we had 4160 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". "unsure" answers were counted 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 and summing them for each visual stimuli, task and participant. Full results are shown in Fig. 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.

6.4 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.

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

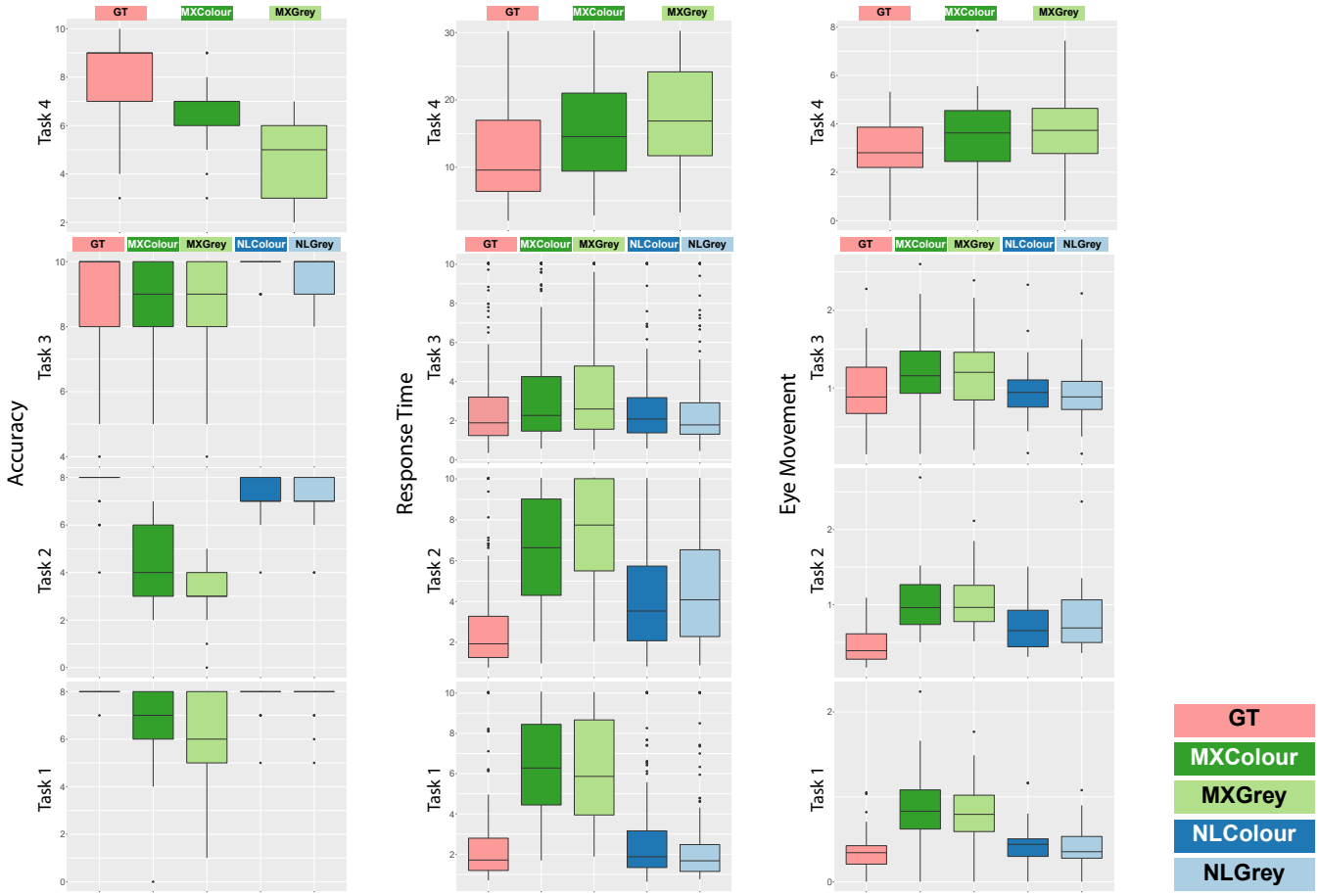


Fig. 13. Study 2 results. All three measures: accuracy on the left, response time (in seconds) in the middle, and total eye movement (in 0.1 megapixels) on the right, show a similar trend where the participants performed better on *Graph Thumbnails* than matrix representations. They also performed better on node-link diagrams than matrix representations.

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 to 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 Fig. 1 and Fig. 14.

We plan to further analyse the eye tracker data. It would be interesting to find out where the participants fixated more on each of the visual stimuli and if it changes across the different tasks. It would also be interesting to find out if there are common tactics that the participants used to perform the tasks.

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 [15], [50], 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 [52].

PPI networks are often represented as hairball node-link diagrams, 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

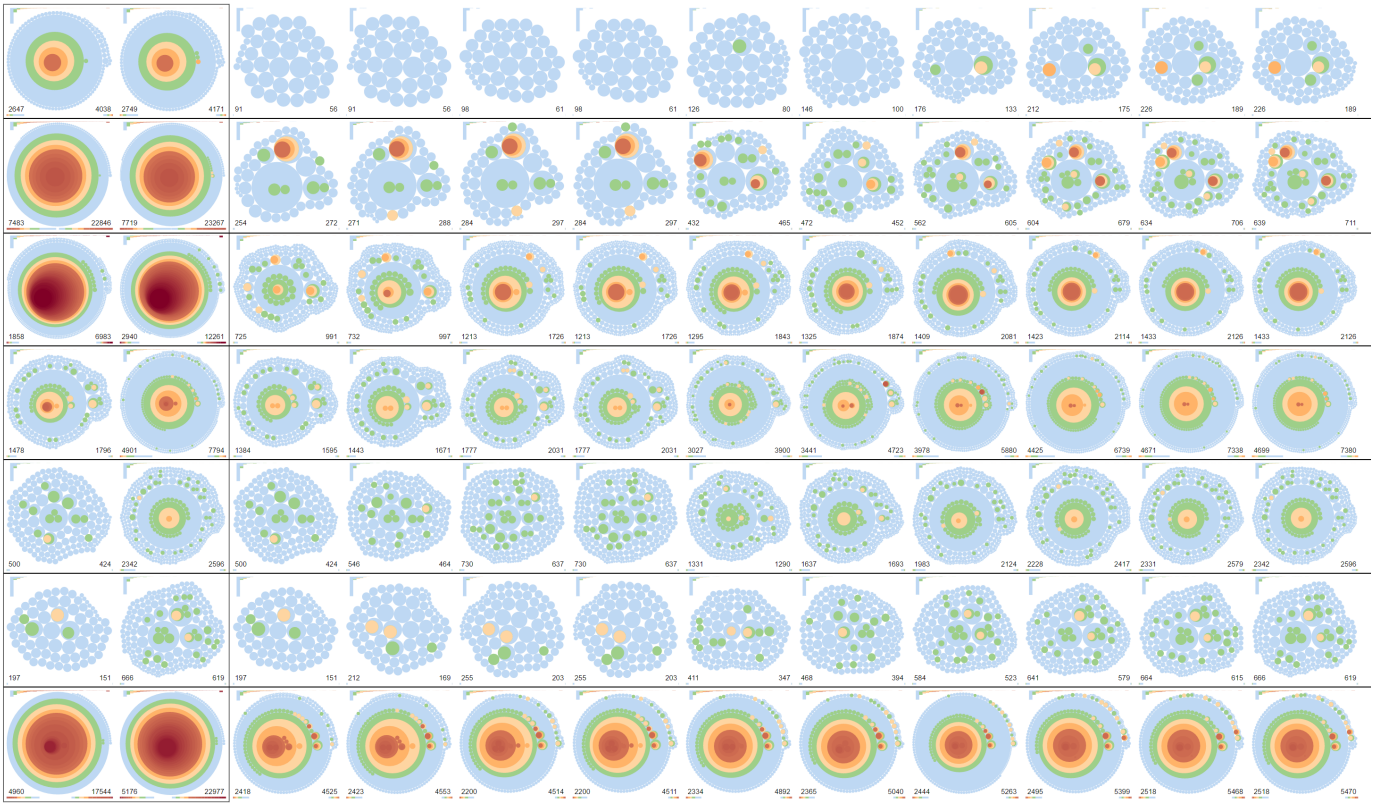


Fig. 14. Evolution of DIP database structure for seven organisms (*C. elegans*, *D. melanogaster*, *E. coli*, *H. sapiens*, *M. musculus*, *R. norvegicus*, *S. cerevisiae*), each row shows data for one organism (from top to bottom in the listed order). The two leftmost columns show the full DIP dataset 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).

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) [45]—which includes manually curated PPI data—for the years 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 Fig. 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.

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 while there are other possibilities, some of which we mention in Section 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.

9 Conclusion and Future Work

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.

In future, we plan to experiment with circle packing algorithms such that the layout can convey other aspects of graph structure. Some possibilities for adjusting, e.g., 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 [9] might be a starting point for this.

Another obvious area to explore is interaction. 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 such as [22]. For example, a smaller neighbourhood of interest could be shown in a detailed high-quality view (e.g. [54]), while the cluster memberships of the nodes in the neighbourhood could be indicated through highlights in the thumbnail overview.

Acknowledgments

This work was supported by the Australian Research Council Discovery Project grant DP140100077. We thank Jens Schmidt and Markus Chimani for sharing their knowledge on graph decompositions.

References

[1] D3.js: http://d3js.org.
[2] Rome Graphs: http://www.graphdrawing.org/data/.
[3] Tutorial slides for Study 2: http://www.vahany.com/gt/study2/tutorial/.
[4] 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.
[5] D. Archambault, T. Munzner, and D. Auber. Topolayout: Multilevel graph layout by topological features. *IEEE transactions on visualization and computer graphics*, 13(2), 2007.
[6] A. Arleo, W. Didimo, G. Liotta, and F. Montecchiani. *A Distributed Multilevel Force-Directed Algorithm*, pages 3–17. Springer International Publishing, Cham, 2016.
[7] 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*, volume 34, pages 31–40. Wiley Online Library, 2015.
[8] 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.
[9] 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*, pages 165–172. ACM, 2005.
[10] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.
[11] V. Batagelj and M. Zaversnik. An O(m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*, 2003.
[12] M. Baur, U. Brandes, M. Gaertler, and D. Wagner. Drawing the as graph in 2.5 dimensions. In *International Symposium on Graph Drawing*, pages 43–48. Springer, 2004.
[13] 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*, page 29. ACM, 2014.
[14] 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.
[15] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Rev.*, 51(4):661–703, Nov. 2009.
[16] 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.
[17] G. Di Battista and R. Tamassia. Incremental planarity testing. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 436–441. IEEE, 1989.
[18] G. Di Battista and R. Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15(4):302–318, 1996.
[19] W. Didimo and F. Montecchiani. Fast layout computation of clustered networks: Algorithmic advances and experimental analysis. *Information Sciences*, 260:185–199, 2014.
[20] 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*, pages 3247–3256. ACM, 2013.
[21] 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.
[22] 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.
[23] J.-D. Fekete. Reorder.js: A JavaScript library to reorder tables and networks. In *IEEE VIS 2015*, 2015.
[24] 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*, pages 213–222. ACM, 2010.

[25] E. R. Gansner, Y. Hu, and S. Kobourov. Gmap: Visualizing graphs and clusters as maps. In *Visualization Symposium (PacificVis), 2010 IEEE Pacific*, pages 201–208. IEEE, 2010.

[26] 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.

[27] 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*, pages 87–93, July 2011.

[28] 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.

[29] C. Gutwenger and P. Mutzel. *A Linear Time Implementation of SPQR-Trees*, pages 77–90. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[30] 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)*, pages 11–15, Pasadena, CA USA, Aug. 2008.

[31] J. E. Hopcroft, J. D. Ullman, and A. Aho. The design and analysis of computer algorithms, 1975.

[32] Y. Hu and L. Shi. Visualizing large graphs. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(2):115–136, 2015.

[33] H. Kabir and K. Madduri. Parallel k-core decomposition on multicore platforms. In *Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2017 IEEE International*, pages 1482–1491. IEEE, 2017.

[34] 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*, pages 498–505. ACM, 2012.

[35] A. Kennedy, K. Klein, A. Nguyen, and F. Y. Wang. The graph landscape: using visual analytics for graph set analysis. *Journal of Visualization*, pages 1–16, 2016.

[36] J. B. Kruskal and J. M. Landwehr. Icicle plots: Better displays for hierarchical clustering. *The American Statistician*, 37(2):162–168, 1983.

[37] M. Krzywinski, I. Birol, S. J. Jones, and M. A. Marra. Hive plotsrational approach to visualizing networks. *Briefings in bioinformatics*, 13(5):627–644, 2011.

[38] 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*, pages 1–5. ACM, 2006.

[39] M. J. McGuffin and J.-M. Robert. Quantifying the space-efficiency of 2d graphical representations of trees. *Information Visualization*, 9(2):115–140, 2010.

[40] 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*, pages 117–128. IEEE, 2002.

[41] C. Muelder and K.-L. Ma. A treemap based method for rapid layout of large graphs. In *Visualization Symposium, 2008. PacificVIS'08. IEEE Pacific*, pages 231–238. IEEE, 2008.

[42] 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.

[43] P. Pons and M. Latapy. Computing communities in large networks using random walks. In *International Symposium on Computer and Information Sciences*, pages 284–293. Springer, 2005.

[44] L. Royer, M. Reimann, B. Andreopoulos, and M. Schroeder. Unraveling protein networks with power graph analysis. *PLoS Comput Biol*, 4(7):e1000108, 2008.

[45] 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.

[46] 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, pages 927–937. ACM, 2015.

[47] 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)*, volume 2008, pages 11–16, 2008.

[48] S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269 – 287, 1983.

[49] S. S. Stevens. On the psychophysical law. *Psychological Review*, 64(3):153–181, 1957.

[50] 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.

[51] 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*, pages 875–882. ACM, 2016.

[52] 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.

[53] 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*, pages 517–520. ACM, 2006.

[54] V. Yoghourdjian, T. Dwyer, G. Gange, S. Kieffer, K. Klein, and K. Marriott. High-quality ultra-compact grid layout of grouped networks. *IEEE transactions on visualization and computer graphics*, 22(1):339–348, 2016.

[55] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, pages 452–473, 1977.

[56] 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*, pages 1325–1334. ACM, 2017.

[57] H. Zhou, P. Xu, X. Yuan, and H. Qu. Edge bundling in information visualization. *Tsinghua Science and Technology*, 18(2):145–156, 2013.

Vahan Yoghourdjian is a PhD candidate at Monash University, Australia, since 2014. His main supervisor is Associate Professor Tim Dwyer. His current research is in network visualisation. He received his M.Sc. in Computer Science from Notre Dame University, Lebanon, in 2013. and B.Sc. in Computer Science from Haigazian University, Lebanon, in 2009.

Tim Dwyer is an Associate Professor of Computer Science with Monash University, Australia. He joined Monash in 2012 as a Larkins Fellow. His research is in the areas of network visualisation, optimisation and immersive analytics. He has also worked for Microsoft, USA—researching and developing tools for software visualisation in the Visual Studio IDE. He received his PhD from the University of Sydney in 2005.

Karsten Klein is a Postdoctoral Research Fellow at the University of Konstanz, Germany. His research is in the areas of visual analytics approaches for complex data from application areas, in particular from the life sciences, and on network analysis and graph drawing algorithms. He received his PhD from TU Dortmund, Germany in 2011.

Kim Marriott is a Professor in Computer Science at Monash University, Australia. His research is in data visualisation, human-in-the-loop analytics, assistive technologies and immersive analytics. After obtaining his PhD from the University of Melbourne in 1989 he worked at the IBM TJ Watson Research Center until joining Monash in 1993.

Michael Wybrow is a Senior Lecturer in Computer Science at Monash University, Australia. His research is in the areas of constraint-based network layout, information visualisation, human-computer interaction, and connector routing. He likes practical problems and has worked on a number of industry-sponsored research projects. He received his PhD from Monash University in 2008.