Comparative Evaluation of Bipartite, Node-Link, and Matrix-Based Network Representations

Moataz Abdelaal, Nathan D. Schiele, Katrin Angerbauer, *Student Member, IEEE*, Kuno Kurzhals, Michael Sedlmair, *Member, IEEE*, and Daniel Weiskopf, *Member, IEEE*

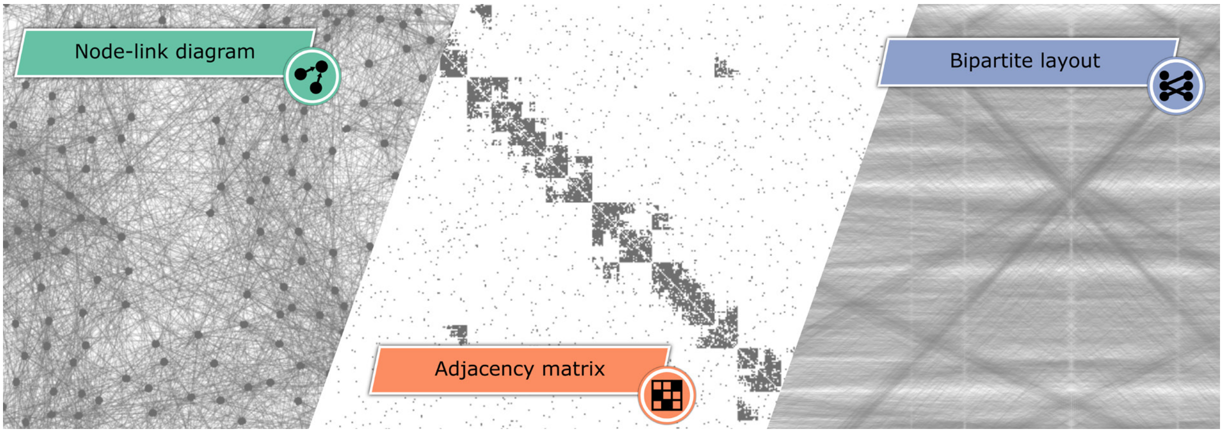


Figure 1: Three visualization techniques for the representation of large networks. Node-link diagrams and adjacency matrices are common in visualization. Bipartite layouts have been proposed as an alternative for solving different tasks. We compare all three techniques with respect to different network properties and tasks.

Abstract—This work investigates and compares the performance of node-link diagrams, adjacency matrices, and bipartite layouts for visualizing networks. In a crowd-sourced user study (n = 150), we measure the task accuracy and completion time of the three representations for different network classes and properties. In contrast to the literature, which covers mostly topology-based tasks (e.g., path finding) in small datasets, we mainly focus on overview tasks for large and directed networks. We consider three overview tasks on networks with 500 nodes: (T1) network class identification, (T2) cluster detection, and (T3) network density estimation, and two detailed tasks: (T4) node in-degree vs. out-degree and (T5) representation mapping, on networks with 50 and 20 nodes, respectively. Our results show that bipartite layouts are beneficial for revealing the overall network structure, while adjacency matrices are most reliable across the different tasks.

Index Terms—Bipartite, network, visualization, evaluation

1 Introduction

Node-link diagrams (**NL**) 🟢 and adjacency matrices (**AM**) 🟠 are two of the most common visualizations for static networks (Figure 1). 🟢 is easy to understand but suffers from visual clutter when the size and density of networks increase. 🟠 is a visualization free of overdrawing and is highly scalable with respect to network density. However, spatial properties are neglected, impairing path-related tasks [28, 29, 35] (e.g., following a path through a network). To address this problem, several techniques have been proposed, either by combining the visual variables of both representations [30, 31, 53] or by introducing a whole new representation altogether [32, 34, 63]. One of those is the bipartite layout (**BP**) 🔵.

In contrast to the aforementioned techniques, 🔵 was originally introduced to visualize bipartite graphs [9], i.e., graphs that have two sets of non-adjacent vertices. In recent years, the layout has been proposed as an alternative visual representation of static and dynamic networks [11, 18]. The layout depicts networks by connecting nodes on two vertical axes (Figure 2c). Due to its compact design, the layout is often used to *compare* multiple instances of the same network, which is an essential analysis task in dynamic network visualization. The technique shares some characteristics with 🟢 and 🟠. It uses the same visual variables as 🟢 to encode nodes and links. Similar to 🟠, it relies on vertex ordering techniques [13, 27, 40] to reveal the network structure. In contrast to 🟠, the axes in 🔵 are parallel and not orthogonal to each other. This makes it more scalable than 🟠 with respect to the network size [4, 16]. To our knowledge, there is no formal evaluation of 🔵 thus far. Despite the increasing number of empirical user evaluations in network visualizations in the last two decades [17], the capabilities of 🔵 for static network visualization remain largely unknown, especially in comparison to 🟢 and 🟠.

In the same vein, Yoghourdjian et al. [62] identified multiple aspects of network visualization that have not been investigated by empirical user evaluation. For example, the majority of studies (80%) were conducted on small-size networks (100 nodes or fewer), and studies that considered large-size networks used network abstraction or aggregation methods for the evaluation [62]. While the basic measures of network complexity, such as size and density, are important evaluation parameters, they cannot be viewed independently from the underlying network class or model. Only few studies [63] considered network class as an important factor in the evaluation and assessed how it might influ-

- *Moataz Abdelaal, Katrin Angerbauer, Kuno Kurzhals, Michael Sedlmair, and Daniel Weiskopf are with University of Stuttgart. E-mail: lastname.firstname@visus.uni-stuttgart.de.*
- *Nathan D. Schiele is with Leiden University. E-mail: n.d.schiele@liacs.leidenuniv.nl.*

Manuscript received 31 March 2022; revised 1 July 2022; accepted 8 August 2022. Date of publication 3 October 2022; date of current version 2 December 2022. Digital Object Identifier no. 10.1109/TVCG.2022.3209427

ence the performance of the participants. When it comes to network directionality, 68% of the studies were based on undirected networks. Regarding user tasks, the most common tasks are the topology-based tasks [38], with path finding being the most common; and overview-based tasks are the least common [62].

In our work, we fill some of the aforementioned gaps by evaluating ⊗ for network visualization and comparing it to the two traditional approaches 🟢 and 🟠, with respect to multiple network classes, properties, and analysis tasks. Our work extends the body-of-knowledge of empirical evaluations for network visualization in several ways:

- We consider large *directed* networks with 500 nodes.
- We explore the interplay between network class and density. In this regard, we consider four different network classes and three density profiles.
- We evaluate five tasks that target different levels of network granularity. Three overview tasks (T1 – T3) on networks with 500 nodes. Two detailed tasks (T4 and T5) on networks with (50 and 20) nodes, respectively.
- We contribute new insights regarding ⊗ as a method for network visualization and how it compares to 🟢 and 🟠.

We conducted a large-scale crowd-sourced user study ($n = 150$) to assess the performance of participants for the different network interpretation tasks. Our results provide empirical evidence about the advantages and shortcomings of the three techniques under different conditions. Based on the results, we derive guidelines for the use of the three techniques depending on the task and data characteristics.

2 Related Work

Related work comprises a multitude of studies investigating node-link and adjacency matrix visualizations. We focus on studies comparing both techniques in terms of task performance. We distinguish ourselves from previous work with respect to: (1) network properties, with a particular focus on network size; (2) the inclusion of network class as a variable in the evaluation; and (3) the tasks, with a particular focus on network overview and comparison tasks. Furthermore, we discuss the use of bipartite graph layouts in the literature and how it was applied for network visualization.

2.1 Evaluation of Node-Link and Adjacency Matrix

In the last two decades, there has been an increasing number of empirical user evaluations in network visualization. For a detailed and thorough review, we refer the readers to the surveys of Yoghourdjian et al. [62] and Burch et al. [17]. An early evaluation of 🟢 was conducted by Purchase [51], where they asked the participants to perform three tasks, such as *finding the shortest path between two given nodes* on eight different layouts of 🟢. They found no statistical difference in performance, except for one layout. The finding is interesting as it shows that graph aesthetic criteria had little influence on task performance. Ghoniem et al. [28, 29] conducted a seminal study in which they compared 🟠 and 🟢 using synthetically generated networks on a number of simple and abstract tasks, such as *finding the most connected node*, or *finding a link between two nodes*. The main result of the work is that 🟠 is useful for visualizing dense networks, whereas 🟢 is better suited for path-related tasks. These findings were confirmed later by Keller et al. [35] and Henry and Fekete [30] on real-world datasets. Beradi et al. [14] compared 🟢 and 🟠 in the context of intelligence tasks such as *identifying the potential leaders within the network* or *identifying network clusters*. In their evaluation, 🟢 performed better with respect to accuracy and response time. Hlawatsch et al. [32] compared both techniques for visualizing weighted, directed, and dynamic networks. Participants were faster and made fewer errors in 🟠 when they had to compare edge weights across several points in time.

While previous work evaluated basic measures of complexity such as size and density, more recent studies tried to leverage crowdsourcing evaluation to compare 🟢 and 🟠 across a multitude of network tasks. For example, Okoe et al. recruited 557 online users in [47] and 864 in [48] to evaluate 14 different tasks on two real-world datasets. Similarly, Ren et al. [52] recruited 600 participants to evaluate how people understand social networks by measuring their performance in 16 different tasks. Nobre et al. [46] compared 🟢 and 🟠 in the context of multivariate networks. They evaluated 16 tasks with a special focus on attribute-based tasks. Except for the work of Okoe et al. [47, 48], most of the previous studies evaluated networks with 100 nodes or less. Additionally, most of the evaluated tasks are low-level (i.e., topology- or attribute-based tasks) [38], except for the task *"How many clusters are there in the visualization?"* in the work of Okoe et al. [47, 48]. In contrast to previous work, we consider larger networks with 500 nodes. Additionally, we focus our evaluation more on network overview and comparison tasks.

There are a few prior studies that considered network overview and comparison tasks. For example, Alper et al. [6] evaluated network comparison tasks on both 🟢 and 🟠, using brain connectivity networks. Similarly, Jin et al. [34] considered "overall similarity" as one of the tasks in their evaluation. Our work builds upon that by evaluating large networks and investigating different network classes in the evaluation. We position ourselves close to the work done by Yoghourdjian et al. [63], where they compared 🟢 and 🟠 using large networks with thousand nodes across different network classes. In contrast to their work, we investigate the interplay between the network class and network density by including the density as an additional parameter in the evaluation. Additionally, while Yoghourdjian et al. evaluated one overview task (i.e., identifying similarity), we extend that procedure by evaluating three overview tasks and two detailed tasks for directed networks. Furthermore, we leverage a large number of participants by conducting a crowdsourcing user study with 150 participants. Finally, we evaluate ⊗ layout as a method for visualizing directed networks and compare it against the two standard techniques 🟢 and 🟠, which has not been done to date.

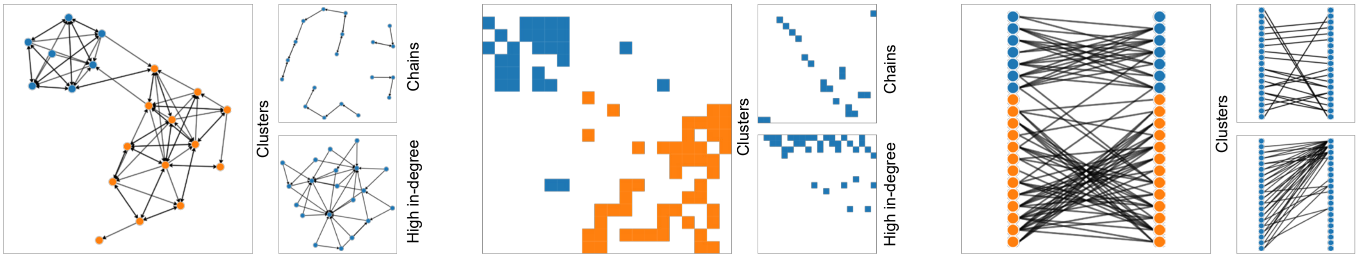
2.2 Bipartite Graph and Layout

Bipartite graphs are a special kind of graph where the vertices are partitioned into two disjoint sets. These graphs are often found in biological and biochemical reaction networks [9]. While there is a substantial body of work on visualizing bipartite graph structures [43, 50, 54, 61], we focus on using bipartite layouts for visualizing single-mode networks (i.e., one set of vertices). Inspired by parallel coordinates plots, Burch et al. proposed Parallel Edge Splatting [18], a technique for visualizing dynamic networks where they used a bipartite layout as the underlying visual representation (Figure 2c). To encode the time dimension, the individual networks from different timepoints are juxtaposed next to each other in a small multiples fashion. To reduce visual clutter, the edges are splatted by computing a pixel density map. Several subsequent techniques were introduced aiming to increase the scalability of parallel edge splatting with respect to the number of timepoints, either by applying a time sliding window [12], interleaving [16], stacking [3], or by partially drawing the edges [4]. All this work showed that bipartite layouts can be beneficial for dynamic network visualization. The 1D arrangement of vertices allows the layout to be scalable with respect to the network size and number of timepoints. Furthermore, with proper vertex ordering, the layout can reveal several network structures [18] and is therefore beneficial for network overview and comparison tasks. In previous work, evaluations were typically done by showing use cases on a few real-world datasets. While this is a valid evaluation method [37], in our work, we complement the prior work by conducting an empirical user evaluation to assess the capability of bipartite layout for static network visualization and compare it against the traditional network representations.

3 Network Representations

We investigate three visualization techniques for large and directed graphs. We consider undirected graphs a sub-case of the directed ones. In this section, we briefly introduce the ⊗ layout as it is less commonly used for visualizing networks and compare it against the baseline techniques, 🟢 and 🟠.

The ⊗ layout consists of two vertical, parallel axes. The left axis corresponds to the source nodes, while the right axis corresponds to



(a) Node-link diagram ● (b) Adjacency matrix ▣ (c) Bipartite layout ⊗

Figure 2: The three network representations investigated in the study. ● depicts nodes by dots connected by directional arrows. ▣ shows incoming edges row-wise and outgoing ones (column-wise). In ⊗, edges go from the nodes on the left axis to duplicates of nodes on the right axis. Visual patterns of clusters, chains, and nodes with high in-degree are apparent in all techniques.

the target nodes. To depict a network, the nodes are replicated and placed on both axes in the same order, and the links are shown by drawing lines connecting the source and target nodes. Therefore, in ⊗, the link direction always read from left to right (Figure 2c). In comparison to ● and ▣, ⊗ has a limited scalability with respect to the number of edges, due to the limited drawing space. To mitigate this problem, edge rendering [4, 18] or bundling techniques [39, 49] could be applied to improve the visibility of links in cluttered areas. Similar to ▣, ⊗ relies on vertex ordering algorithms [13, 40], not only to reveal the network structures, but also to reduce visual clutter. ⊗ combines some of the features of ▣ and ●. On the one hand, it encodes the connectivity information the same way as ● does. That is, by drawing lines connecting the source and target nodes. On the other hand, similar to ▣, the nodes are replicated and positioned on two axes representing the source and target axes. Having a separate drawing space for network nodes allows ⊗ and ▣ to avoid the problem of overlapping nodes existing in ●. Therefore, one could hypothesize that ⊗, similar to ▣, would be better suited than ● for tasks that are based on node attributes, such as node lookup or identifying node degree. Additionally, depicting each node twice—once as a source node and once as a target node—could make ▣ and ⊗ better suited than ● for visualizing directed networks. Despite being inherently built for visualizing directed networks, ⊗ and ▣ can also be used to visualize undirected networks as well. That is, by transforming them into directed ones (i.e., adding the edges in both directions).

Due to the replication of nodes on two separate axes, both ▣ and ⊗ are not well suited for path finding tasks. Both rely on memorizing node labels to solve the task, which might be challenging without the aid of interaction techniques [56], if the path is more than just one hop between the source and target nodes. In contrast, in ●, one does not need to read the node labels to find a path between two nodes because the path information is visually encoded by both the topological and geometrical properties of the nodes. In contrast to ▣, the node axes in ⊗ are rather parallel than orthogonal to each other. Such an arrangement makes ⊗ more scalable than ▣ with respect to network size. However, such scalability comes at the cost of restricting the drawing space to only the area between the two axes, making ⊗ the least scalable among the three representations with respect to network density.

Figure 2 shows three structural properties of real-world networks and their appearance in each representation. The cluster structure is easy to spot in ●, due to the two-dimensional spatial layout (Figure 2a). This pattern forms a rectangular shape along the diagonal (Figure 2b) in an ordered ▣ or as an hourglass-like shape (Figure 2c) in an ordered ⊗. Sparse, small-world networks exhibit chain-like structures as seen in Figure 2(a), at the top-right corner. This structure is translated into parallel lines in ⊗ or entries along the diagonal in ▣. Nodes with high in-degree are also common in networks. These nodes form fan-like shapes in ⊗ or appear as filled rows/columns in ▣. In this example, we use hierarchical clustering to order both ▣ and ⊗ (Section 4.3).

4 Methods

Many of the design choices we made throughout the study are based on a qualitative data study [55] that we performed at an early phase of the project. During this phase, we investigated different network properties, classes, and analysis tasks. We also experimented with different ways of rendering networks on each of the aforementioned representations, the use of color, and interactivity. This section details our methodological approach and explains the design choices we made.

4.1 Network Parameters

We visualize different networks with respect to density, size, and network class. An overview of how the stimuli change under varying parameters is displayed in Figure 4.

Density Profile Network density has a strong impact on the visualization. As described in the previous section, while ▣ scales very well with dense networks, ● and ⊗ suffer from overdrawing problems and therefore have limited scalability. Hence, it is relevant to our study to evaluate the different representations under different densities. We adopt a linear density definition with respect to edges and vertices $d = |E|/|V|$, which is considered to be a good descriptor of densities found in real-world networks [42]. In our study, we aim to evaluate six different densities $d = \{1,2,4,8,16,32\}$. To avoid a large number of trials during the study, we group density values into *density profiles* such that we have three profiles $d_{profile} = \{d_{low}, d_{mid}, d_{high}\}$, where $d_{low} = \{1,2\}, d_{mid} = \{4,8\}$, and $d_{high} = \{16,32\}$.

Network Size Most previous studies used networks with 100 nodes or less [62]. In our experiment, we consider networks with approx. 500 nodes for evaluating the overview tasks. Such size is comparable to large networks used in some of the previous studies [47, 48]. Yoghourdjian et al. [63] used networks with 1000 nodes. However, since our study does not involve the use of interaction techniques, including such size entails scaling down the resulting network image to fit the visualization to a standard computer screen. While this is true for all three representations, it could impair ▣, making the links hardly visible with low-density profiles. We opted for approx. 500 nodes to provide a fair comparison with ▣. For detailed tasks, we use networks of sizes 20 and 50, which is consistent with previous studies [28, 52].

Network Class In our study, we consider four network classes: Barabási [10], Erdős-Rényi [25], Watts-Strogatz [59], and FARZ [26]. Each class comes with different topological features. Barabási networks exhibit the *preferential attachment* property, where new nodes tend to get connected to the "popular" nodes in the network (i.e., scale-free networks). We call these nodes "the main hubs", as they are characterized by a high in-degree property. Watts-Strogatz networks exhibit small-world network properties: high clustering coefficient and short average path length. Erdős-Rényi networks have a low clustering coefficient as a result of the random connectivity between the nodes. FARZ networks exhibit modular structures (i.e., community structure), a property found in many real-world datasets [23, 26]. Our selection of the four classes overlaps with the work of Yoghourdjian et al. [63], who used Barabási, Erdős-Rényi, Watts-Strogatz, and Clustered-Barabási. The latter is a hybrid version of Barabási to emulate the community structure by loosely connecting two separate Barabási networks. In our case, we use FARZ for that purpose.

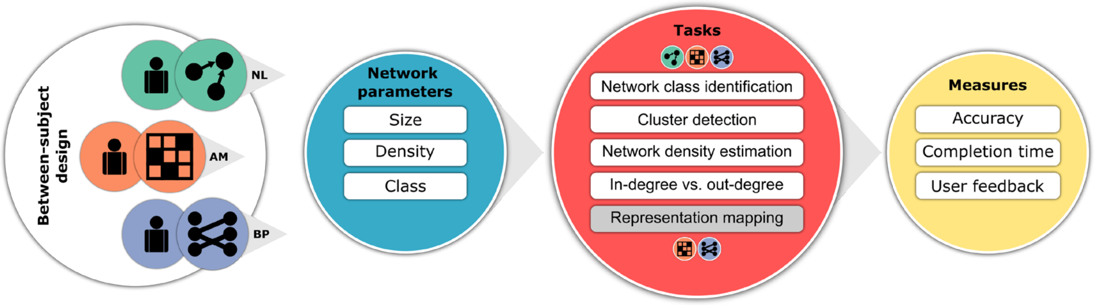


Figure 3: Study design and variables. We presented the visualizations in a between-subject design with a variation in network parameters. All participants performed a set of tasks with one visualization technique. We measured their performance and user feedback for evaluation.

4.2 Tasks and Hypotheses

We selected five tasks for evaluation. Tasks T1 – T3 are network overview tasks, whereas T4 and T5 are detailed tasks for directed networks. The choice of tasks and the formulation of the hypotheses were based on the literature [7, 38, 57, 58, 63] and our qualitative data study. In this section, we briefly explain each task and state our hypotheses along with the rationale behind them.

T1: Network Class Identification In this task, we assess the capability of each representation in depicting different network classes and the ability to maintain the visual characteristics of each class under varying densities. Since we want to rule out memory as a factor in the experiment, we designed this task as a matching task. During the experiment, we showed the participants one network image at the top and asked them: *To which class does the network belong?* At the bottom of the screen, we showed the participants a reference image for each network class to choose from. In other words, the participants had to find out which reference image is the most similar to the given network image at the top. To account for the density parameter, we paired different density profiles of the given network image with different density profiles of the reference images. Since we had three density profiles, this resulted in six different pairs. In total, there are 4 (classes) × 6 (density profile pairs) = 24 trials, per visualization condition. Our hypothesis is as follows:

H_1: **🟠 and 🔵 are more accurate than 🟢 for T1.**

As seen in Figure 4, 🟠 and 🔵 are able to maintain the visual characteristics of each network class irrespective of the density profile. In contrast, the visual characteristics seem to disappear as the density changes in 🟢. This can be seen in dense networks, where the hairball generated by Barabási does not differ much from the one generated by Erdős-Rényi. However, this kind of problem also occurs at low density, where it is hard to differentiate FARZ from Watts-Strogatz. Yoghourdjian et al. [63] arrived at a similar conclusion when they compared 🟠 against 🟢 across different network classes.

T2: Cluster Detection In this task, we assess the capability of each representation in revealing network clusters under varying densities. We only considered networks generated by the FARZ model (Section 4.1). In this task, there are two parameters to vary: the density profile $d_{profile}$ and the number of clusters $k = \{1,2,3,4,5,6,7,8\}$. For the low-density profile, we excluded density $d = 1$, since the generated networks are too sparse to form any cluster. We arranged the k values into four groups with consecutive values. In total, there were 3 (density profiles) × 4 (cluster groups) = 12 different combinations. For each, we sampled two networks, making 24 trials, per visualization. During the experiment, we showed the participants one network image and asked them: *How many clusters can you detect in this network?* Our hypothesis is as follows:

H_2: **🟢 and 🔵 are more accurate than 🟠 for T2.**

With respect to this task, Okoe et al. [47, 48] and Berardi et al. [14] found no significant difference in accuracy between 🟢 and 🟠. In the study by Nobre et al [46], 🟠 performed better than 🟢. However, the authors stated that the clusters were difficult to spot in 🟢, partially due to the large node sizes that were used to accommodate many of the node attributes (i.e., multivariate network). We expect 🟢 and 🔵 to obtain more accurate results than 🟠, especially with low-density profiles. As seen in Figure 4, the links are barely visible in 🟠 when the density profile is low. In contrast, links are more visible in 🟢 and 🔵 as a result of overdrawing. One could argue that 🟢 and 🔵 amplify the recognition of links in sparse networks, which will lead to a better detection of clusters.

T3: Network Density Estimation In this task, we measure the sensitivity of each representation to changes in network density. We also investigate whether such sensitivity depends on the underlying network density and/or the network class. During the experiment, we showed the participants a pair of network images from the same class side-by-side and asked them: *Which network has more connections (Left or Right)?* In addition to the network class and density profile parameters, we defined an additional parameter δ_d that accounts for the density difference between the network pairs. We evaluated δ_d for four values $\delta_d = \{0.25, 0.5, 0.75, 1.0\}$. This corresponds to one network having $\{25\%, 50\%, 75\%, 100\%\}$ more connections than the other. In total, there are 3 (density profiles) × 4 (δ_d values) × 4 (network classes) = 48 trials, per visualization condition. Our hypothesis is as follows:

$H_3(0)$: **There is no statistical significance in accuracy or speed between the three representation for T3.**

Yoghourdjian et al. [63] and Okoe et al. [47] evaluated the same task and found no statistical significance in accuracy between 🟢 and 🟠. The network properties were $n_{nodes} = 20$ in the former, $n_{nodes} = 258$ and $n_{links} = 1090$ in the latter. There was no information provided regarding the value of δ_d. We expect the density difference δ_d and the density profile $d_{profile}$ to have an influence on the results. With low-density profile and small density difference δ_d, 🟢 and 🔵 are expected to be more accurate than 🟠, due the low visibility of links in 🟠. In contrast, 🟠 is expected to be more accurate than 🟢 and 🔵 in high density and with small density differences δ_d, due to the edge overdrawing in 🟢 and 🔵. Regarding the network class, the density comparison task on 🟠 and 🔵 is expected to be more accurate for Erdős-Rényi and Barabási than for FARZ and Watts-Strogatz. As seen in Figure 4, Erdős-Rényi networks are characterized by a uniform distribution of links. This makes the density comparison task relatively easy, since one does not have to pay attention to the network details but rather to the overall picture. In Barabási networks, participants are expected to pay attention to the part of the network where the "main hubs" are located (i.e., the top part of the visualization). In the case of FARZ and Watts-Strogatz, the networks are composed of several clusters or sub-communities and, therefore, the comparison task involves inspecting the density within each cluster separately, which makes the task harder, and therefore, participants are expected to be less accurate, especially in 🟢.

T4: Node In-degree Vs. Out-degree Comparing in- and out-degrees of nodes is one of the most common tasks on a directed network. In contrast to the previous tasks, we considered a relatively small network ($n_{nodes} = 50$). While we originally planned to evaluate the task

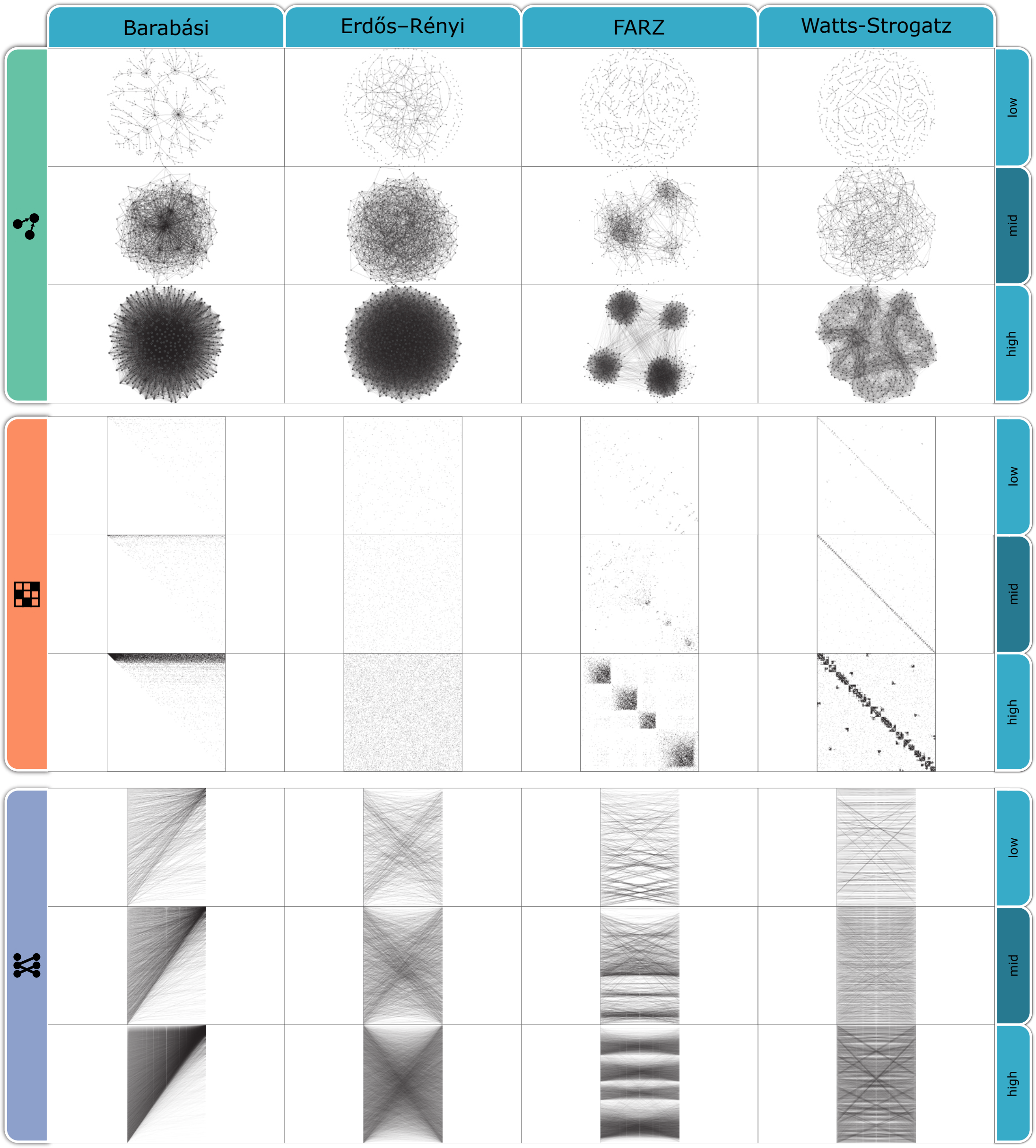


Figure 4: Overview of different network classes presented in the study. For the three visualizations ⁖, ▦, and ⤨, we presented visual stimuli of each class with different densities. Some tasks become more difficult with decreasing density of the data (e.g., detecting clusters in FARZ).

on large networks ($n_{nodes} = 500$), we found out that solving the task is not realistic without the use of interaction techniques. We avoided including interaction features since they would have introduced another degree of freedom in the experiment, and it would have been quite hard to implement interactive features without favoring one representation over the others. In this task, there are two parameters: $d_{profile}$ and the network class. We excluded the Barabási class as it is trivial to solve. Therefore, there are 3 (density profiles) $\times$ 3 (network classes) $\times$ 3 (repetitions) = 27 trials, per visualization condition. During the experiment, we showed the participants one network image where one node is randomly highlighted and asked them: *Does the highlighted node have more incoming or outgoing links?* Our hypothesis is as follows:

H_4: **▦ is more accurate than ⤨ and ⁖ for T4.**

⁖ and ⤨ both suffer from overdrawing problems, whereas ▦ does not. Nevertheless, since ⤨ provides two separate axes for source and target nodes, the participants are expected to be more accurate in ⤨ than in ⁖. We expect $d_{profile}$ to influence the results. For mid-density and high-density profiles, ⁖ would require the participants to visit the two endpoints of each link to determine whether it is bidirectional or not, which is a more laborious and error-prone process than for ⤨ and ▦. In contrast, with low-density profiles, we expect no differences, which is supported by findings from previous work [32, 35]. We expect the network class to not have an influence on the results as this is a detailed task that does not take into account the overall structure of the network, which is more amplified by highlighting the node under question. We decided not to use color to encode direction information in ⁖, as it would have introduced a bias into the experiment toward one representation. Instead, we reduced the size of the arrows and altered their shape to enhance their recognition in cluttered areas.

T5: Representation Mapping The design of this task was inspired by the work of Yoghourdjian et al. [63] and Kriglstein et al. [36]. In this task, we want to measure which representation — ▦ or ⤨ — is "easier" to map to ⁖. We used ⁖ as a reference representation, since it is known to be an intuitive and easy-to-understand representation of networks. Therefore, we hypothesized that a network representation is easy to understand if it could be mapped "easily" to ⁖. We used small

Table 1: Summary of tasks and presented stimuli.

	n_{nodes}	$d_{profile}$						Network class				Trials		
		low		mid		high								
		1	2	4	8	16	32	Barabási	Erdős	FARZ	Watts	🟢	🟧	⊗
T1: Class Identification	500	●	●	●	●	●	●	●	●	●	●	24	24	24
T2: Cluster Detection	500		●	●	●	●	●			●		24	24	24
T3: Density Estimation	500	●	●	●	●	●	●	●	●	●	●	48	48	48
T4: Node In-degree Vs. Out-degree	50	●	●	●	●	●	●		●	●	●	27	27	27
T5: Representation Mapping	20	●						●		●	●	-	12	12
												123	135	135

($n_{nodes} = 20$) and sparse ($d = 1$) networks and varied the network class parameter. We intentionally left out the node labels to force participants into looking at the network structure rather than doing a one-to-one mapping. We also excluded Erdős-Rényi networks since pilot testing showed that they were particularly hard to solve due to the absence of distinctive structure properties, compared to other classes (Figure 4). During the experiment, we showed the participants one network image in ⊗ or 🟧 representation at the top and provided them with two 🟢 diagrams at the bottom and asked them: *Which* 🟢 *diagram corresponds to the given representation at the top?* In total, there are 3 (network classes) × 4 (repetitions) = 12 trials, per visualization condition. Our hypothesis is as follows:

H_5: **⊗ is more accurate than 🟧 for T5.**

Without node labels, we expect this task to be generally challenging to solve regardless of the underlying visual representation. This was also confirmed during pilot testing. However, since 🟢 and ⊗ use the same visual variables to encode nodes and links, we hypothesize that it would be easier to solve the task in ⊗ than 🟧. In 🟧, participants will have to count the rows and columns to uniquely identify the nodes which would be more laborious, even with small networks. Out of the three classes, we expect Barabási to be the easiest to solve since it is characterized by the existence of the "main hub" nodes, which would be easier to spot in either representation.

4.3 Stimuli Data

We generated stimuli data for each task. Table 1 gives an overview of the different network parameters used in each task. When a task involves multiple repetitions (i.e., T4 and T5), a new network is generated for each repetition. We decided on generating synthetic networks in order to have fine-grained adjustability of network parameters. Task T1 revolves around being able to discern separate network classes, which requires relatively objective classes in the study stimulus. Real-world data often exhibit traits of multiple network classes, making it hard to find a ground truth for the task. Additionally, task T2 revolves around finding the number of clusters in the provided network. This requires us to have an objective measure of exactly how many clusters are present in a given network. This is trivial with a generated dataset, as the number of clusters is a generation parameter (i.e., k) of the FARZ network model. With real-world data, it is often unclear how many clusters are present. We set the k parameter to 4 for tasks T1 and T3, 1 for tasks T4 and T5, and $\{1,2,3,4,5,6,7,8\}$ for task T2. We re-implemented the FARZ model [26] and used the *igraph R package* [20] to generate the data for the other classes. We used the *d3-force* [1] algorithm to layout the 🟢 diagram. Our experience [5] shows that *d3-force* obtains results that are comparable with the best-of-breed layout algorithms such as neato or sfdp. To order the vertices of 🟧 and ⊗, we used agglomerative hierarchical clustering [2] with average linkage. To calculate the similarity between two nodes, we use the Jaccard distance based on the sets of direct neighbors. Hierarchical clustering is widely used in the literature and known to provide good results [13, 60].

For 🟢, we applied the most common technique for graph visualization with solid dots for vertices and straight lines with arrowheads for edges (Figure 2a). For 🟧, the columns were used to depict the outgoing links while the rows depicted the incoming ones. In that sense, the link direction always reads top to left (Figure 2b). The visual stimuli were rendered as static images with no interactivity. We omitted the use of color to focus on the spatial layout of the different techniques. We used the opacity variable in general to increase the visibility of links in 🟢 and ⊗. It was also used to highlight the nodes in task T4. Finally, we downscaled the images by pixel averaging, to make sure they could fit into a browser window.

4.4 Study Design and Procedure

We used a between-subject study design with three conditions, one per representation (Figure 3). For a large number of participants and variety of demographics, we conducted an online crowdsourcing user study using Amazon Mechanical Turk (mTurk). Similar to prior work [19, 33, 47], we aimed for 50 participants per condition. The order of presented stimuli was counter-balanced between the conditions. In the case of ⊗ and 🟧, participants had to perform five tasks, and only four (excluding T5) in the case of 🟢. The order of the tasks was assigned randomly.

At the beginning of the study, participants were presented with a demographics questionnaire. After that, participants read a brief tutorial that explained the basics of network visualization and gave an introduction to the visualization being evaluated. Participants could browse through the tutorial back and forth until they felt ready to perform the tasks. Each task started with basic instructions and examples, followed by a few training trials where participants had a chance to see the correct answer. When participants were ready, they performed the actual trials. For each trial, we recorded the participants' answers and response times. After all trials of one task, we asked participants to rate the difficulty of the task on a 5-point Likert scale from "Very Easy" to "Very Difficult" and inform us about the strategy they followed to solve the task. Both questions were required fields. The study had in total 135 trials for 🟧 and ⊗ and 123 trials for 🟢 (Table 1). The study web-application was built using client-server architecture using PHP, JavaScript, and jsPsych [24]. We refer to the supplemental materials for the source code, data files, and screenshots of the study [5].

4.5 Study Participants and Piloting

We required participants to be at least 18 years old, speak fluent English, and have a screen-resolution of at least 1920 × 1080 px. The first and second conditions were self-reported. The third was verified automatically using the jsPsych library. To ensure a high quality of responses on mTurk, we set a Human Intelligence Task (HIT) success rate $\geq 97\%$ and a total number of accepted HITs ≥ 1000. Furthermore, we added several attention-check trials throughout the study. We filtered out participants who did not pass the attention-check trials or those who took less than 10 minutes to complete the study. In total, we recruited 150 online users after applying our filtering criteria (35% female). The majority of the participants (44%) had an age between 31 and 40 years, followed by 19% above 50 years of age. 65% of the participants were located in North America, followed by 17% located in Asia.

Our pilot testing consisted of two phases. First, we recruited three visualization experts to test-run the study in an informal laboratory setting. The objective of this phase was to verify the clarity of the tasks and the usability of the user interface and to get an initial estimation of the time and the difficulty of each task. During the testing, we encouraged the participants to think aloud and ask questions. In the end,

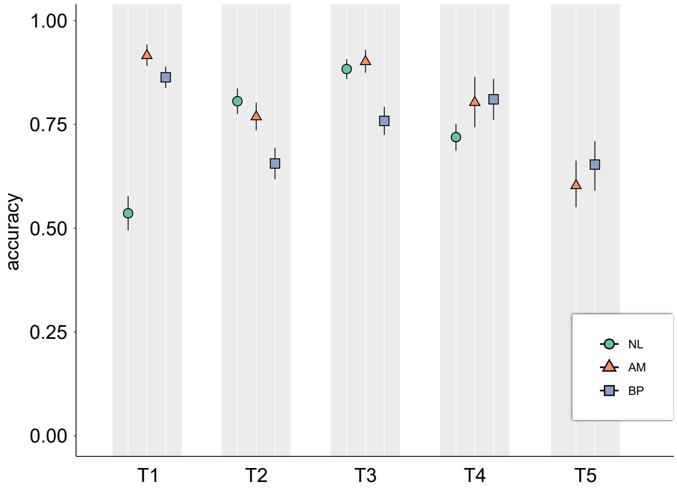


Figure 5: The overall accuracy of tasks T1 – T5. The shapes represent the means, and the error bars their 95% CIs.

Table 2: Results of the overall statistical evaluation for the Kruskal-Wallis and pairwise Wilcoxon tests. p-values for multiple comparisons were Bonferroni corrected. We also report the confidence intervals of the mean difference [21, 22].

Hypothesis	Vis Pair	*p-value*	M_{diff}	CI_{diff}
T1: H_1(✓) $\chi^2(2) = 96.357, p < 0.001$	⊗ - 🟢	<0.001	0.33	[0.28, 0.38]
	🟧 - ⊗	0.0031	0.05	[0.02, 0.09]
	🟧 - 🟢	<0.001	0.38	[0.33, 0.43]
T2: H_2 (✗) $\chi^2(2) = 42.342, p < 0.001$	🟢 - ⊗	<0.001	0.15	[0.10, 0.20]
	🟧 - ⊗	<0.001	0.11	[0.06, 0.16]
	🟢 - 🟧	0.23	0.04	[-0.01, 0.08]
T3: $H_3(0)$ (✗) $\chi^2(2) = 51.782$, p<0.001	🟢 - ⊗	<0.001	0.13	[0.08, 0.17]
	🟧 - ⊗	<0.001	0.14	[0.10, 0.19]
	🟧 - 🟢	0.21	0.02	[-0.02, 0.05]
T4: H_4 (✓) $\chi^2(2) = 19.54, p < 0.001$	⊗ - 🟢	<0.001	0.09	[0.03, 0.15]
	⊗ - 🟧	1	0.01	[-0.07, 0.08]
	🟧 - 🟢	<0.001	0.08	[0.02, 0.15]
T5: H_5 (✗) $W = 1411$, *p-value* $= 0.264$	⊗ - 🟧	0.264	0.04	[-0.04, 0.12]

we held an open discussion where we asked the participants for general feedback. In addition to shortening the text, fixing the typos, and fine-tuning the user interface, one of the main results of this phase was to exclude the Erdős-Rényi class from task T5 as participants showed signs of frustration or quickly gave up when they were presented with Erdős-Rényi stimuli compared to other classes. The second piloting phase involved running a test batch of 20 participants on mTurk. The objective of this phase was not only to test the functionality of the study web application but also to ensure the quality of the responses and obtain a more accurate estimation of the length of the study. The main outcome of this phase was adding the attention-check trials. Our pilot testing showed that the study lasted for approximately 45 minutes. We compensated the participants with a rate of $12 per hour.

5 Results

To investigate the overall hypothesis, we do statistical testing with Kruskal-Wallis and Wilcoxon Signed Rank tests, as the data was not normally distributed. In addition, we provide 95% confidence intervals (CIs) to enrich the evaluation with an estimation-based approach. We also use this estimation-based approach for further exploratory investigations of the network parameters, as recommended by Cumming [21]. As the network parameters were within-subject, we also normalize the between-subject variability [41, 44]. Figure 5 shows an overview of the accuracy of the different representations across the tasks in our study and Table 2 reports the results of the statistical analysis. The raw data and the source code used for statistical analysis are provided in the supplemental materials [5].

T1 As seen in Figure 5, we can confirm H_1 (✓), see also Table 2. For 🟢 ($M = 0.54, SD = 0.15, CI = [0.49, 0.58]$), the task was solved less accurately compared to 🟧 ($M = 0.92, SD = 0.09, CI = [0.89, 0.94]$) and ⊗ ($M = 0.86, SD = 0.09, CI = [0.84, 0.89]$). We also looked at the effect of network class and density pairing as seen in Figure 6a and Figure 6b. 🟢 seems to perform worst on Erdős-Rényi, while ⊗ seems best for Barabási. Further, low-low density pairings seem considerably worse than high-high density pairings for 🟢.

T2 We cannot confirm H_2 (✗). ⊗ ($M = 0.66, SD = 0.13, CI = [0.62, 0.69]$) performs worse than 🟧 ($M = 0.77, SD = 0.12, CI = [0.74, 0.80]$), and there is a small difference for 🟢 ($M = 0.81, SD = 0.11, CI = [0.78, 0.84]$) and 🟧. Figure 6c shows that 🟢 works best with 3 or 4 clusters, compared to the other representations. ⊗ performs worst for low-density profiles (Figure 6d).

T3 We can reject $H_3(0)$ (✗), as we see an effect of network representation. ⊗ ($M = 0.76, SD = 0.12, CI = [0.72, 0.79]$) leads to worse accuracy compared to AM ($M = 0.90, SD = 0.10, CI = [0.87, 0.93]$) and 🟢 ($M = 0.88, SD = 0.08, CI = [0.86, 0.91]$). ⊗ performs worse than 🟧 and 🟢 for high-density profiles (Figure 6f). Similarly, ⊗ performs worse than 🟢 and 🟧, when looking at network classes. It seems particularly bad for Barabási (Figure 6e). For the delta parameter δ_d, we could not see an effect.

T4 We can only partially confirm H_4 (✓). 🟧 ($M = 0.80, SD = 0.21, CI = [0.74, 0.86]$) is better than 🟢 ($M = 0.72, SD = 0.11, CI = [0.69, 0.75]$), but on par with ⊗ ($M = 0.81, SD = 0.17, CI = [0.76, 0.86]$). The results for the network class parameters reflect the overall trend and show that the task in the 🟢 condition was solved less accurately for Watts-Strogatz compared to 🟧 and ⊗ (Figure 6g). Density-wise, 🟢 performs worst on high-density profiles (Figure 6h).

T5 As reflected in Table 2, we did not find a significant effect of the representation for T5, so we cannot accept H_5 (✗). As there was no main effect, we also did not look deeper into the effect network class parameters might have.

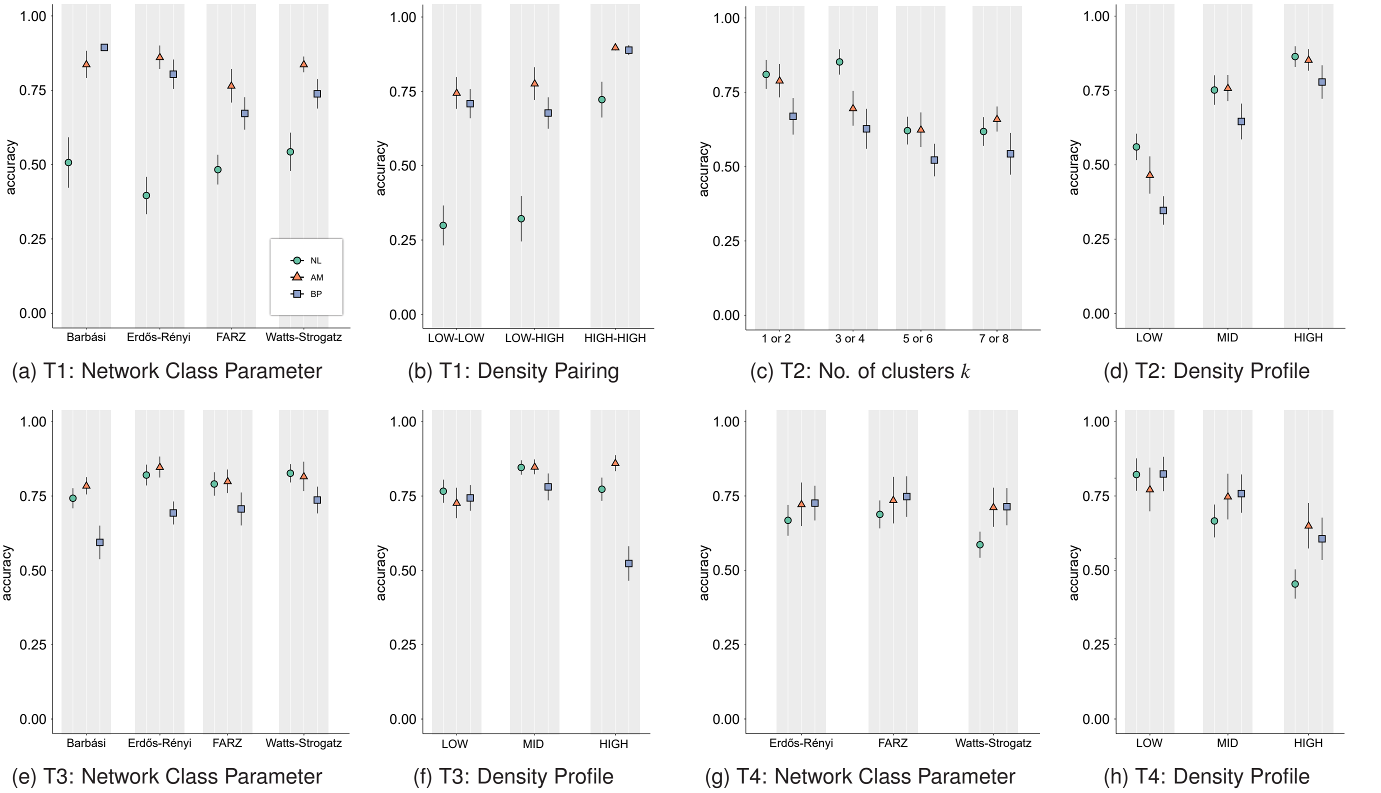
6 Discussion

Based on our results, we discuss the individual tasks, limitations, and finally, formulate guidelines for the use of the visualizations.

6.1 Task Performance

The results provide valuable information about how well the individual visualizations were suited to solve the tasks and what strategies were often applied by the participants. In this section, we discuss the performance results and shed light on the participants' subjective assessment of the task difficulty, along with their commentary feedback.

T1 Both 🟧 and ⊗ are able to maintain the visual characteristics of each network class under varying densities. This is especially true when the network is sparse and the reference images are dense or vice versa (i.e., low-high density pairing). 🟢 seems to suffer when the given image or the reference images are of low density (Figure 6b). While both 🟧 and ⊗ provide high accuracy, the results show that the classes seem to look more distinctive in 🟧, except for Barabási networks (Figure 6a). We assume that this is due to the low visibility of links in 🟧 when the network density is low, which might impair the recognition of class characteristics. In contrast, for ⊗, such characteristics would be amplified on low-density networks. From the qualitative feedback, we could identify that for 🟢, density comparison was mentioned frequently: *"... I tried to focus on density, or where many connections came together and then the overall shape."* *"I ... paid attention to how the nodes and edges were placed and tried to extrapolate how the network would look at different stages of denseness..."* The terms *difficult* and *hard* were mentioned more often than for the other techniques. Overall, 🟧 and ⊗ were considered easier to compare: *"(about 🟧)...I matched the pattern*



(a) T1: Network Class Parameter (b) T1: Density Pairing (c) T2: No. of clusters k (d) T2: Density Profile

(e) T3: Network Class Parameter (f) T3: Density Profile (g) T4: Network Class Parameter (h) T4: Density Profile

Figure 6: The accuracy of each representation plotted with respect to different task parameters. The shapes represent the means, and the error bars are the 95% confidence intervals. The accuracy is calculated by averaging first over the repetitions per task and then averaging over the mean per participant with respect to the considered parameter.

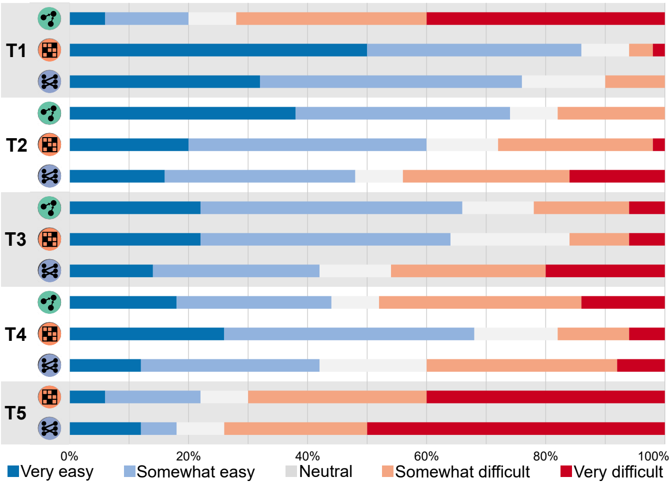


Figure 7: Subjective assessment of task difficulty.

I saw, regardless of density...”; *“(about ⊗) I just looked at the overall pattern. It seemed like it was either a diagonal, an x-shape, bands, or solid.”* This impression is also reflected in task difficulty (Figure 7).

T2 We hypothesized that ● and ⊗ will be more accurate than ▦. The results could not support our hypothesis. We noticed that denser networks increase the accuracy, regardless of the representation (Figure 6d). However, the results show that ⊗ is more affected than ● and ▦ when the network density is low. As it appears, the visibility of links plays a less important role than the proximity of nodes, when it comes to detecting clusters in sparse networks. This makes ⊗ and ▦ less suited than ● because both representations rely on link visibility to reveal cluster structure when the nodes are ordered in a proper way (Figure 6d). However, the results suggest that the overdrawing in ⊗ might have impaired the recognition of clusters, in comparison to ▦. This is also reflected in the participants' comments. It was frequently mentioned that the task became hard to solve with ⊗ for sparse networks. *“My strategy was to look for the darker lines as they seemed to be a cut-off point for one cluster. This was not always evident or obvious in the more sparse clusters.”*; *“I tried to count how many bands there were. It was a little hard for sparser networks.”* A similar approach was mentioned for ▦: *“I looked for the spots that were most dense and counted them.”* However, difficulties with sparse networks were less frequently mentioned than for ⊗. For ●, many participants reported that the task was easy and they did not provide further details on their strategies. By investigating the influence of the number of clusters in relation to the accuracy, we found that participants tended to underestimate the number of clusters in the network more than overestimate them (Figure 6c). Nevertheless, both effects occurred with low-density and mid-density profiles. This is understandable since the network size is constant while the number of clusters k varies.

T3 In general, ● and ▦ are significantly more accurate than ⊗. The results suggest that ⊗ has the least sensitivity to density changes. The density plot (Figure 6f) shows that ● and ⊗ are slightly better than ▦ at low densities, whereas ▦ outperforms both at high densities. However, ⊗ seems to be impacted at high densities. ⊗, and to a lesser degree ●, rely on transparency to reduce visual clutter. As such, the denser the network, the lighter and smoother the lines will appear. For a non-trained eye, that might be misleading. The participants' comments support this: *“(about ⊗) I tried to focus more on the white space and darkness of the images overall.”*; *“For the ones that had the darker images, I looked at the darkest area and checked to see how smooth it was...”* In contrast, ▦ relies on the *density*, which increases the overall contrast of the image, making the comparison task more intuitive. This is also reflected in the participants' comments: *“(about ▦) I chose the square that was more dense or darker.”* With respect to the network class, Figure 6e shows that for ▦ and ●, Erdős-Rényi obtained the most accurate results among the four classes, which is consistent with what we expected. However, the performance did not degrade that much for FARZ and Watts-Strogatz. This suggests that changes in the overall structure of the network did not impair the participants from finding which network has more connections. For 🔵, the results in Figure 6e suggest that the task was harder to solve on Barabási in comparison to the other classes. One explanation could be that the participants focused more on the top part of the diagram where the main hubs are located and neglected other parts of the network.

T4 🟢 is not well suited to solve this task, especially when the network density is high. Contrary to our expectation, the performance of 🔵 seems to be on par with 🟠, suggesting that visual clutter did not impair the participants from estimating the node degree (Figure 6h). Looking at the participants' comments revealed slightly different strategies to solve the task on 🔵 versus 🟠. For 🔵, the words "estimating" or "squinting" was often mentioned: *"...some squinting on very dense clusters. A good bit of counting on more sparse graphs where you could make out individual lines..."*;*"I tried to estimate how many were on each side. It helped to blur my vision slightly and look at the density of the lines."* For 🟠, participants mentioned: *"I counted the black squares unless they were a lot..."*;*"I counted the empty white squares..."*

T5 Although the results show that 🔵 is more accurate than 🟠, the difference was not significant (Figure 5). While there are few participants who managed to obtain 100% accuracy, the majority found this task hard to solve, regardless of the representation (Figure 7). We assume that most participants did not spend enough time solving this task. The data revealed that, on average, participants spent about 30 sec per trial. Only 9 participants in 🔵 and 6 participants in 🟠 managed to obtain an accuracy of more than 83%. The data revealed that these participants tended to spend one minute on average for each trial. Looking at the comments reveals some strategies. 🔵 was investigated to identify missing and existing connections: *"I checked the points that had no connections and also those points that made multiple connections to make it easier to identify the correct option."*; *"Check how many solo nodes there were and then compare to the diagram. Also checking nodes that had a lot of connections coming in..."* A similar behavior was mentioned for 🟠: *"I tried to match up the empty rows/columns with how many outliers there were, and if necessary counted the largest amount of connections..."* A short attention span is a common pitfall of crowd-sourced evaluation [8]. We believe that the performance of this task might have been better if it had been done in a controlled-lab setup.

With respect to the comparison between 🟢 and 🟠, our findings are consistent with the previous work with respect to task T1 [63], T2 [14,47,48], and T3 [47,48,63], despite the fact that authors evaluated smaller networks. With respect to task T4, our findings partially match the previous work [32,35] in low-density profiles. However, we found that 🟠 performs significantly better than 🟢 in dense networks.

6.2 Limitations

The results of our crowd-sourced study provide new insights for the visualization of network data. Regarding the number of participants, we are on par with other work [33,47]. However, we identified some limitations for the interpretation of the results.

Crowd-sourced Evaluation Crowd-sourced evaluation has the benefit of large sample sizes but also comes with problems [8]. Our pilot testing showed that the study needs about 45 min for a novice participant (a participant who has no prior knowledge of network visualization) to complete. To ensure the validity of our results, we set our filtering criteria based on the time spent to complete the study but also based on the successful responses to the attention checks we put throughout the study. We originally recruited 240 unique online users to take our experiment, but only 150 (62%) passed our filtering criteria. This shows that some of the participants were not paying attention during the study, which had to be handled later by the experimenters.

Network Size In our study, we fixed the network size parameter to not increase the evaluations' complexity. For the overview tasks, we chose size 500 as it appeared to be the fairest for the three representations, considering that we do not include interactive features in the evaluation. Based on our experience, we argue that the study findings would still hold for 🟢 and 🔵 at a network size of 1000. But it might be different for 🟠. It is an open question how each representation would scale as the network grows even beyond 1000 nodes.

Network Representation In our experiment, we tried to be fair in selecting the layout methods and clutter-reduction techniques. This can explain why we did not include edge bundling techniques in the evaluation. While edge bundling would certainly be beneficial for 🟢 and 🔵, it does not have a direct mapping in the 🟠 representation, and the question of how edge bundling influences the appearance of network structures would require more investigation that we deemed beyond the scope of the study. For layout methods, we used the *d3-force* [1] algorithm to lay out 🟢 and hierarchical clustering [2] to order the nodes in 🟠 and 🔵 representations. Our selection of these methods was driven by their wide adoption in the community [13, 15] and the availability of their implementations in many languages/tools. Nevertheless, there is still room for improvement to ensure the fairest comparison between the different representations [45].

Selected Tasks The tasks evaluated in this study are not meant to be an exclusive list. During the data study, we experimented with eight different tasks. We decided to only include five tasks to not extend the study beyond a realistic length. The other three tasks that we did not include are: *identifying common neighbors between two nodes*, *looking up a node*, and *finding a path between two nodes*. In contrast to the overview tasks we included in the study, these three are detailed network tasks. While these tasks are certainly valid candidates for further evaluation, we decided to focus on network overview tasks, since they were not often considered by other studies [62].

6.3 Guidelines

In this work, we evaluated three overview tasks (T1 – T3) and two detailed tasks (T4 and T5) for directed networks. Depending on the task at hand, one representation might perform better than others. For tasks that involve comparing the overall network structure, 🟠 and 🔵 might be better candidates than 🟢 . Both representations are versatile enough to depict the various structural properties and maintain these properties under varying densities. In contrast, when the task involves identifying network clusters or sub-communities, 🟢 seems to be superior to both representations, since the cluster information is double encoded by the proximity and connectivity between the nodes. Due to the overdrawing problem, 🔵 does not seem to be a good choice when the task is about estimating overall density or detecting subtle changes in connectivity. In such situations, 🟠 provides the most accurate estimation of the true density within the network. Finally, for tasks that are based on node attributes, 🟠 and 🔵 offer a better overview, especially in the context of directed networks, as both representations have two separate axes for encoding source and target nodes. Overall, 🟠 seems to be the most reliable among the three representations across the different tasks.

7 Conclusion

We presented a comparative user study about node-link diagrams, adjacency matrices, and bipartite layouts for the visualization of network data. By assessing the performance in five different tasks, we derived a set of guidelines for using the techniques with respect to the task and network properties at hand. Overall, this study shows the strengths and limitations of established graph visualization techniques and provides insights into when a bipartite layout might help interpret the data. In the future, we want to investigate solution strategies for the tasks in detail. Eye tracking methodology might help understand how people read the visualizations and provide more information about conscious and subconscious processes than post-experimental self-reports. Further, we plan to extend the catalog of tasks to derive a more comprehensive set of guidelines for using the visualizations.

Acknowledgments

This work is supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2120/1 – 390831618 and – Project ID 251654672 – TRR 161.

References

[1] GitHub - d3/d3-force: Force-directed graph layout using velocity verlet integration. https://github.com/d3/d3-force. (Accessed on 03/27/2022).

[2] hclust function - RDocumentation. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/hclust. (Accessed on 06/29/2022).

[3] M. Abdelaal, M. Hlawatsch, M. Burch, and D. Weiskopf. Clustering for stacked edge splatting. In *Proceedings of the Conference on Vision, Modeling, and Visualization*, p. 127–134, 2018. doi: 10.2312/vmv.20181262

[4] M. Abdelaal, A. Lhuillier, M. Hlawatsch, and D. Weiskopf. Time-aligned edge plots for dynamic graph visualization. In *24th International Conference Information Visualisation (IV)*, pp. 248–257, 2020. doi: 10.1109/IV51561.2020.00048

[5] M. Abdelaal, N. D. Schiele, K. Angerbauer, K. Kurzhals, M. Sedlmair, and D. Weiskopf. Supplemental materials for: Comparative evaluation of bipartite, node-link, and matrix-based network representations, 2022. doi: 10.18419/darus-3100

[6] 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*, p. 483–492, 2013. doi: 10.1145/2470654.2470724

[7] R. Amar, J. Eagan, and J. Stasko. Low-level components of analytic activity in information visualization. In *IEEE Symposium on Information Visualization.*, pp. 111–117, 2005. doi: 10.1109/INFVIS.2005.1532136

[8] D. Archambault, H. Purchase, and T. Hoßfeld. *Evaluation in the Crowd. Crowdsourcing and Human-Centered Experiments*. Springer, 2017. doi: 10.1007/978-3-319-66435-4_1

[9] A. S. Asratian, T. M. Denley, and R. Häggkvist. *Bipartite Graphs and Their Applications*. Cambridge University Press, 1998. doi: 10.1017/CBO9780511984068

[10] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. doi: 10.1126/science.286.5439.509

[11] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. The state of the art in visualizing dynamic graphs. In *EuroVis (STARs)*, 2014. doi: 10.2312/eurovisstar.20141174

[12] F. Beck, M. Burch, C. Vehlow, S. Diehl, and D. Weiskopf. Rapid serial visual presentation in dynamic graph visualization. In *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 185–192, 2012. doi: 10.1109/VLHCC.2012.6344514

[13] M. Behrisch, B. Bach, N. Henry Riche, T. Schreck, and J.-D. Fekete. Matrix reordering methods for table and network visualization. *Computer Graphics Forum*, 35(3):693–716, 2016. doi: 10.1111/cgf.12935

[14] C. W. Berardi, E. T. Solovey, and M. L. Cummings. Investigating the efficacy of network visualizations for intelligence tasks. In *IEEE International Conference on Intelligence and Security Informatics*, pp. 278–283, 2013. doi: 10.1109/ISI.2013.6578843

[15] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011. doi: 10.1109/TVCG.2011.185

[16] M. Burch, M. Hlawatsch, and D. Weiskopf. Visualizing a sequence of a thousand graphs (or even more). *Computer Graphics Forum*, 36(3):261–271, 2017. doi: 10.1111/cgf.13185

[17] M. Burch, W. Huang, M. Wakefield, H. C. Purchase, D. Weiskopf, and J. Hua. The state of the art in empirical user evaluation of graph visualizations. *IEEE Access*, 9:4173–4198, 2021. doi: 10.1109/ACCESS.2020.3047616

[18] M. Burch, C. Vehlow, F. Beck, S. Diehl, and D. Weiskopf. Parallel edge splatting for scalable dynamic graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2344–2353, 2011. doi: 10.1109/TVCG.2011.226

[19] P. Chapman, G. Stapleton, P. Rodgers, L. Micallef, and A. Blake. Visualizing sets: An empirical comparison of diagram types. In T. Dwyer, H. Purchase, and A. Delaney, eds., *Diagrammatic Representation and Inference*, pp. 146–160, 2014. doi: 10.1007/978-3-662-44043-8_18

[20] G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal Complex Systems*, 1695(5):1–9, 2006.

[21] G. Cumming. The new statistics: Why and how. *Psychological Science*, 25(1):7–29, 2014. doi: 10.1177/0956797613504966

[22] G. Cumming and S. Finch. Inference by eye: Confidence intervals and how to read pictures of data. *American Psychologist*, 60(2):170, 2005. doi: 10.1037/0003-066X.60.2.170

[23] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008–P09008, 2005. doi: 10.1088/1742-5468/2005/09/p09008

[24] J. R. De Leeuw. jsPsych: A javascript library for creating behavioral experiments in a web browser. *Behavior Research Methods*, 47(1):1–12, 2015. doi: 10.3758/s13428-014-0458-y

[25] P. Erdös and A. Rényi. On random graphs I. *Publicationes Mathematicae*, 6:290–297, 1959.

[26] J. Fagnan, A. Abnar, R. Rabbany, and O. R. Zaiane. Modular networks for validating community detection algorithms. *arXiv*, 2018. doi: 10.48550/arXiv.1801.01229

[27] J.-D. Fekete. Reorder.js: A javascript library to reorder tables and networks. IEEE VIS 2015, 2015. Poster.

[28] M. Ghoniem, J. D. Fekete, and P. Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. *IEEE Symposium on Information Visualization*, pp. 17–24, 2004. doi: 10.1109/INFVIS.2004.1

[29] 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

[30] N. Henry and J.-D. Fekete. MatLink: Enhanced matrix visualization for analyzing social networks. In *Human-Computer Interaction – INTERACT 2007*, pp. 288–302, 2007. doi: 10.1007/978-3-540-74800-7_24

[31] N. Henry, J.-D. Fekete, and M. J. McGuffin. NodeTrix: A hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1302–1309, 2007. doi: 10.1109/TVCG.2007.70582

[32] M. Hlawatsch, M. Burch, and D. Weiskopf. Visual adjacency lists for dynamic graphs. *IEEE Transactions on Visualization and Computer Graphics*, 20(11):1590–1603, 2014. doi: 10.1109/TVCG.2014.2322594

[33] 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, 2014. doi: 10.1109/TVCG.2014.2315995

[34] Z. Jin, N. Chen, Y. Shi, W. Qian, M. Xu, and N. Cao. TrammelGraph: visual graph abstraction for comparison. *Journal of Visualization*, 24(2):365–379, 2021. doi: 10.1007/s12650-020-00706-2

[35] 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

[36] S. Kriglstein, M. Pohl, and J. Doppler Haider. How users transform node-link diagrams to matrices and vice versa. In *Diagrammatic Representation and Inference*, pp. 526–534, 2018. doi: 10.1007/978-3-319-91376-6_48

[37] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale. Empirical studies in information visualization: Seven scenarios. *IEEE Transactions on Visualization and Computer Graphics*, 18(9):1520–1536, 2012. doi: 10.1109/TVCG.2011.279

[38] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, p. 1–5, 2006. doi: 10.1145/1168149.1168168

[39] 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

[40] I. Liiv. Seriation and matrix reordering methods: An historical overview. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 3(2):70–91, 2010. doi: 10.1002/sam.10071

[41] G. R. Loftus and M. E. Masson. Using confidence intervals in within-subject designs. *Psychonomic Bulletin & Review*, 1(4):476–490, 1994. doi: 10.3758/BF03210951

[42] 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*, p. 1–7, 2006. doi: 10.1145/1168149.1168167

[43] K. Misue. Drawing bipartite graphs as anchored maps. In *Proceedings of the Asia-Pacific Symposium on Information Visualisation*, pp. 169–177, 2006.

[44] R. D. Morey. Confidence intervals from normalized data: A correction to Cousineau (2005). *Tutorials in Quantitative Methods for Psychology*, 4(2):61–64, 2008. doi: 10.20982/tqmp.04.2.p061

[45] A. Noack. Modularity clustering is force-directed layout. *Physical Review*

E, 79:026102, 2009. doi: 10.1103/PhysRevE.79.026102

[46] C. Nobre, D. Wootton, L. Harrison, and A. Lex. Evaluating multivariate network visualization techniques using a validated design and crowdsourcing approach. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, p. 1–12, 2020. doi: 10.1145/3313831.3376381

[47] M. Okoe, R. Jianu, and S. Kobourov. Revisited experimental comparison of node-link and matrix representations. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 287–302, 2018. doi: 10.1007/978-3-319-73915-1_23

[48] M. Okoe, R. Jianu, and S. Kobourov. Node-link or adjacency matrices: Old question, new insights. *IEEE Transactions on Visualization and Computer Graphics*, 25(10):2940–2952, 2019. doi: 10.1109/TVCG.2018.2865940

[49] G. Palmas, M. Bachynskyi, A. Oulasvirta, H. P. Seidel, and T. Weinkauf. An edge-bundling layout for interactive parallel coordinates. In *IEEE Pacific Visualization Symposium*, pp. 57–64, 2014. doi: 10.1109/PacificVis.2014.40

[50] N. Pezzotti, J.-D. Fekete, T. Höllt, B. P. Lelieveldt, E. Eisemann, and A. Vilanova. Multiscale visualization and exploration of large bipartite graphs. *Computer Graphics Forum*, 37(3):549–560, 2018. doi: 10.1111/cgf.13441

[51] H. C. Purchase. Performance of layout algorithms: Comprehension, not computation. *Journal of Visual Languages and Computing*, 9(6):647–657, 1998. doi: 10.1006/jvlc.1998.0093

[52] D. Ren, L. R. Marusich, J. O'Donovan, J. Z. Bakdash, J. A. Schaffer, D. N. Cassenti, S. E. Kase, H. E. Roy, W. Y. Lin, and T. Höllerer. Understanding node-link and matrix visualizations of networks: A large-scale online experiment. *Network Science*, 7(2):242–264, 2019. doi: 10.1017/nws.2019.6

[53] J. Sansen, R. Bourqui, B. Pinaud, and H. Purchase. Edge visual encodings in matrix-based diagrams. In *19th International Conference on Information Visualisation*, pp. 62–67, 2015. doi: 10.1109/iV.2015.22

[54] H.-J. Schulz, M. John, A. Unger, and H. Schumann. Visual analysis of bipartite biological networks. In *Eurographics Workshop on Visual Computing for Biomedicine*, pp. 135–142, 2008. doi: 10.2312/VCBM/VCBM08/135-142

[55] M. Sedlmair, A. Tatu, T. Munzner, and M. Tory. A taxonomy of visual cluster separation factors. *Computer Graphics Forum*, 31(3pt4):1335–1344, 2012. doi: 10.1111/j.1467-8659.2012.03125.x

[56] Z. Shen and K.-L. Ma. Path visualization for adjacency matrices. In *Proceedings of the 9th Joint Eurographics / IEEE VGTC Conference on Visualization*, pp. 83–90, 2007. doi: 10.5555/2384179.2384192

[57] F. van Ham and B. Rogowitz. Perceptual organization in user-generated graph layouts. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1333–1339, 2008. doi: 10.1109/TVCG.2008.155

[58] C. Vehlow, F. Beck, and D. Weiskopf. Visualizing group structures in graphs: A survey. *Computer Graphics Forum*, 36(6):201–225, 2017. doi: 10.1111/cgf.12872

[59] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998. doi: 10.1038/30918

[60] L. Wilkinson. The grammar of graphics. In J. E. Gentle, W. K. Härdle, and Y. Mori, eds., *Handbook of Computational Statistics: Concepts and Methods*, pp. 375–414. Springer, 2012. doi: 10.1007/978-3-642-21551-3_13

[61] P. Xu, N. Cao, H. Qu, and J. Stasko. Interactive visual co-cluster analysis of bipartite graphs. In *IEEE Pacific Visualization Symposium (PacificVis)*, pp. 32–39, 2016. doi: 10.1109/PACIFICVIS.2016.7465248

[62] V. Yoghourdjian, D. Archambault, S. Diehl, T. Dwyer, K. Klein, H. C. Purchase, and H.-Y. Wu. Exploring the limits of complexity: A survey of empirical studies on graph visualisation. *Visual Informatics*, 2(4):264–282, 2018. doi: 10.1016/j.visinf.2018.12.006

[63] 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*, 24(12):3081–3095, 2018. doi: 10.1109/TVCG.2018.2790961