EuroVis Workshop on Visual Analytics (2022)
M. Angelini and J. Bernard (Editors)

Towards Understanding Edit Histories of Multivariate Graphs

P. Berger and H. Schumann and C. Tominski

Institute for Visual & Analytic Computing, University of Rostock, Germany

Abstract

The visual analysis of multivariate graphs increasingly involves not only exploring the data, but also editing them. Existing editing approaches for multivariate graphs support visual analytics workflows by facilitating a seamless switch between data exploration and editing. However, it remains difficult to comprehend performed editing operations in retrospect and to compare different editing results. Addressing these challenges, we propose a model describing what graph aspects can be edited and how. Based on this model, we develop a novel approach to visually track and understand data changes due to edit operations. To visualize the different graph states resulting from edits, we extend an existing graph visualization approach so that graph structure and the associated multivariate attributes can be represented together. Branching sequences of edits are visualized as a node-link tree layout where nodes represent graph states and edges visually encode the performed edit operations and the graph aspects they affect. Individual editing operations can be inspected by dynamically expanding edges to detail views on demand. In addition, we support the comparison of graph states through an interactive creation of attribute filters that can be applied to other states to highlight similarities.

CCS Concepts

• ***Human-centered computing*** → ***Visual analytics;***

1. Introduction

In addition to analysis-oriented tasks, data editing and data wrangling has become increasingly important in visual analytics scenarios [Bau06, KHP*11]. Editing the data can be necessary to prepare the data for their analysis, to update information, to correct errors, or to experiment with different *what-if* scenarios. Baudel was among the first to propose the integration of data manipulation facilities directly into visual representations of data [Bau06]. Based on what Baudel calls the direct manipulation principle, several previous works have proposed dedicated solutions to edit graphs and data attributes directly in the visualization [GSE*14, EGST16, HBS*21]. These approaches support visual analytics workflows where users can switch seamlessly between data exploration and data editing. Users can carry out edits directly in the visualization and immediately see how they take effect locally (e.g., deletion of an edge or change of an attribute value) and impact the overall graph globally (e.g., changes of graph properties and value distributions). This is not only useful for data correction, but also for hypothesis testing of what-if scenarios (e.g., planning new bus routes and stops [WZD*21] or developing soccer teams [BST22]).

In general, data editing and what-if testing are iterative processes where users may try out several alternative edits before arriving at a final result. Therefore, comprehending editing steps in retrospect and comparing multiple alternative outcomes are important tasks to be supported by visual analytics tools. In addition to understanding *what* has changed during an editing phase, it can also be of interest to understand *how* the edits were performed (e.g., via individual alphanumeric inputs or via a continuous direct manipulation gesture). However, so far, the iterative character of editing in terms of what has changed during the process and how it was changed is mostly ignored by current tools.

We follow the argumentation of Ragan et al. [RESC15], who state that provenance information and its visualization can enhance visual data analysis workflows and help users comprehend them in retrospect. In our case, we focus on the purpose of understanding edit histories. We build on previous research in the context of dynamic graph visualization [BBDW17] and interaction history visualization [XOW*20]. Dynamic graph visualization has proven to be useful for understanding *what* has changed in a graph over time, or over a sequence of edit operations. However, dynamic graph visualizations usually do not consider *how* the data was changed. This is where the visualization of interaction histories comes into play, which depicts how performed user actions led to data changes. We propose a combined visualization of aspects of dynamic graphs and interaction histories to visually track changes caused by edit operations on multivariate graphs. Our contributions are (1) a model for describing edit operations on multivariate graphs, (2) a novel visualization for representing edit histories on multivariate graphs, and (3) interaction facilities to support detail inspection and comparison of edit operations and affected data.

2. Related Work

Our work is related to the visualization of multivariate graphs and their editing via direct manipulation as well as provenance visualization, which touches aspects of visualizing dynamic graphs and interaction histories.

Visualizing Multivariate Graphs The visualization of multivariate graphs is comprehensively discussed in the literature [KPW14, NMSL19]. Existing approaches are typically based on node-link diagrams or matrix representations. These base visualizations are usually extended in order to represent multivariate data, for example via incorporating additional views [KLS*17, NSL19], embedding additional encodings [MB19, BST19], or laying out the graph based on its attributes [SA06, WT08].

Editing Multivariate Graphs Following Baudel's direct manipulation principle [Bau06], previous work has combined exploration and editing for fluent visual analytics workflows. The existing approaches differ in what data edits they support and how the edit operations are performed. Eichner et al. [EGST16] propose editing node attributes in attribute-driven node-link representations by continuous drag gestures. In contrast, the EditLens [GSE*14] facilitates the insertion and deletion of nodes and edges through discrete tap and flick interactions. Editing edges in matrix representations is similarly performed through tap and drag interactions on matrix cells [GSLT15]. In so-called responsive matrix cells [HBS*21] editing facilities are revealed dynamically as matrix cells respond to size changes via focus+context. The graph structure and data attributes can then be edited either through alphanumeric input or by dragging data elements in the visualization directly.

While the listed approaches support on-the-fly editing of multivariate graphs, they do not consider provenance information about what data has changed and how during the process.

Visualizing Provenance Information In general, the visualization of provenance information supports the recalling of actions, the replication of workflows, and the presentation of results [RESC15]. Here, we are concerned with edit operations on multivariate graphs, a topic that is also related to dynamically changing graph models, for example, in biology [GSE*14, SGT*18].

Changes in dynamic graphs can be visualized in various ways [BBDW17], where two fundamental distinctions can be made. The dimension of time is either mapped to time, resulting in animations, or time is mapped to space, creating a timeline. Animation-based approaches exist for node-link diagrams [BPF14, FT04] and for matrices [RM14]. They provide an intuitive visualization of what graph aspects change over time. However, since an animation shows only a single snapshot of the graph at a time, the mental effort required to integrate all observed snapshots to a coherent overall mental model can be high.

Timeline-based approaches, on the other hand, show the entire changes of the graph in a single image. There are approaches using node-link diagrams [BVB*11], matrices [BCD*10], and adjacency list [HBW14]. While changes of graph characteristics can be identified and traced along the timeline, scalability is a problem, as only limited display space is available per graph representations.

Understanding *how* the data was edited is related to previous work on visualizing interaction histories, which may include different system states, sequences of actions, or a combination of both [HMSA08, XOW*20]. System states are often displayed as snapshots with annotations of the performed interactions [FCM18, HMSA08, BCD*10]. The visualization of branching action sequences is usually done via node-link diagrams [CGL20, KNS04]. Visualizing states and actions together is useful for externalizing the iterative processes of editing data [JE13, DHRL*12].

In summary, dynamic graph visualization helps us to understand *what* changed in a graph, whereas interaction history visualization allows us to see *how* a graph was changed. In this work, we aim to bring *what* and *how* together. Previous graph analysis approaches [ZK15, ZK17] propose a combination of animated changes and action sequences to visualize a linear edit history. As we assume editing phases with a moderate number of operations and branching histories, we utilize a time-to-space mapping and an integrated encoding of edit operations to visualize *what* has changed and *how* it was changed.

3. Approach Overview

In this section, we briefly discuss the requirements and present the basic idea of our approach for visualizing edit operations on multivariate graphs.

Requirements Based on the problem description and the review of related work, we identified the following requirements:

R1: Visualize what has changed. We need to display the different states of a graph that are created due to edit operation in order to understand what has changed (i.e., structure or attributes) and what the consequences are (e.g., creation of new substructures or changes in the attribute distribution).

R2: Visualize how it was changed. To understand how the graph was changed, the performed edit operations should be visualized to spot patterns (e.g., repetition of edits on the same parts of the graph) and to identify details on how edits were performed (e.g., alphanumeric input or continuous direct manipulation gestures).

Basic Approach Addressing these requirements, we design our approach as follows. Visualizing a multivariate graph is already challenging. Visualizing multiple states of an edited multivariate graph in full detail is therefore hardly possible. Instead, we have to visualize the different graph states in an abstract and compact manner. Here, we utilize and extend Graph thumbnails [YDK*18], which allows us to visualize the graph states as compact thumbnails in a tree layout. While the thumbnails address **R1**, the tree layout is geared toward fulfilling **R2**. In the tree layout, the edges represent the edit operations that lead from one graph state to another. The edges are visually enriched with stacked color-coded bars for showing the edit operations and the affected graph aspects. Figure 1 gives an overview of our approach.

Next, we introduce a model for describing what graph aspects can be edited and explain our approach for visualizing graph edit histories based on that model in detail.

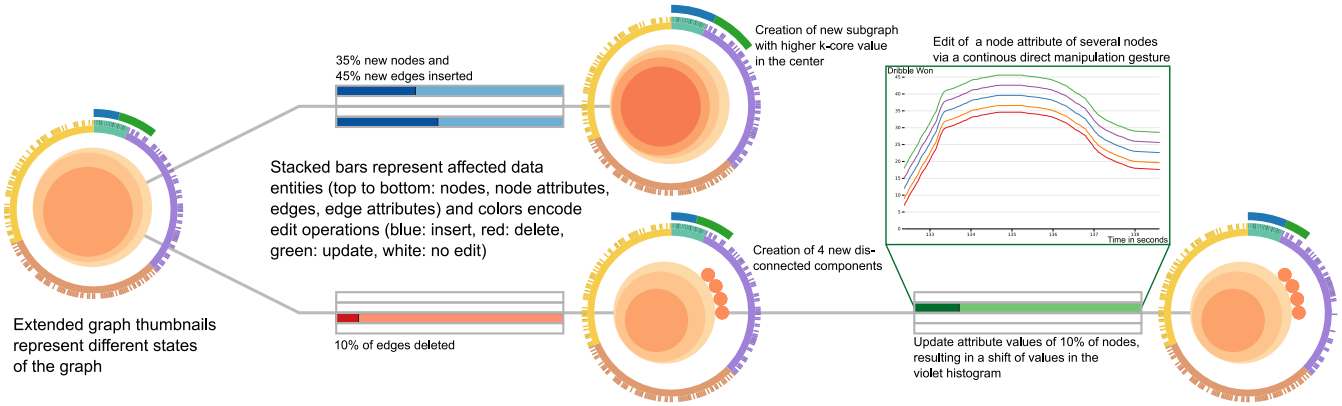

Figure 1: *Visualization of the editing history of a multivariate graph of soccer players, edited in a matrix representation [HBS*21]. Nodes represent the aggregated graph structure and attribute distributions. Edges display the edit operation and affected graph aspects (stacked color-coded bars). On demand, integrated visualizations are provided for details on the editing operation.*

4. Modeling Edit Operations on Multivariate Graphs

In multivariate graphs, edit operations can affect different data entities, including nodes, edges, node attributes, and edge attributes. On these entities, three basic operations can be performed: insert, delete and update [GSE*14]. In our context of direct visual editing, the edit operations are not just singular events, such as an alphanumeric input, but can consist of a sequence of multiple value changes. Taking this into account, we model an edit history of a multivariate graph $\mathcal{G}$ as another multivariate graph $\mathcal{E}_\mathcal{G} = (S, T, A)$ where the nodes S define states, the edges $T = S \times S$ define state transitions, and the edge attributes A capture detailed information about edit operations.

- States: A node $s \in S$ represents a state that the graph $\mathcal{G}$ assumed at one point in time. In other words, s represents a snapshot of the multivariate graph being analyzed and edited.
- Transitions: An edge $t \in T$ models a transition from one state to another due to an edit operation. That is, a transition $t = (s, s')$ exists if s' was created by performing edit operations on s.
- Edit operations: Transitions store information about the performed edit. An edit is modeled as a tuple $e = (o, d, \mathbf{v})$ where o is an operation $o \in \{\text{insert, delete, update}\}$, d represents the manipulated data entities, which may be any subset of the union[†] of the states involved in the edit operation $d \subseteq s \cup s'$, and $\mathbf{v}$ is a sequence of time-value pairs. For discrete edit operations (e.g., via alpha-numeric value input), $\mathbf{v}$ consists of only a single time-value pair $\mathbf{v} = (t, v)$. For continuous edit operations (e.g., via a slider or a direct manipulation gesture), $\mathbf{v}$ consists of multiple time-value pairs capturing all intermediate value changes $\mathbf{v} = (t_1, v_1), \ldots, (t_n, v_n)$.

Next we describe how an edit history $\mathcal{E}_\mathcal{G}$ of a multivariate graph $\mathcal{G}$ can be visualized for understanding editing provenance.

[†] The union is necessary to cope with the insertion of new information not yet existing in state s and also the deletion of information which is no longer present in state s'.

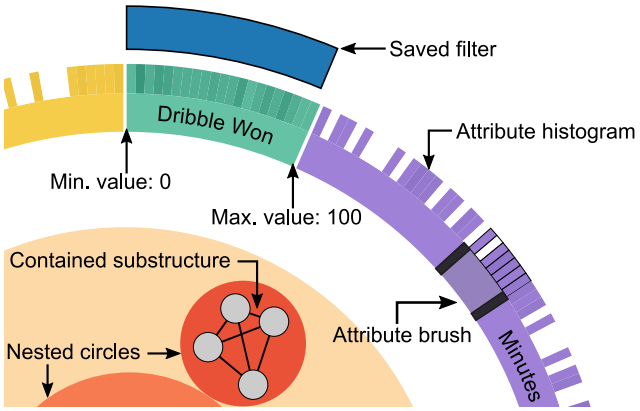

Figure 2: *Illustration of our extension of Graph thumbnails.*

5. Visualizing Edit Operations on Multivariate Graphs

As outlined before, graph states are visualized as compact thumbnails in a tree layout whose edges encode information about the edit operations. Aspects are explained in detail next.

Representing What has been Edited To understand what has changed during editing, several states $s \in S$ of the multivariate graph $\mathcal{G}$ have to be represented, where each s has its own topological graph structure and multivariate attributes. However, since not all aspects of the graph can be visualized at once, a compact representation is needed. A suitable approach for a graph state's topological structure are Graph thumbnails [YDK*18], which visually preserve the key structural aspects of graphs (e.g., number and properties of substructures). The compact thumbnail representation outperforms treemaps and icicle plots in conveying hierarchies and is comparable to node link and matrix representations in regards to overview tasks on the structure [YDK*18].

For Graph thumbnails, a graph is decomposed hierarchically based on its connectivity using k-cores, which are subgraphs with

minimum degree k. This decomposition always guarantees identical hierarchies for isomorphic graphs and takes linear time depending on the number of edges. The resulting hierarchy of subgraphs is then represented as nested circles whose area corresponds to the subgraphs' size as illustrated in Figure 2.

However, Graph thumbnails do not represent multivariate node and edge attributes. Therefore, we extend Graph thumbnails by adding histograms for individual attributes of the graph as ring segments as shown in Figure 2. Within these ring segments, the distribution of attribute values is visualized as color-coded bar segments. A darker bar indicates that a value occurs more often and a brighter bar indicates that a value occurs less often. White segments, on the other hand, mean that no value lies in this range. In addition, the differences between the minimum and maximum of the value ranges are depicted in the size of the individual ring segments.

Relations between the graph structure (thumbnails core) and the multivariate attributes (thumbnail ring) can be revealed through interactive highlighting. Hovering nested circles highlights the corresponding value ranges of nodes and edges in the attribute histograms and vice versa.

In addition, we enable the on-demand display of individual structure elements within the nested circles during highlighting as node-link diagram. This makes it possible to associate individual nodes or edges with their multivariate attribute values (see Figure 2).

Together, the nested circles and the ring segments form a compact visual representation where structural characteristics and attribute properties are visible in a single coherent view. Key aspects of a graph state's structure and its attributes can still be detected, for example, the number and composition of substructures as well as attribute distributions and outlier values. This allows users to compare multiple graph states for similarities and dissimilarities on an overview level and the interactive highlighting provides additional details on demand to better understand data changes due to edit operations.

To further support the comparison of graph states, findings made for one state should ideally be transferable to other states. To this end, users can create attribute filters, which are specified by selecting individual values or by brushing entire value ranges in the histogram rings. Once defined, filters appear as extra circle segments above the histograms in all graph thumbnails as shown in Figure 2 (top). The sizes of these extra segments visually encode the number of edited graph entities in a graph state matching the filter specification. As stated above, graph entities corresponding to the selected values can be displayed on demand within the nested circles. In this way, found properties in one graph state can be quickly compared with all other states.

With the extended Graph thumbnails described so far, we fulfill **R1**, that is, we can visually explore *what* has changed.

Representing How Edits were Performed In order to comprehend *how* edits have been performed, the specific operation o, the affected data entities d and the value changes $\mathbf{v}$ must be shown.

To this end, we incorporate stacked bars into the edge representation of the tree layout of the overall edit history, as illustrated in Figure. 1. For each transitions $t \in T$ between two states, we show four horizontal bar charts to represent the data entities d. Each bar chart corresponds to a type of data entity: nodes, node attributes, edges, and edge attributes (from top to bottom). Edit operations o are encoded by color: blue for inserts, red for deletes, and green for updates. The lengths of the bars indicate the relative number of edited nodes and edges. A white bar chart indicates that no edits affected a particular type of data entities.

To visualize the actual value changes over time $\mathbf{v}$, we again use on-demand views. Each bar chart can be expanded dynamically to a line plot to reveal details of the underlying data changes, where different edit operations are shown as separate lines.

With the help of these views, users can understand *how* the data edits were performed as demanded by **R2**. For example, abrupt value changes indicate discrete alphanumeric inputs, whereas smoother value changes suggest that edits were made using continuous direct manipulations gestures. Figure 1 shows an on-demand view in which attribute values for multiple nodes are changed continuously (e.g., moving nodes in a scatterplot [EGST16]).

6. Discussion

Our approach is meant to complement existing multivariate graph visualizations with functionality to communicate and make comprehensible direct editing operations. The history shown in Figure 1 is an extract of a what-if analysis of a multivariate graph of soccer players, which is edited in a matrix representation [HBS*21]. Our edit history is used to comprehend the editing process and to compare different graph states. The stacked bars and extended thumbnails enable us to spot patterns in the editing process and to get feedback on the edit effects, e.g., changes in the attribute distribution or substructures. Detailed information on individual edits can be viewed on-demand. However, the aggregation of information creates a dense representation in which it is difficult to identify individual graph elements. In this regard, defining filters proved to be helpful in finding and comparing substructures in different graph states. For a detailed exploration of a graph state (incl. structure and attributes) it is preferable to utilize a dedicated multivariate graph visualization.

7. Conclusion

In this work, we presented a novel approach for the visualization of edit histories of multivariate graphs. We introduced a model describing edit operations on graphs, a novel compact visual representation of edit histories based on extended Graph thumbnails, and corresponding interaction facilities to visually track and compare data edits across several graph states. Our solution is meant to complement existing multivariate graph visualizations with functionality to communicate and make comprehensible direct editing operations that may have occurred during visual analytics workflows involving data corrections or what-if testing.

In the future, we plan to further improve the utility of our techniques. For example, many individual edit operations lead to a large number of graph states in the tree layout, which makes exploring and comparing difficult. We plan to support these tasks by aggregating non important edit operations and graph states. A recent interaction ranking model [FCM18] provides a good starting point.

References

[Bau06] BAUDEL T.: From Information Visualization to Direct Manipulation: Extending a Generic Visualization Framework for the Interactive Editing of Large Datasets. In *Proc. ACM UIST* (2006), ACM. doi:10.1145/1166253.1166265. 1, 2

[BBDW17] BECK F., BURCH M., DIEHL S., WEISKOPF D.: A Taxonomy and Survey of Dynamic Graph Visualization. *Computer Graphics Forum 36*, 1 (2017). doi:10.1111/cgf.12791. 1, 2

[BCD*10] BEZERIANOS A., CHEVALIER F., DRAGICEVIC P., ELMQVIST N., FEKETE J.: GraphDice: A System for Exploring Multivariate Social Networks. *Computer Graphics Forum 29*, 3 (2010). doi:10.1111/j.1467-8659.2009.01687.x. 2

[BPF14] BACH B., PIETRIGA E., FEKETE J.-D.: GraphDiaries: Animated Transitions andTemporal Navigation for Dynamic Networks. *IEEE TVCG 20*, 5 (2014). doi:10.1109/TVCG.2013.254. 2

[BST19] BERGER P., SCHUMANN H., TOMINSKI C.: Visually Exploring Relations between Structure and Attributes in Multivariate Graphs. In *Proc. IEEE International Conference on Information Visualization* (2019), IEEE. doi:10.1109/IV.2019.00051. 2

[BST22] BERGER P., SCHUMANN H., TOMINSKI C.: Integrating Visual Exploration and Direct Editing of Multivariate Graphs. In *Integrating AI and Visualisation for Visual Knowledge Discovery*. Springer International Publishing, 2022. (To appear). doi:10.1007/978-3-030-93119-3_18. 1

[BVB*11] BURCH M., VEHLOW C., BECK F., DIEHL S., WEISKOPF D.: Parallel Edge Splatting for Scalable Dynamic Graph Visualization. *IEEE TVCG 17*, 12 (2011). doi:10.1109/TVCG.2011.226. 2

[CGL20] CUTLER Z., GADHAVE K., LEX A.: Trrack: A library for provenance-tracking in web-based visualizations. In *IEEE VIS - Short Papers* (2020), IEEE. doi:10.1109/VIS47514.2020.00030. 2

[DHRL*12] DUNNE C., HENRY RICHE N., LEE B., METOYER R., ROBERTSON G.: GraphTrail: Analyzing Large Multivariate, Heterogeneous Networks while Supporting Exploration History. In *Proc. ACM CHI* (2012). doi:10.1145/2207676.2208293. 2

[EGST16] EICHNER C., GLADISCH S., SCHUMANN H., TOMINSKI C.: Direct Visual Editing of Node Attributes in Graphs. *Informatics 3*, 4 (2016). doi:10.3390/informatics3040017. 1, 2, 4

[FCM18] FUJIWARA T., CRNOVRSANIN T., MA K.-L.: Concise provenance of interactive network analysis. *Visual Informatics 2*, 4 (2018). doi:10.1016/j.visinf.2018.12.002. 2, 4

[FT04] FRISHMAN Y., TAL A.: Dynamic drawing of clustered graphs. In *IEEE InfoVIS* (2004). doi:10.1109/INFVIS.2004.18. 2

[GSE*14] GLADISCH S., SCHUMANN H., ERNST M., FÜLLEN G., TOMINSKI C.: Semi-Automatic Editing of Graphs with Customized Layouts. *Computer Graphics Forum 33*, 3 (2014). doi:10.1111/cgf.12394. 1, 2, 3

[GSLT15] GLADISCH S., SCHUMANN H., LUBOSCHIK M., TOMINSKI C.: Toward using Matrix Visualizations for Graph Editing. In *Poster at IEEE Conference on Information Visualization* (2015). 2

[HBS*21] HORAK T., BERGER P., SCHUMANN H., DACHSELT R., TOMINSKI C.: Responsive Matrix Cells: A Focus+Context Approach for Exploring and Editing Multivariate Graphs. *IEEE TVCG 27*, 2 (2021). doi:10.1109/TVCG.2020.3030371. 1, 2, 3, 4

[HBW14] HLAWATSCH M., BURCH M., WEISKOPF D.: Visual Adjacency Lists for Dynamic Graphs. *IEEE TVCG 20*, 11 (2014). doi:10.1109/TVCG.2014.2322594. 2

[HMSA08] HEER J., MACKINLAY J., STOLTE C., AGRAWALA M.: Graphical Histories for Visualization: Supporting Analysis, Communication, and Evaluation. *IEEE TVCG 14*, 6 (2008). doi:10.1109/TVCG.2008.137. 2

[JE13] JAVED W., ELMQVIST N.: ExPlates: Spatializing Interactive Analysis to Scaffold Visual Exploration. *Computer Graphics Forum 32*, 3pt4 (2013). doi:https://doi.org/10.1111/cgf.12131. 2

[KHP*11] KANDEL S., HEER J., PLAISANT C., KENNEDY J., VAN HAM F., RICHE N. H., WEAVER C., LEE B., BRODBECK D., BUONO P.: Research Directions in Data Wrangling: Visualizations and Transformations for Usable and Credible Data. *Information Visualization 10*, 4 (2011). doi:10.1177/1473871611415994. 1

[KLS*17] KERZNER E., LEX A., SIGULINSKY C., URNESS T., JONES B., MARC R., MEYER M.: Graffinity: Visualizing Connectivity in Large Graphs. *Computer Graphics Forum 36*, 3 (2017). doi:10.1111/cgf.13184. 2

[KNS04] KREUSELER M., NOCKE T., SCHUMANN H.: A History Mechanism for Visual Data Mining. In *IEEE InfoVIS* (2004). doi:10.1109/INFVIS.2004.2. 2

[KPW14] KERREN A., PURCHASE H. C., WARD M. O. (Eds.):. *Multivariate Network Visualization* (2014), vol. 8380 of *Lecture Notes in Computer Science*, Springer. doi:10.1007/978-3-319-06793-3. 2

[MB19] MAJOR T., BASOLE R. C.: Graphicle: Exploring Units, Networks, and Context in a Blended Visualization Approach. *IEEE TVCG 25*, 1 (2019). doi:10.1109/tvcg.2018.2865151. 2

[NMSL19] NOBRE C., MEYER M., STREIT M., LEX A.: The State of the Art in Visualizing Multivariate Networks. *Computer Graphics Forum 38*, 3 (2019). doi:10.1111/cgf.13728. 2

[NSL19] NOBRE C., STREIT M., LEX A.: Juniper: A Tree+Table Approach to Multivariate Graph Visualization. *IEEE TVCG 25*, 1 (2019). doi:10.1109/tvcg.2018.2865149. 2

[RESC15] RAGAN E. D., ENDERT A., SANYAL J., CHEN J.: Characterizing Provenance in Visualization and Data Analysis: An Organizational Framework of Provenance Types and Purposes. *IEEE TVCG 22*, 1 (2015). doi:10.1109/TVCG.2015.2467551. 1, 2

[RM14] RUFIANGE S., MELANÇON G.: AniMatrix: A Matrix-Based Visualization of Software Evolution. In *IEEE Working Conf. on Software Visualization* (2014). doi:10.1109/VISSOFT.2014.30. 2

[SA06] SHNEIDERMAN B., ARIS A.: Network Visualization by Semantic Substrates. *IEEE TVCG 12*, 5 (2006). doi:10.1109/TVCG.2006.166. 2

[SGT*18] SCHARM M., GEBHARDT T., TOURÉ V., BAGNACANI A., SALEHZADEH-YAZDI A., WOLKENHAUER O., WALTEMATH D.: Evolution of computational models in biomodels database and the physiome model repository. *BMC Systems Biology 12*, 1 (2018). doi:10.1186/s12918-018-0553-2. 2

[WT08] WU Y., TAKATSUKA M.: Visualizing Multivariate Networks: A Hybrid Approach. In *Proc. IEEE Pacific VIS* (2008), IEEE. doi:10.1109/pacificvis.2008.4475480. 2

[WZD*21] WENG D., ZHENG C., DENG Z., MA M., BAO J., ZHENG Y., XU M., WU Y.: Towards Better Bus Networks: A Visual Analytics Approach. *IEEE TVCG 27*, 2 (2021). doi:10.1109/TVCG.2020.3030458. 1

[XOW*20] XU K., OTTLEY A., WALCHSHOFER C., STREIT M., CHANG R., WENSKOVITCH J.: Survey on the Analysis of User Interactions and Visualization Provenance. *Computer Graphics Forum 39*, 3 (2020). doi:10.1111/cgf.14035. 1, 2

[YDK*18] YOGHOURDJIAN V., DWYER T., KLEIN K., MARRIOTT K., WYBROW M.: Graph Thumbnails : Identifying and Comparing Multiple Graphs at a Glance. *IEEE TVCG 24*, 12 (2018). doi:10.1109/TVCG.2018.2790961. 2, 3

[ZK15] ZIMMER B., KERREN A.: Displaying User Behavior in the Collaborative Graph Visualization System OnGraX. In *Graph Drawing and Network Visualization* (2015), Di Giacomo E., Lubiw A., (Eds.), Springer International Publishing. 2

[ZK17] ZIMMER B., KERREN A.: OnGraX: A Web-Based System for the Collaborative Visual Analysis of Graphs. *Journal of Graph Algorithms and Applications 21*, 1 (2017). doi:10.7155/jgaa.00399. 2