This paper is very nicely written, particularly for a group of first year students. Nevertheless, the writing shows some typical novice writer errors pertaining argumentation, readability and academic style. You will be able to find in-text comments aiming to address a variety of common errors found in student writing. It is important that you become aware of similar issues in your own writing as addressing such issues will significantly improve the quality of your work.

We wish you happy writing!

Guiding Graph Exploration by Visual Patterns from Several Layouts and Reorderings

Submission ID 1225

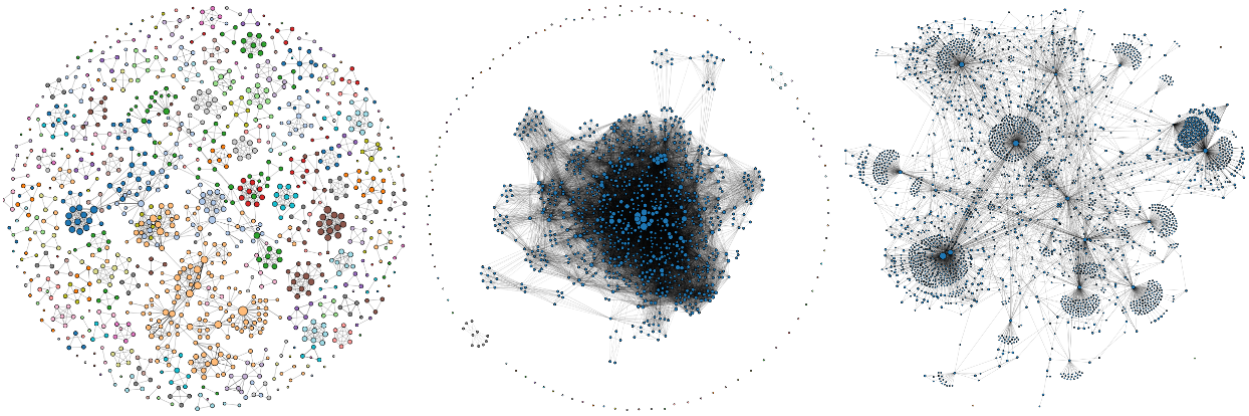


Figure 1: *Node-link diagrams generated from real-world datasets: the leftmost is made from co-authorship data, the middle one from co-citation data, and the rightmost one from a CPAN distributions dataset [cpa19]. All of the datasets contain several hundred to thousands of nodes and edges and some of them carry edge weights.*

Abstract

Visualizing graphs is a challenging task due to the various properties of the underlying relational data. For sparse and small graphs the perceptually most efficient way are node-link diagrams whereas for dense graphs with attached data, adjacency matrices might be the better choice. Since graphs can contain both properties, being globally sparse and locally dense, a combination of several visualizations is beneficial. In this paper we describe a visually and algorithmically scalable approach to provide views and perspectives about graphs as interactively linked node-link as well as adjacency matrix visualizations. ~~For illustrative purposes we support several node-link layouts and matrix reorderings from a large repertoire.~~ The novelty of the technique is that insights like clusters or anomalies from one or several combined views can be used to influence the layout or reordering of the others. Moreover, the importance of nodes and node groups can be detected, computed, and visualized by taking into account several layout and reordering properties ~~in combination~~ as well as different edge properties for the same set of nodes. We illustrate the usefulness of our web-based tool by applying it to graph datasets like co-authorships, co-citations, and a CPAN distribution. ~~Finally, we discuss limitations and scalability issues.~~

CCS Concepts

• ***Human-centered computing*** → ***Graphical user interfaces; Visualization; Graph drawings;***

1. Introduction

Graph data occurs in various application fields like call dependencies in software engineering [BMR*12], friendship relations in social networks [HF06, HFM07, HF07], areas of interest connections in eye tracking data [BBR*14], or traffic situations in road networks [GBD09].

Visually exploring such data requires advanced visual metaphors, in the best case interactively linking several of those to benefit from the positive effects of all of them. For example, node-link diagrams [DETT99] are useful for small and sparse graphs (see Figure 1) whereas adjacency matrices [EDG*08] are best suited when it comes to large and dense networks [GFC05, OJK19]. However, using only one concept may

lead to degradations of performance at some task [RLMJ05] since a combined and linked view provides many more insights in case they are linked in a clever way and interactions are supported to filter the graph data based on several visual perspectives.

To give even more ways to find insights than just one fixed visualization technique we provide several node-link layouts [SLH*18] as well as several matrix reordering techniques [BBR*16]. The node-link diagrams follow aesthetic graph drawing criteria [Pur97] while the adjacency matrices support the finding of different grouping and clustering patterns depending on the user tasks and which reordering strategy is requested. All of the views and parameter adaptations can be selected on users' demand but the tool can also automatically suggest one, based on graph data properties. Our novel linked visualization strategy provides ways to adapt a view based on insights from other views, for example clusters found in the adjacency matrix can be used to guide the layout of the node-link diagram and vice-versa.

We illustrate our web-based tool by means of applying it to several real-world datasets that consist of up to several thousand vertices and edges. Those datasets are based on co-author, co-citation, and CPAN graph data. We show the patterns that we found by applying the techniques in combination and we show the interactions that are integrated to filter the data and to support the building, confirming, refining, or rejecting of hypotheses based on user defined tasks [KAF*08].

Our novel technique allows interactively selecting clusters in several views and letting the algorithm compute the intersection of node sets. The output is a filtered version of the graph that contains all nodes ~~that occur~~ for all requested properties. This gives a hint about the importance of a node or node group based on algorithmic concepts like layouts and reorderings but additionally enhanced by visual depictions of the output of such algorithms in combination.

The novelty in this work compared to the MatrixExplorer [HF06] can be described in the following points:

- **Visualization-based cluster and outlier detection:** Clusters and groups of related vertices can be automatically detected based on the visual properties given by the layout of a node-link diagram or the positions of nodes in an adjacency matrix. A similar strategy can be applied for outliers like isolated nodes.
- **Combined cluster information:** Clusters or outliers can be identified in several views. This visual information can be used to link the identification processes for nodes that occur in clusters based on several layouts or matrix re-arrangements. Hereby we support two modes: 1. intersection/common nodes and 2. union/all occurring nodes. The intersection gives 'highly' clustered nodes, i.e. nodes that belong together in all layout and reordering categories. The union only gives all node candidates, i.e. it filters for nodes that get never clustered. The visual cluster size can be manually selected or given by density or proximity.
- **Web-based:** Providing a web-based visualization makes it easier to get started, without the need to install software or libraries. Moreover, the data can be shared with other researchers as well as the visual results as some kind of dissemination process or even as collaborative interaction.
- **Scalability for vertices and edges:** Applying a pixel-based representation supports the identification of clusters and groups in very large graphs, although the matrix reorderings can take quite some time until a suitable clustering result is generated.

Finally, we discuss limitations and scalability issues focusing on algorithms, visualizations, interactions, and perceptual aspects by the human users [War04, War08]. Although graph visualization is a well-known and well-researched discipline we will outline still existing future challenges and white spots.

2. Related Work

Graphs have been introduced by Leonhard Euler [Eul41] trying to find a solution to the graph-theoretic problem known as the 'Seven Bridges of Königsberg'. To visually illustrate the problem he used the visual metaphor of node-link diagrams [DETT99]. Several years ago they were pretty powerful due to the fact that the underlying graph data was small and sparse, consisting of only a few vertices and edges (without edge weights attached).

In these days, graphs are typically huge and handling, preprocessing, and managing graph data is already a big challenge [EK18]. However, visualizing the relational data [vLKS*11] as a node-link diagram in a naive way leads to visual clutter [RLMJ05], 'a state in which excess items or their disorganization lead to a degradation of performance at some task'. Even advanced layout algorithms [KW01] can only partially solve this problem since the immense size and density of the relational data does not allow to follow individual outliers or anomalies. In most cases those are occluded by dense graph regions, for example clusters of nodes.

Adjacency matrices on the other hand are quite powerful visual concepts [EDG*08]. They allow thousands of vertices to be represented as well as all the weighted and directed edges in-between. However, adjacency matrices suffer from problems when following paths, which is on the other hand also problematic for node-link diagrams if the graphs exceed a certain size [GFC05, KEC06]. Moreover, an unordered adjacency matrix will not show any structure and hence, advanced matrix reordering strategies must be supported [BBR*16], also guided by a human user [BSP20].

Hybrid representations have been developed trying to combine the benefits of both concepts, node-link diagrams and adjacency matrices [HFM07, HF07], however, following a direct integration of both concepts does not support several independent layouts and reorderings, at least it becomes difficult. On the other hand such a combination might hide insights that might be seen in the node-link diagrams or adjacency matrices when represented separately. Consequently, we follow the original approach by Henry and Fekete [HF06] in their MatrixExplorer tool. Although their approach is already equipped with various features we further add the concept of exploiting visual properties from formerly laid out and reordered node-link diagrams and adjacency matrices to select and filter nodes that guide a new layout.

In this paper we focus on displaying a graph in several views supporting the human observer by interactively changing the views, layouts, and reorderings on demand, inspired by a multiple visual metaphors approach [Bur15, Bur17]. For this reason, we take into account insights from the views and parameter configurations while

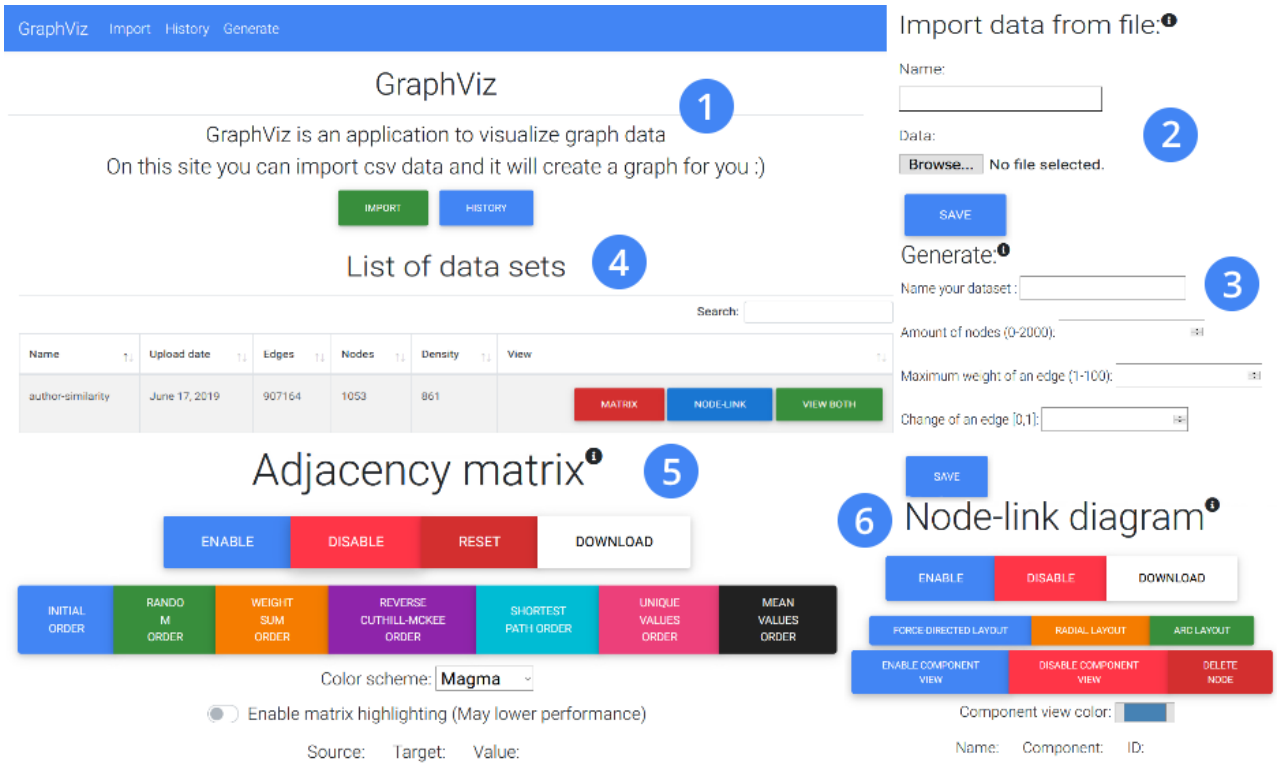


Figure 2: *The GUI of our website showing the homepage of our website (1), as well as the pages, where users can upload datasets (2), generate random datasets (3), see the uploaded datasets (4), and view the adjacency matrix and node-link diagram (5), (6).*

also linking and highlighting the insights. With this strategy the observer has more abilities to explore a graph from different perspectives, even if it is pretty dense and needs to be represented in a more structured way first.

3. Data Model and Preprocessing

We describe a data model for graphs, how a layout and a reordering can be generated and which processes and algorithms are involved, as well as how visual graph properties can be combined and linked to influence other views and graph visualizations.

3.1. Graph Data

A graph $G = (V,E)$ consists of a finite number of vertices $V := \{v_1,\ldots,v_n\}$ and a finite number of edges $E := \{e_1,\ldots,e_m\} \subseteq V \times V$. The edges might be weighted, meaning a weight function is attached, mapping each edge to a real-valued number, $f : E \longrightarrow \mathbb{R}$. If the direction and the weights of the edges play a role we denote this graph by the term network.

3.2. Layouts and Reorderings

A layout $\Gamma(G)$ of a graph G is a positioning of the given vertices V to n (normally) distinct locations in the display space whereas the edges in-between are drawn as either straight, curved, orthogonal [HIvWF11], or partial links [BVKW11, BCG*17], depending on several aesthetic graph drawing criteria to be followed [WPCM02].

A reordering can be applied to an adjacency matrix [BBR*16]. This follows a certain property function that produces a rearrangement of the vertices in a way that a certain (typically user-defined) property holds, for example, showing a cluster structure at the diagonal.

3.3. Combining Graph Properties

The novelty of the technique is that insights like clusters or anomalies from one view can be used to influence the layout or reordering of the other. Moreover, the importance of nodes and node groups can be detected, computed, and visualized by taking into account several layout and reordering properties in combination. The first step towards this direction is that several node-link layouts and adjacency matrix reorderings of the same graph dataset are computed and then the visual properties of the produced diagrams are taken into account to build an intersection or union set of the nodes under investigation. This principle helps to identify certain node groups that belong together under different circumstances, meaning their relation among each other is stronger than if just one representation is taken into account.

The identification of clusters works ~~hereby~~ in two stages: First several algorithms are applied to a graph dataset and secondly

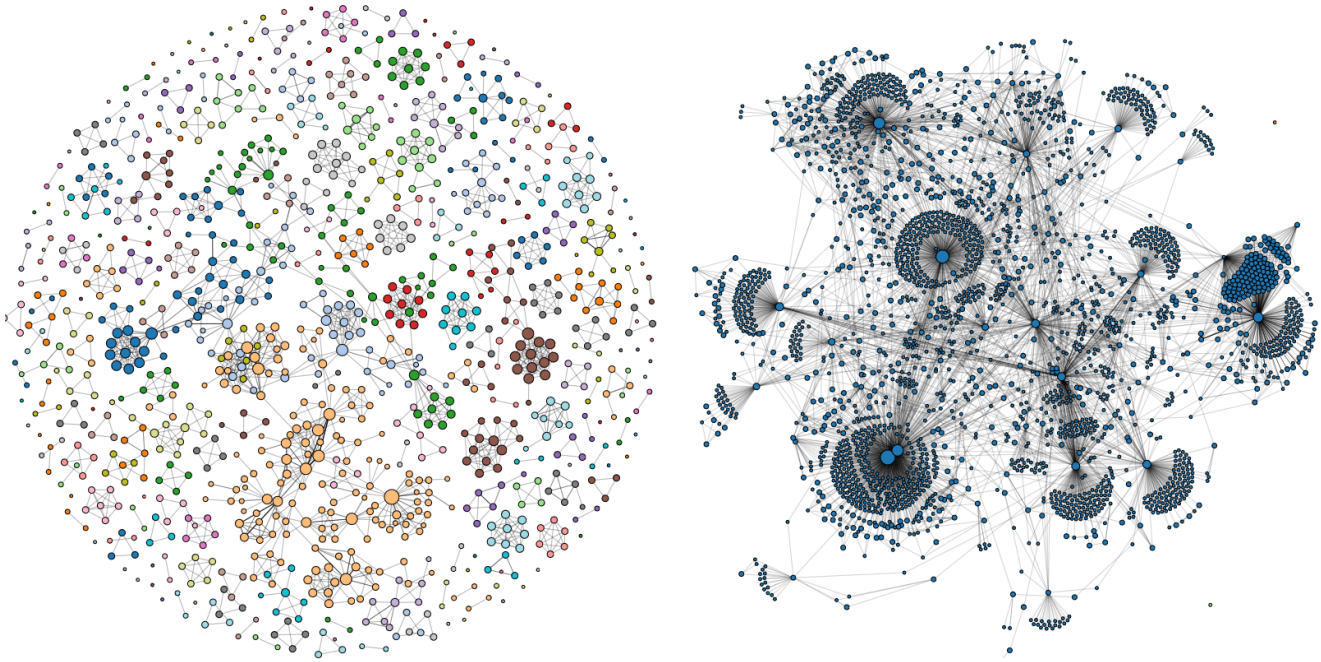


Figure 3: *On the left: Force-directed node-link diagram of a co-authorship dataset with 1,053 nodes and 3,504 edges. On the right: Force-directed node-link diagram of the CPAN distributions dataset with 2,724 nodes and 7,669 edges.*

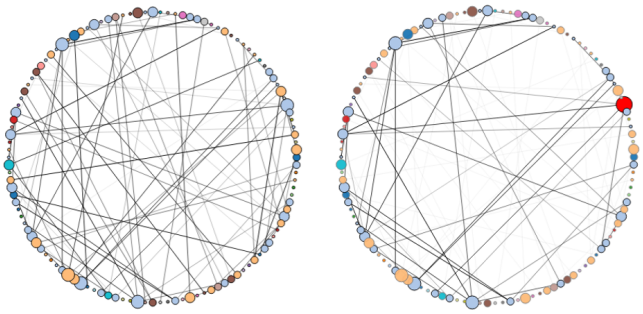


Figure 4: *The radial layout of the node-link diagram of the randomly generated dataset - on the left is the initial radial layout and on the right is the filtered radial layout.*

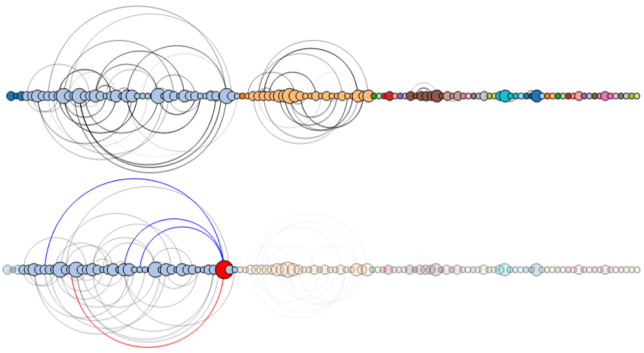


Figure 5: *The arc layout of the node-link diagram of the randomly generated dataset - on top is the initial arc-diagram and on the bottom is the filtered arc diagram.*

the visual outcomes of these algorithms are combined to detect strongly clustered nodes or nodes that do not fall in any of the clusters, which might considered 'real' outliers. Such a feature is useful as a filtering function and has several benefits compared to the standard 'one visualization' filtering.

Moreover, we can apply an automatic node group detection based on node group densities. This can be done for both, the node-link layouts as well as the adjacency matrices. Although this is a useful approach, it is best if the human users are involved in the node detection process equipped with their perceptual abilities to detect visual patterns rapidly [War04].

4. Visualization Tool

This section explains how the tool can be used, how we set up the server, how the visualization techniques and interactions were designed, and which algorithms we implemented.

4.1. Graphical User Interface (GUI)

The design of the visualization tool follows the major principle of making it easy-to-understand. Figure 2 shows the main web page and gives a short introduction for the user of the tool. In (1) we provide the starting point of an exploration process where data can be imported or even a history of the already taken steps can be

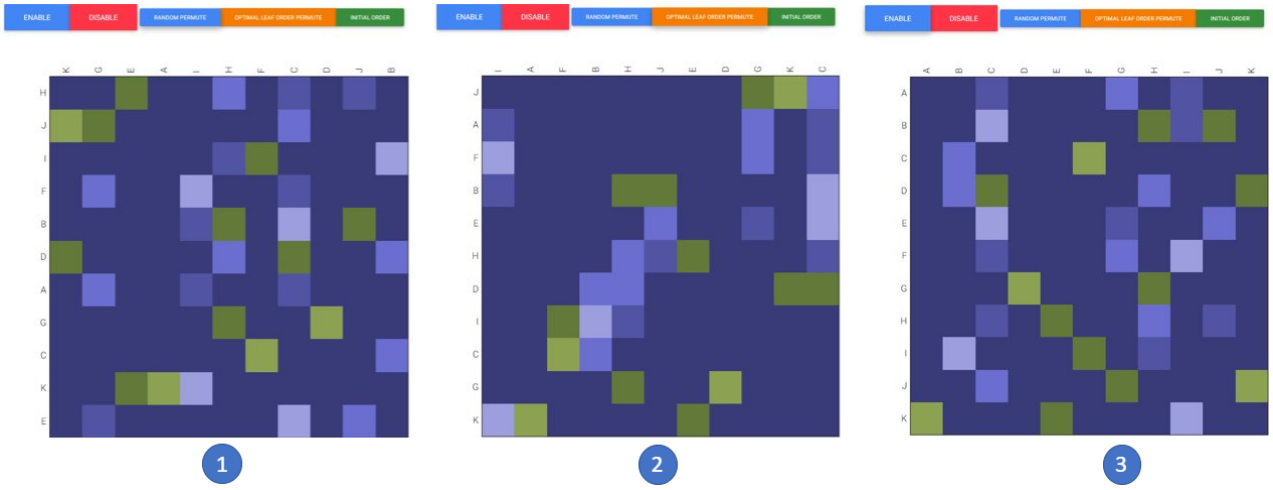


Figure 6: *Three reorderings of the same graph visualized as an adjacency matrix.*

requested. On the toolbar of the website there are the same buttons as shown in the GUI.

In (2) we can import data by choosing a file to upload while in (3) a random graph dataset can be generated just for testing purposes. In (4) users can choose among all the different datasets stored in the database and view it as a node-link diagram, an adjacency matrix, or both.

Adjacency matrices are shown in (5) while the corresponding node-link diagrams are depicted in (6). All the views can be downloaded as an image and by doing a reset they can be transformed into the original view. The node-link diagrams support several layouts and the adjacency matrices several reorderings.

4.2. Server Side

We make use of a Django server that runs on a virtual machine using the Ubuntu operating system on a local personal computer. The virtual server uses 2 cores of the computer's CPU running on 2.0 GHz, 1 GB of RAM, and internet speed of around 500 Mbps.

In order to make sure that all datasets imported to the server can be seen by all users, the server holds a database that any user can query on simply by adding "/items/id" in the URL, or by going through the GUI on the Items page and selecting the dataset needed.

If a file is uploaded the server saves the entry in the database (with a new ID and the name entered in the form). If it is a csv file it is converted into a json file. Finally, if a user wants to display this entry the server sends the JS and CSS scripts to the HTML page and the corresponding json file, meaning the D3 script can render it in the user's web browser.

4.3. Node-Link Diagrams

Our tool provides three layouts for a node-link diagram: a force-directed [FR91], a radial [BB07], and an arc layout [GBD09]. The force-directed layout presents the larger clusters close to the center and the smaller ones and single isolated nodes close to the border of the diagram (see Figure 3). This helps to better identify outliers than if those are located inside a hairball-like node-link diagram.

The radial layout displays the nodes aligned as a circle in which each node has an equal distance to the center and edges represented as chords connecting the nodes (see Figure 4). The radial layout is useful for seeing the density of a graph in local regions as well as the connections between nodes.

The arc layout aligns the nodes on a straight line, with edges represented as arcs connecting the nodes (see Figure 5). The arc layout has the advantage of being able to highlight components if the node order is optimized.

Figure 7 shows ways to interact with node-link layouts: The first one uses an HTML canvas to render each layout of the whole graph very fast, but the HTML canvas does not have arrowheads to indicate direction or a detailed view of every node. This was implemented using a D3 example of a force-directed layout as an inspiration for our layout and a basis for the other layouts. The tool has implemented features such as coloring different components with different colors and nodes with a higher number of incoming edges that appear larger. The second one uses an SVG to render the graph in detail, this version does include arrowheads and a highlight tool but is not very performance-friendly especially for larger graphs, which is why we only use the SVG for small parts of the graph.

4.4. Adjacency Matrices

Currently, the adjacency matrix is displayed as a pixel map, where the links are shown as filled grid cells where the corresponding connected nodes meet. Each cell's color depends on the weight of the edge - edges with a higher weight are lighter compared to edges with a lower weight (~~with the~~ default color scheme). In the tool, we implemented five reordering strategies, a random reordering and an option to go back to the original ordering (see Figure 6 for three reordering examples in a simple adjacency matrix visualization).

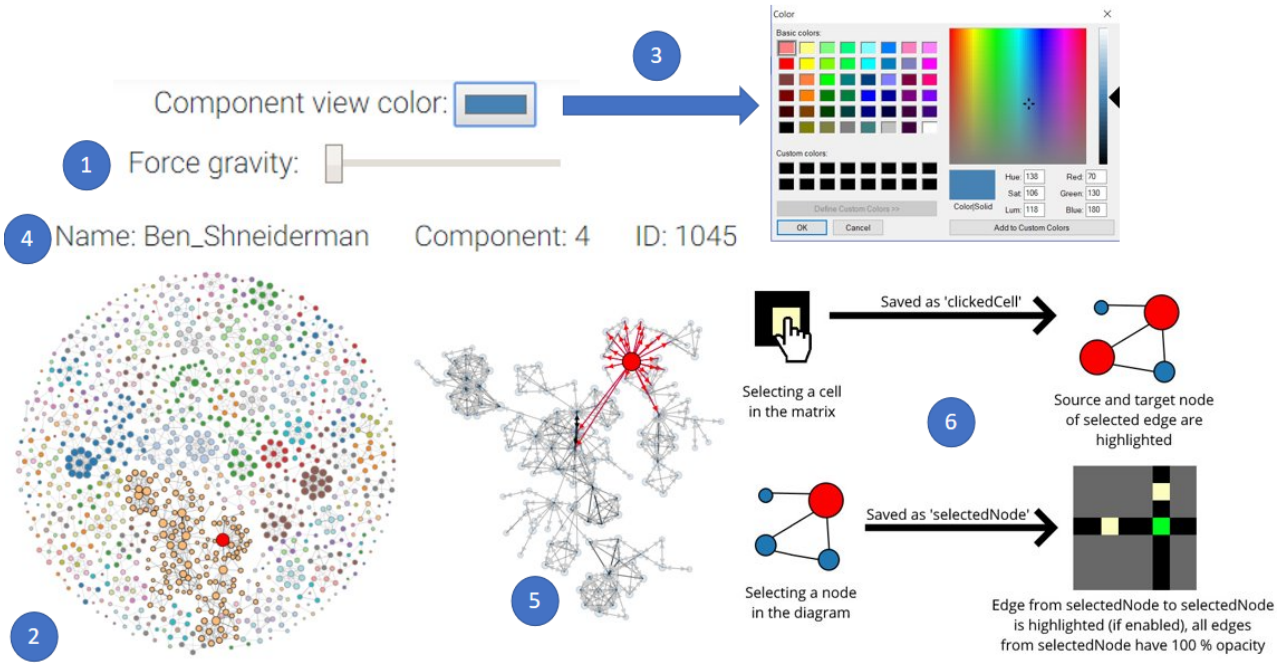


Figure 7: *Summary of the interactions using the co-authorship dataset with 1,053 vertices and 3,504 edges. We support interaction techniques like select, reconfigure, encode, elaborate, filter, and connect.*

- **Weight Sum Order:** This ordering uses the sum of weights in each of the rows and columns to reorder the matrix. These sums are then sorted and we return the list of row and column permutations as indices.
- **Reverse Cuthill-McKee Order:** To implement the reverse Cuthill-McKee algorithm, which is based on bandwidth reduction, we used the SciPy python library. In order to achieve this, we converted our matrix data frame into a CSR matrix and then ran the algorithm on it. This returns a permutation of the rows, which we use to reorder both the rows and the columns of the matrix returning block patterns.
- **Shortest Path Order:** The shortest path ordering was implemented using SciPy's Dijkstra's algorithm using Fibonacci Heaps to find the shortest path between the nodes in the dataset. The matrix is converted to a CSR format, then Dijkstra's algorithm is executed.
- **Unique Values Order:** This strategy reorders the matrix based on the number of unique weights in the dataset. The algorithm makes a copy of the matrix and adds a row and a column with the number of unique weights within a data frame. After this, we sort the data frame by the number of unique weights of the rows and then the columns. Finally, we get the permutation of the rows and columns from this sorted data frame.
- **Mean Values Order:** The mean value is used for each row and column to reorder the matrix. The algorithm used to achieve this makes a copy of the matrix and adds a new row and column with computed mean values for each row and column, this is all done within a data frame. The data frame is then sorted based on the mean values of the rows and columns. Finally, we get the permutation of the rows and columns from this sorted data frame.
- **Random Permutation Order:** This ordering rearranges the matrix randomly. This is done by making an array of all the row and column names. Then this array is split in two so that there are different orderings for the rows and columns. Then, the lists are shuffled using the Fisher-Yates algorithm and used to rearrange the columns and rows of the original matrix.

4.5. Interaction Techniques

Our tool has implemented the following interactions (see Figure 7) based on the visualization interactions presented in the paper by Yi et al. [YaKSJ07].

- **Select**: When the users click a node in the node-link diagram the node is selected and changes its color to red. Users can select edges on the adjacency matrix by clicking on cells in it. This is done by a function that checks if a node is selected and if so it updates the color.
- **Explore**: Users are offered panning and the ability to zoom in and out on the matrix. Panning works by clicking on the matrix and dragging it with the mouse. Zooming in and out can be done with the mouse wheel. The node-link diagram's SVG view also allows users to zoom in and out with the mouse wheel. Another form of possible zooming is to change the nodes' distance using a slider, which makes the graph also change size.
- **Reconfigure**: Each layout features a slider (1), which rearranges the data based on the distance between the nodes. For example, if a dataset is originally presented in a shrunken configuration,

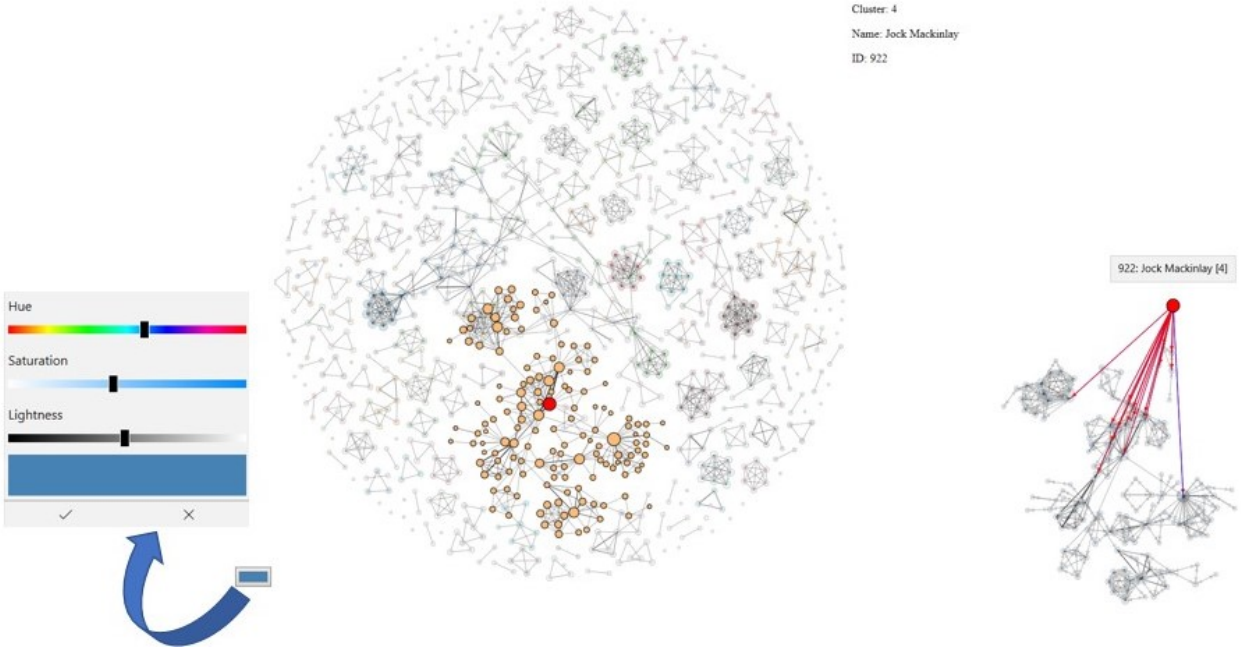


Figure 8: *Selecting a certain cluster of nodes to take a closer look at details, for example, a node that is important in this cluster.*

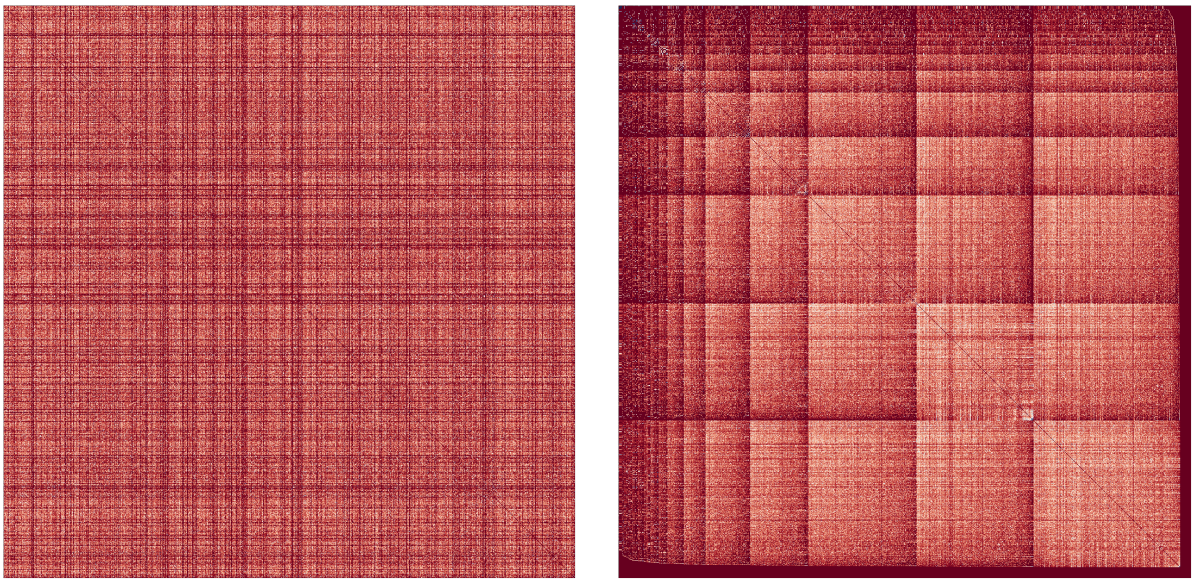
users can increase the distance between the nodes and analyze the data better. To implement this, the value set by the slider is being used to change the distance between the nodes, the force gravity, or the circle radius in the functions that draw the node-link diagram.

- **Encode**: The node-link diagram colors the nodes belonging to the same component with the same color (2). This way the user can easily differentiate between components. Users are also offered a button (3) to change the hue, saturation, and brightness of the nodes in the SVG view. This can be done using the information about components in the json file of the dataset and D3's commands to set the color of the nodes.
- **Elaborate**: When the users select a node in the node-link diagram they can see the node's id, name, and component (4). If the users select an edge in the matrix they can see information about its source, target, and weight. This is possible because the information can be extracted from the json file of the data for each node or edge using D3.
- **Filter**: When the users select a component by clicking on a node belonging to it in the canvas view (5), an SVG view displays only this component next to the canvas view. On the SVG view, the user can see incoming/outgoing edges displayed with arrowheads. A D3 example was used as inspiration for our SVG view. Additionally, when users select nodes in the radial or arc layouts of the node-link diagram the selected component is highlighted.
- **Connect**: When users select an edge in the matrix, the corresponding nodes linked by this edge are colored red in the node-link diagram (6). If highlighting is enabled when users select a node in the node-link diagram the matrix will highlight all edges connected to this node. This is being done using global variables which store information about the nodes and edges, and a select function that detects if a node or edge is selected and updates the information of these variables based on the selected object.

5. Application Example

We illustrate the usefulness of our visualization tool by applying it to an author similarity dataset and show an adjacency matrix, with two datasets for the force-directed layout of the node-link diagram containing CPAN distributions and co-authorship data. It will also use a generated small dataset to showcase the radial and arc layout as a node-link diagram. Different datasets will be used because adjacency matrices with a high number of edges are more interesting to study, but the node-link diagram struggles with rendering graphs with a high number of edges and the radial and arc layouts are more useful for smaller datasets.

The co-authorship data includes 1,053 nodes and 3,504 edges. Once the users have uploaded the dataset, they can view it as a node-link diagram. The force-directed layout of the node-link diagram displaying the co-authorship dataset is shown on the left in Figure 3. Users can see that bigger connected components are displayed in the center of the visualization, while smaller ones are displayed relatively closer to the border of the visualization and single isolated nodes are even closer to the border. Also, nodes with a larger amount of incoming edges appear bigger compared to others. Using this information it can be concluded that the single nodes can be considered as outliers and ignored, hence they do not flow into the selection of the most important and strongly connected nodes. Lastly, it can be observed that every connected component of the graph has its own color, making it easy for the user to differentiate between the different components of the graph. Selecting individual components from the node-link layouts can hence lead to useful



(a) (b)

Figure 9: *Two reorderings of adjacency matrices: (a) The adjacency matrix of the initial author-similarity dataset with 1,053 nodes and 907,164 edges. (b) The adjacency matrix of the reordered author-similarity dataset using the reverse Cuthill-McKee algorithm.*

insights that could not be found by the standard solutions (see Figure 8).

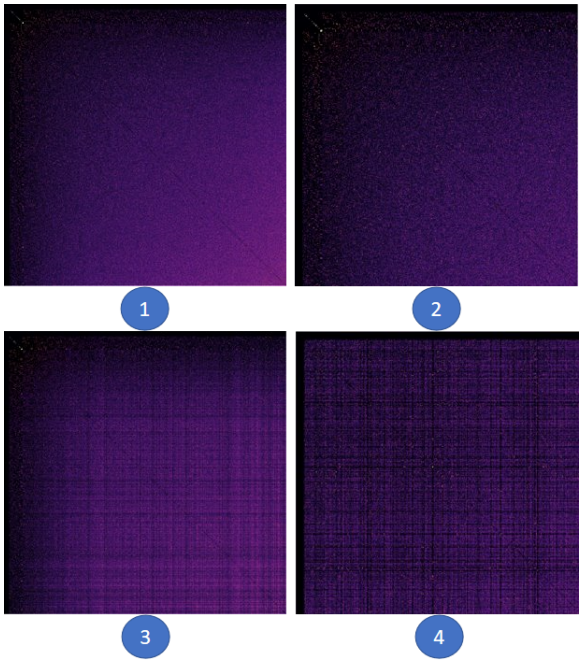


Figure 10: *Adjacency matrices of the author-similarity dataset reordered using the mean values ordering (1), weight sum ordering (2), unique value ordering (3), and shortest path ordering (4).*

The CPAN distributions dataset contains 2,724 nodes and 7,669 edges. The force-directed layout of the node-link diagram for this dataset can be seen on the right in Figure 3. When the graph is rendered, users can immediately notice how most of the nodes in the graph are connected in a large connected component. Within it, there are a few nodes that appear larger, as they have a higher number of incoming edges. Also, these nodes appear to be surrounded by groups of smaller nodes with fewer edges. These nodes represent packages, on which other packages depend. They are also used more frequently, compared to others. Nodes that are not part of the large connected component can be interpreted as outliers and therefore be ignored.

The randomly generated dataset contains 120 nodes, the maximum weight of an edge is 30 and the chance of an edge is 0.007 (0.7 %). The radial layout of the node-link diagram for this dataset can be seen in Figure 4 and the arc layout in Figure 5. In Figure 4 the initial radial layout on the left shows the whole dataset. In the filtered radial layout on the right users can see that most of the edges belong to the component of blue colored nodes if they select a node from the component of blue nodes. The arc layout can help the user finding co-occurrences in the data. In Figure 5 on the left the initial layout reveals the connected components in the graph. The filtered layout on the right reveals connections of a selected node within its component.

The author-similarity data includes 1,053 nodes and 907,164 edges. The initial adjacency matrix displaying this dataset is shown in Figure 9 (a). In the initial order there are no clear patterns that can be observed. However, when the reverse Cuthill-McKee ordering is executed on the data, users can see symmetric diagonal block patterns. The reordered matrix can be seen in Figure 9 (b).

This means that there are strongly connected components in the data and the nodes within them share similar characteristics. Users can also see that the components increase their size from the top left corner to the bottom right corner of the matrix, where the connected components are the biggest. Also, in each component, the edges are ordered by their weight from the bottom right corner of the component to the top left corner, where the weight of the edges is the biggest.

In Figure 10 the adjacency matrix displays the same dataset using the mean values ordering (1), weight sum ordering (2), unique value ordering (3), and shortest path ordering (4).

Users can see the reordered matrices in (1), (2), and (3) ordering the edges by weight from the top left corner to the bottom right corner of the matrix, where the weight of the edges is the largest. The reordered matrices (3) and mostly (4) form block patterns, which also confirm that there are strongly connected components in the data.

We can see that our tool can accomplish its task to find clusters, outliers, and patterns in the provided data, as well as combining them in a way to find strongly connected nodes in a graph dataset based on several layouts and reorderings.

6. Discussion and Limitations

We decided to use the Django framework to develop and maintain our website as it is scalable, versatile, allows for fast building of applications and working with a simple syntax. To visualize the datasets we have chosen the D3 library written in JavaScript. The key advantages it offers is the ability to generate visualizations using just HTML, CSS, JS, and SVG, but also with the canvas when in need of more performance. However, during the work with D3 we encountered some disadvantages such as the fact that it is not easy to learn quickly and if the datasets provided are large it would render SVG visualizations slowly. That is why we use SVG to visualize only certain parts of the data.

When users upload a csv file it is being used to visualize the graphs, run an algorithm to find clusters, apply the reorderings and convert the csv file into a json file. The performance of this process depends on the number of nodes and edges. This is because, for every node and edge, an object is made to achieve the conversion. However, the time required for this depends more on the nodes, unless there are relatively more edges than nodes (high density). Most algorithms do not take much time to execute during the conversion. The only exception is the shortest path ordering for large datasets.

The conversion performance was tested on a computer with an Intel i5-8600K CPU, 32 GB DDR4 RAM, and an NVIDIA GeForce RTX 2070 GPU (Figure 12 shows the test results.).

In Figure 11 and Figure 12 we can see how the application performs for datasets with a different number of nodes and edges. As the number of nodes and edges in the datasets increase, so does the conversion time. Using this information, a recommended limit should be given so that users can have a smooth experience with the tool. Datasets with a large number of nodes cause the node-link diagram and the interactions to render slowly. Users are recommended to use datasets with less than 5,500 nodes and not more than 700,000 edges. Datasets which are around this limit could still be rendered, but the conversion time will probably take more than a minute. If the dataset exceeds the limit then the application would probably crash. Also, datasets with file size larger than 100MB time-out the website. It should also be mentioned that browsers are not responsive when having to display many elements on a single page.

File name	.csv file size	.json file size	Node count	Edge count	Conversion time (milliseconds)
input2	552 B	3.06 KB	11	33	27
random30	2 KB	5.8 KB	30	30	31
random50	5.1 KB	10.4 KB	50	67	42
random100	20 KB	26 KB	100	257	64
word_adjacencies	25.1 KB	38.5 KB	112	510	73
random200	79.1 KB	82.5 KB	200	1213	131
jazz	78 KB	154 KB	198	2925	148
neural_network	174 KB	158 KB	297	2608	213
primary_school	111 KB	540 KB	236	11891	317
java	1 MB	193 KB	724	1746	566
random500	491 KB	406 KB	500	7652	746
co_authorship	2.14 MB	406 KB	1053	3504	903
random1000	1.91 MB	1.40 MB	1000	29811	1944
co_citation	2.14 MB	2.22 MB	1053	47022	2143
political_blogs	4.24 MB	1.11 MB	1490	20471	2255
online_social_network	6.89 MB	1.25 MB	1899	22160	3904
social_network	14.1 MB	838 KB	2724	7669	4171
random2500	11.9 MB	2.30 MB	2500	43862	7622
random2000	7.65 MB	5.45 MB	2000	119461	8722
random3000	17.1 MB	3.21 MB	3000	62918	11606
random3500	23.3 MB	4.28 MB	3500	85561	16272
random4000	30.5 MB	5.52 MB	4000	111862	22760
random4500	38.6 MB	6.90 MB	4500	141569	30444

Figure 11: *Table of the datasets, which were used in the tests, with their size, number of nodes/edges, and conversion time in milliseconds.*

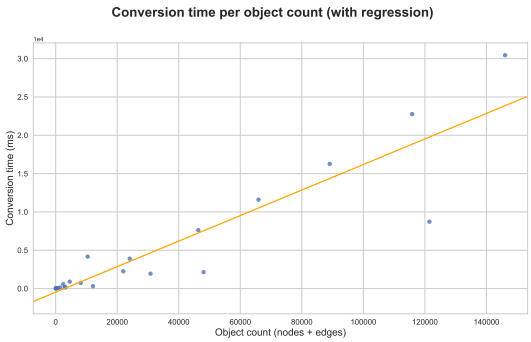


Figure 12: *Regression plot of the conversion time per number of nodes and edges for each of the datasets used in the test.*

7. Conclusion and Future Work

In its current state, our web-based tool has achieved its goal of rendering beautiful visualizations, which could be used to detect patterns and insights from the data, combining several layouts and reorderings. It provides useful layouts, interactions and reordering strategies that help the user to achieve this. However, there are ~~always~~ aspects that could be improved to make our application even better. Currently, the tool can only work with datasets in a specific format, which is not user-friendly. To improve this, our application has to support all possible datasets or at least the most common ones. Although our tool works on touchscreen devices, there are issues related to panning, zooming, and selecting elements. It would be useful to extend the repertoire of reordering strategies for the adjacency matrix by implementing other complex approaches.

References

[BB07] BAUR M., BRANDES U.: Multi-circular layout of micro/macro graphs. In *Proceedings of 15th International Symposium of Graph Drawing, GD* (2007), pp. 255–267. 5

[BBR*14] BURCH M., BECK F., RASCHKE M., BLASCHECK T., WEISKOPF D.: A dynamic graph visualization perspective on eye movement data. In *Proceedings of Symposium on Eye Tracking Research and Applications, ETRA* (2014), pp. 151–158. 1

[BBR*16] BEHRISCH M., BACH B., RICHE N. H., SCHRECK T., FEKETE J.: Matrix reordering methods for table and network visualization. *Computer Graphics Forum 35*, 3 (2016), 693–716. 2, 3

[BCG*17] BRUCKDORFER T., CORNELSEN S., GUTWENGER C., KAUFMANN M., MONTECCHIANI F., NÖLLENBURG M., WOLFF A.: Progress on partial edge drawings. *Journal of Graph Algorithms and Applications 21*, 4 (2017), 757–786. 3

[BMR*12] BURCH M., MÜLLER C., REINA G., SCHMAUDER H., GREIS M., WEISKOPF D.: Visualizing dynamic call graphs. In *Proceedings of the Vision, Modeling, and Visualization Workshop* (2012), pp. 207–214. 1

[BSP20] BEHRISCH M., SCHRECK T., PFISTER H.: GUIRO: user-guided matrix reordering. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (2020), 184–194. 2

[Bur15] BURCH M.: Dynamic graph visualization with multiple visual metaphors. In *Proceedings of the 8th International Symposium on Visual Information Communication and Interaction, VINCI* (2015), pp. 27–34. 2

[Bur17] BURCH M.: The dynamic graph wall: visualizing evolving graphs with multiple visual metaphors. *Journal of Visualization 20*, 3 (2017), 461–469. 2

[BVKW11] BURCH M., VEHLOW C., KONEVTSOVA N., WEISKOPF D.: Evaluating partially drawn links for directed graph edges. In *In Proceedings of the 19th International Symposium on Graph Drawing, GD* (2011), pp. 226–237. 3

[cpa19] Datasets, 2019. URL: https://github.com/gephi/gephi/wiki/Datasets. 1

[DETT99] DI BATTISTA G., EADES P., TAMASSIA R., TOLLIS I. G.: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999. 1, 2

[EDG*08] ELMQVIST N., DO T., GOODELL H., HENRY N., FEKETE J.: ZAME: interactive large-scale graph visualization. In *Proceedings of IEEE VGTC Pacific Visualization Symposium* (2008), pp. 215–222. 1, 2

[EK18] EADES P., KLEIN K.: Graph visualization. In *Graph Data Management, Fundamental Issues and Recent Developments*. 2018, pp. 33–70. 2

[Eul41] EULER L.: Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Petropolitanae 8* (1741), 128–140. 2

[FR91] FRUCHTERMAN T. M. J., REINGOLD E. M.: Graph drawing by force-directed placement. *Software, Practice, and Experience 21*, 11 (1991), 1129–1164. 5

[GBD09] GREILICH M., BURCH M., DIEHL S.: Visualizing the evolution of compound digraphs with TimeArcTrees. *Computer Graphics Forum 28*, 3 (2009), 975–982. 1, 5

[GFC05] GHONIEM M., FEKETE J., CASTAGLIOLA P.: On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Information Visualization 4*, 2 (2005), 114–135. 1, 2

[HF06] HENRY N., FEKETE J.: MatrixExplorer: a dual-representation system to explore social networks. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 677–684. 1, 2

[HF07] HENRY N., FEKETE J.: Matlink: Enhanced matrix visualization for analyzing social networks. In *Proceedings of International Conference on Human-Computer Interaction INTERACT* (2007), pp. 288–302. 1, 2

[HFM07] HENRY N., FEKETE J., MCGUFFIN M. J.: Nodetrix: a hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1302–1309. 1, 2

[HIvWF11] HOLTEN D., ISENBERG P., VAN WIJK J. J., FEKETE J.: An extended evaluation of the readability of tapered, animated, and textured directed-edge representations in node-link graphs. In *Proceedings of the IEEE Pacific Visualization Symposium, PacificVis* (2011), pp. 195–202. 3

[KAF*08] KEIM D. A., ANDRIENKO G. L., FEKETE J., GÖRG C., KOHLHAMMER J., MELANÇON G.: Visual analytics: Definition, process, and challenges. In *Information Visualization - Human-Centered Issues and Perspectives*. 2008, pp. 154–175. 2

[KEC06] KELLER R., ECKERT C. M., CLARKSON P. J.: Matrices or node-link diagrams: which visual representation is better for visualising connectivity models? *Information Visualization 5*, 1 (2006), 62–76. 2

[KW01] KAUFMANN M., WAGNER D. (Eds.):. *Drawing Graphs, Methods and Models (the book grow out of a Dagstuhl Seminar, April 1999)* (2001), vol. 2025 of *Lecture Notes in Computer Science*, Springer. 2

[OJK19] OKOE M., JIANU R., KOBOUROV S. G.: Node-link or adjacency matrices: Old question, new insights. *IEEE Transactions on Visualization and Computer Graphics 25*, 10 (2019), 2940–2952. 1

[Pur97] PURCHASE H. C.: Which aesthetic has the greatest effect on human understanding? In *Proceedings of the International Symposium on Graph Drawing, GD* (1997), pp. 248–261. 2

[RLMJ05] ROSENHOLTZ R., LI Y., MANSFIELD J., JIN Z.: Feature congestion: a measure of display clutter. In *Proceedings of the Conference on Human Factors in Computing Systems, CHI* (2005), pp. 761–770. 2

[SLH*18] SONI U., LU Y., HANSEN B., PURCHASE H. C., KOBOUROV S. G., MACIEJEWSKI R.: The perception of graph properties in graph layouts. *Computer Graphics Forum 37*, 3 (2018), 169–181. 2

[vLKS*11] VON LANDESBERGER T., KUIJPER A., SCHRECK T., KOHLHAMMER J., VAN WIJK J. J., FEKETE J., FELLNER D. W.: Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum 30*, 6 (2011), 1719–1749. 2

[War04] WARE C.: *Information Visualization: Perception for Design*. Morgan Kaufmann, 2004. 2, 4

[War08] WARE C.: *Visual Thinking: for Design*. Morgan Kaufmann Series in Interactive Technologies, Paperback, 2008. 2

[WPCM02] WARE C., PURCHASE H. C., COLPOYS L., MCGILL M.: Cognitive measurements of graph aesthetics. *Information Visualization 1*, 2 (2002), 103–110. 3

[YaKSJ07] YI J. S., AH KANG Y., STASKO J. T., JACKO J. A.: Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1224–1231. 6