Graph Exploration by Multiple Linked Metric Views

A. Panagiotidis[1], M. Burch[1], O. Deussen[2], D. Weiskopf[1], T. Ertl[1]
[1]*Visualization Research Center, University of Stuttgart, Germany*
{*panagiotidis,burch,weiskopf,ertl*}@*visus.uni-stuttgart.de*
[2]*Dept. of Computer and Information Science, University of Konstanz, Germany*
oliver.deussen@uni-konstanz.de

***Abstract*—The visualization of relational data by node-link diagrams quickly leads to a degradation of performance at some exploration tasks when the diagrams show visual clutter and overdraw. To address this challenge of large-data graph visualization, we introduce Graph Metric Views, a technique that enriches the visualization of traditional layout strategies for node-link diagrams by additionally allowing an analyst to interactively explore graph-specific metrics such as number of nodes, number of link crossings, link coverage, or degree of orthogonality. To this end, we support an analyst with additional histogram-like representations at the axes of the display space for graph-specific metrics. In this way, a cluttered and densely packed node-link diagram becomes more explorable even for dense graph regions: The user can use the distribution of metric values as an overview and then select regions of interest for further investigation and filtering.**

***Keywords*-Graphs, node-link diagrams, metrics.**

I. Introduction

Graphs allow modeling and illustrating relationships among a group of objects. Euler [1] was one of the pioneers who modeled such relationships in a graphical way by using nodes for the objects and links expressing the pairwise relations when he found an abstraction for the problem of the “Seven Bridges of Königsberg”. Nowadays, even sophisticated methods for graph drawing produce cluttered visualizations for large and dense networks. This is even more problematic considering that there are further attributes in the data that cannot be integrated into a node-link diagram.

In this paper, we introduce *Graph Metric Views*: Histograms attached to the sides of a node-link diagram (see Figure 1), similar to marginal histograms aligned with 2D scatter plots. These views depict information about the layout of the graph, its graph-theoretic properties, and the underlying data. While users are exploring the graph and interacting with it, the metric views are updated accordingly. Furthermore, interesting regions of the metric views can be visually linked to the node-link diagram by coordination strategies like brushing and linking or direct visual linking.

We argue that this approach is advantageous when the graph becomes cluttered [2] or all visual channels are exhausted in the node-link diagram and thus no further information can be shown. Furthermore, an analyst can explore regions that are not clearly visible in the layout due to occlusion and overdraw. For example, cliques may not be perceived due to the high link density in these regions, whereas we can clearly show where and how big they are.

Our interactive technique is designed around the Visual Information Seeking Mantra [3]: overview first, zoom and filter, then details-on-demand. We employ multiple coordinated views [4], [5] and use brushing and linking [6]. To preserve the user’s mental map we introduce the concept of a graph minimap to provide context information. This minimap always shows the whole graph in a smaller view while a user focuses on parts of it in the node-link diagram. Our approach is extensible since we rely on common techniques for the graph layout and its rendering, while further metrics can be added to support new data sets.

II. Related Work

Graph drawing is a well-established field, with many ways to visualize relational data and networks directly [7], [8] or derived from multivariate, temporal data [9]. Typically, a node-link diagram is used that is laid out in a force-directed approach [10] or by stress majorization [11], often motivated by cognitive and aesthetic reasons [12], [13].

Despite sophisticated algorithms, graph layouts still result in local and global hairball-like structures for complex and large input data. This visual clutter is undesirable because it leads to degradation of performance at some exploration and analysis tasks [2], for example due to many link crossings. In general, there are many ways to deal with and reduce visual clutter [14].

Regarding node-link diagrams specifically, there are several ways to improve the presentation to reduce visual clutter and to reveal interesting patterns. Edge bundling is a prominent method to combine links according to their hierarchy [15] or in a force-directed manner [16]. Similarly, Cui et al. [17] bundle links by routing them through points of a virtual mesh that is modified interactively or automatically. Several splatting approaches deviate from the typical link drawing and instead use densities to depict the relations in the data [18], [19], [20]. Level-of-detail approaches combine link cumulation with density-based node aggregation [21]. Besides the discussion and analysis in the respective publications, there are many evaluations on the topic of modifying links for graph visualization [22], [23].

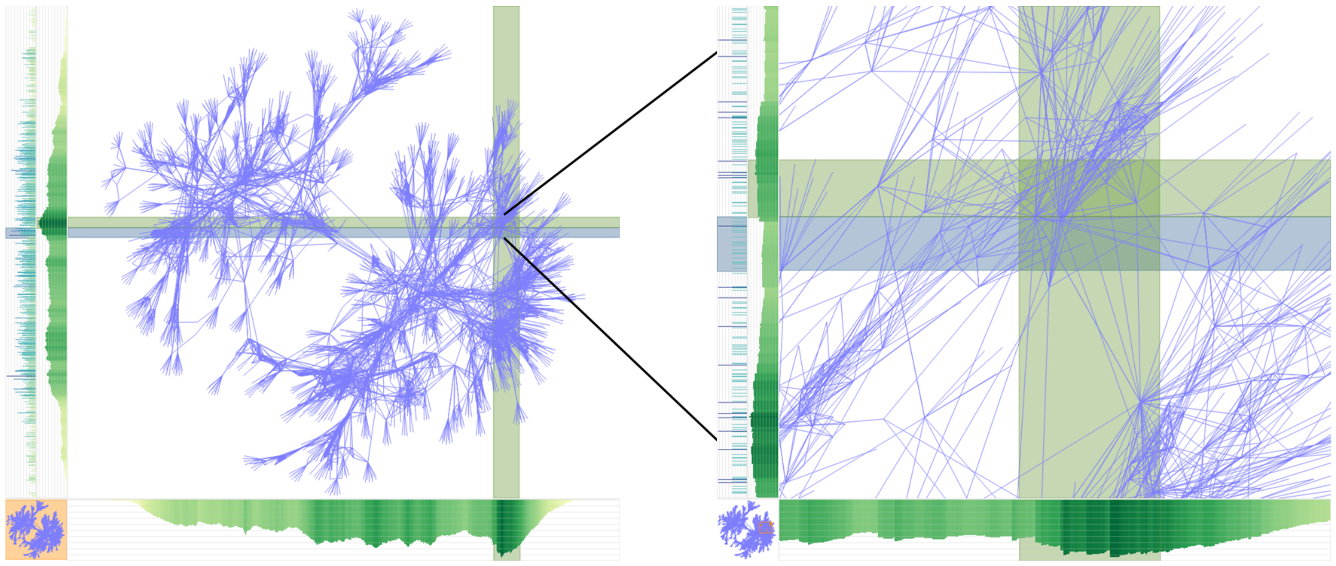

Figure 1. An overview of our technique: The graph view shows a node-link diagram and allows standard interaction like zooming, panning, and picking. The minimap shows a smaller version of the node-link diagram; for orientation, the visible part of the graph is highlighted (focus region). Metric views can be attached to the sides of the graph view. This additional information cannot be integrated in the node-link diagram itself due to clutter and overdraw.

In addition to adapting the visualization, there are many interaction techniques to cope with clutter and dense graphs in general. The MoireGraphs approach [24] arranges nodes in a radial layout and employs a focus+context metaphor to show interesting nodes enlarged with their neighborhood. The EdgeLens method [25] distorts links in the region of a lens to resolve ambiguities. Similarly, Tominski et al. [26] allow for local exploration of dense regions in hierarchies and graph visualization with a fish-eye metaphor. EdgeAnalyzer [27] is an interaction technique to locally group links and render them differently so that their paths can be traced more easily and ambiguities are resolved.

These methods already allow for great flexibility and freedom when dealing with graphs. Thus, our approach is complementary to them and tackles the problem of showing additional information that cannot be included in a node-link diagram. We achieve this by attaching additional views on different aspects of the data directly to the node-link diagram. Our visual design adopts the strategy of multiple coordinated views [4] that we apply and adapt to the specific needs of graph visualization. Network visualization frameworks such as Cytoscape [28] or Gephi [29] are also designed for graph-related tasks and provide additional perspectives on the graph data. However, none of them provides views like ours on the graph data in a similar way. In short, our Graph Metric Views can be combined with any variant of node-link diagram and layout method to facilitate the exploration of complex graphs and additional information attached to those graphs.

Our visualization technique is loosely related to NetLens [30], a tool for visual query refinement. It allows the user to explore relational data by showing different metrics as bar charts. Through selection and filtering, it is possible to drill down into many aspects of a data set. In addition to showing data-related metrics, we also visualize the relational structure of the data set. Furthermore, we utilize layout-related as well as graph-theoretic metrics besides data-inherent information.

Our technique is closely related to VINCENT [31]. There, graphs are laid out radially and have bar charts of network centralities attached. These centralities are node metrics based on graph-theoretic properties of a graph, like the degree or closeness. VINCENT uses histograms of these centralities but shows them detached from the graph visualization. Our approach shares the same motivation, namely that often more information is needed in a graph visualization than can be meaningfully presented in that same visualization. Yet, we do not enforce any layout for the graph and allow for any kind of metric related to the data set. Furthermore, we attach the histograms directly to the graph view to allow for visual linking between the metrics and the node-link diagram.

III. Graph Metric Views

Relational data can be modeled as a (directed) graph $G = (V,E)$ consisting of n vertices $V := \{v_1,\dots,v_n\}$. The set of edges $E \subseteq V \times V$ denotes the relations between vertices, while their weight $w : E \longrightarrow \mathbb{R}$ describes another attribute of the data set. Additional attributes have to be shown through visualizations of the graph, for example by laying out the vertices as node-link diagram or by using different shapes for nodes and edges. In the following, we denote elements of the abstract graph by vertices and edges, while nodes and links are used when we refer to the layout or visual representation of the graph.

Typically, node-link diagrams suffer from visual clutter and overdraw that are the result of high spatial densities of elements within regions of the graph. In Section II, we mentioned techniques to cope with this problem. Yet, it is not easy to convey further information in node-link diagrams. Thus our goal was to design a technique that amends existing node-link diagrams and allows for enriched interaction methods.

Figure 1 depicts an overview of our visualization technique. A *graph view* shows the users' focus, an interesting region of a node-link diagram, while the *minimap* provides its context. We attach additional linked views to the sides of the node-link diagram that show arbitrary metrics of the data and its representation (see Figure 2). These *metric views* depict a one-dimensional projection of the display space of the node-link diagram to one of its axes. Section III-A describes the different kinds of metrics, while Section III-B details how these can be visualized.

A. Metric Types

For our metrics, we distinguish between node characteristics and link characteristics. For example, the vertex degree is a property of nodes, while the edge weight and the link length are link properties. In both cases, we can identify three types of metrics that differ in the way their values are determined.

Layout-related metrics are based on the visual representation of a graph, i.e., the rendering of the nodes and links as graphical shapes. As such, these provide straightforward information, like the number of nodes and links or the sum of the edge weights along the abscissa or ordinate. They can be further refined, e.g., by considering or counting only incoming or outgoing links per node, link intersections, or angles between links.

Graph-theoretic metrics provide deeper insight into the relational and hierarchical aspects of graph data sets. While cliques and communities can often be seen in certain layouts easily, they may be hard to spot or discern due to clutter and overdraw. Additional metrics like centralities (e.g., eccentricity, closeness) are often interesting but cannot be embedded in the node-link diagram itself if there are no visual channels available [31]. We show these in our metric views as supplemental information attached to the original graph layout.

Data-inherent metrics can be used to correlate arbitrary values from the data to the layout. As an example, let us consider a software project where typical call graphs contain many attributes besides the call relation: file sizes, lines of code, hierarchy levels, various types of complexity, and so on. There is also information from the development and test environment (e.g., execution durations, code coverage) as well as the version control (e.g., revision count, last committer). These metrics apply to nodes (e.g., lines of code of a method) as well as to links (e.g., the distance of two methods in the package hierarchy).

Since metric values are accumulated layout-related, graph-theoretic, or data-inherent properties of a graph and its visual representation, it is possible to further aggregate them by using the minimum, maximum, or average values instead, where applicable. For example, the average length of links can be used as an indicator for the aesthetics of layout methods.

B. Metric Views

For visualization, the metric values from Section III-A are accumulated into bins with regard to the display space of a graph layout, similar to histograms. For this accumulation, we project nodes and links onto a one-dimensional axis. Thus, the metric views to the left/right relate to the ordinate of the node-link diagram, while the views at the top/bottom relate to the abscissa.

The computation of smooth histograms is a classical problem of statistical graphics. Typically, kernel density estimation techniques are employed [32]. For histograms of node characteristics, we have the traditional problem of building histograms from point samples. For edge-oriented metrics, we build the histograms for intervals that correspond to the projection of the edge to the histogram axis. In either case, we use a box-filter as kernel, and use pixel-sized bins for the representation of histograms. The width of the box filter can be chosen by the user; the default value is given by the width of the node's visual shape in image space.

The metric views can be stacked in arbitrary order to allow for investigation of multiple attributes of the data set and graph. While we do not limit the number of active views, it is foreseeable that too many views quickly clutter the display. On the one hand, users need to decide how many views they use and how much screen space they allocate. On the other hand, computing and rendering the metrics can be expensive, thus slowing down the user interface as a whole. This issue is alleviated through caching and manual pausing of views, i.e., users can disable data and visual updates to specific views if they are too sluggish.

C. Interaction

All views are linked with each other through brushing and linking. By separate filtering in each metric view, it is possible to drill down into interesting regions of the graph that are occluded in the node-link diagram. Furthermore, marking regions in the metric views highlights the corresponding parts in the graph view so that users can see which parts of the graph contribute to the metric value. This is important to mentally link the elements of the node-link diagram to peaks or other visual features in the metric views. Additionally, visual elements of the graph that have been highlighted through selection in a metric view can be used to

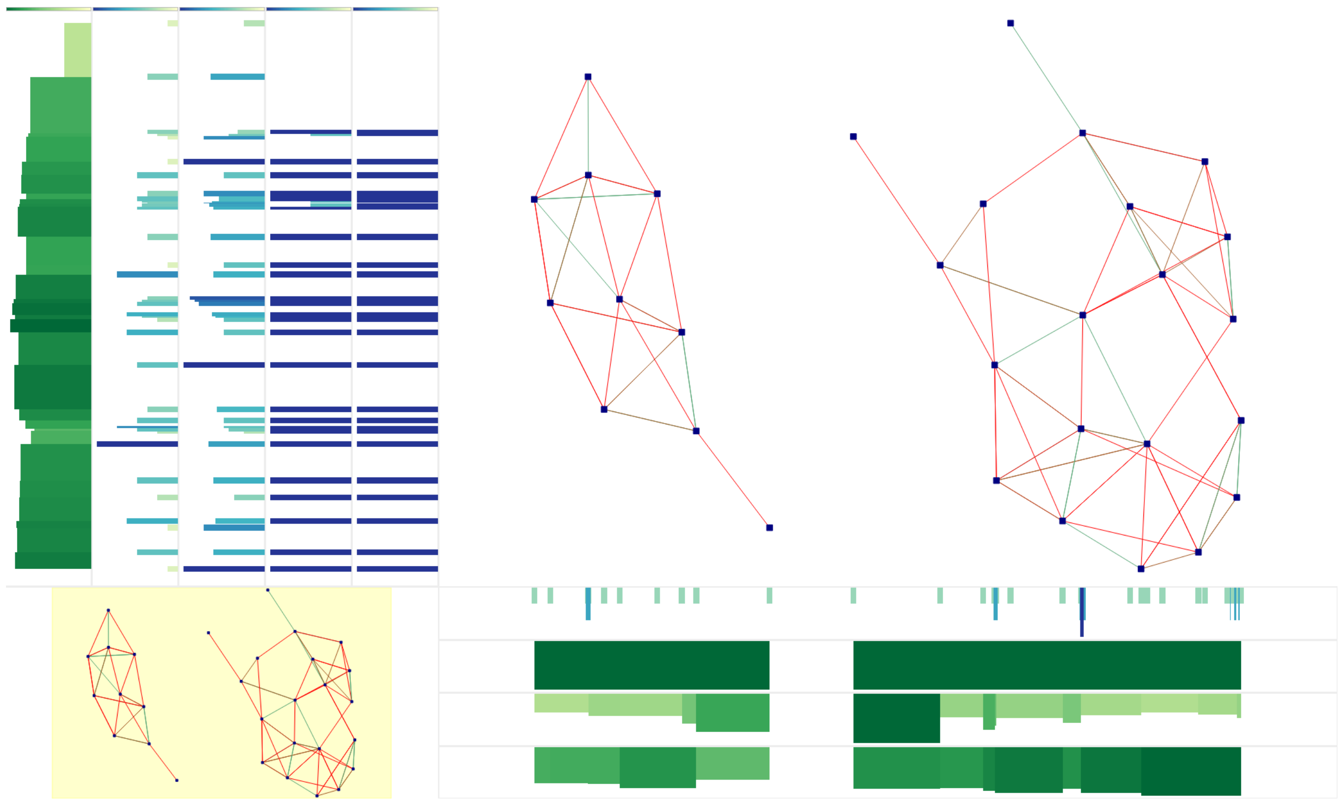

Figure 2. Multiple graph metric views can be shown in arbitrary order at the sides of the graph view for additional information. Resizing those increases their readability but decreases the available space for the graph view. Users resolve this trade-off by manually arranging the metric views and their sizes.

filter the graph itself. For example, nodes that are not marked can be hidden, while marked nodes (and their neighborhood) will remain visible.

The users' focus is the graph and its visual representation, respectively, which typically employs most of the screen-space (see Figure 1). Exploration is possible through typical navigation like scrolling, zooming, picking, and filtering. Interactions with the graph view automatically update all metric views so that these always correspond to the visible area of the graph. Furthermore, interactions with the metric views affect the graph view, for example scrolling in the metric view or highlighting regions.

To preserve the users' mental map, we provide a minimap of the whole graph (see bottom left of Figure 1). This shows also the context of the graph view in which the visible region is highlighted. Users can navigate the graph through the minimap in the same way as through the graph view (i.e., zooming, panning), which in turn updates the graph view as well as the metric views.

Information like node labels and edge weights are available as tooltips on-demand. Metadata and settings—for example vertex and edge count, or the layout used—can be shown as overlay anytime. Through saving the graph fully or partially along with the configuration of the metric views, it is possible to collaborate or continue an analysis at a later point in time.

D. *Implementation Details*

We implemented our technique using C++11 and Qt 5.1. The igraph [33] library is used for all graph-theoretic operations and computing the layouts. While igraph is highly flexible, it is a single-threaded CPU-only library. Thus, computing layouts of large graphs or finding cliques might take a long time. To improve the responsiveness of the user interface we moved all igraph operations to a dedicated thread. A progress bar coupled to igraph's built-in progress reporting informs the user about the status of these operations. For example, while cliques are sought, we can still navigate the graph or interact with the metric views.

IV. Case Study

We demonstrate how our visualization approach can be used for a typical real-world problem of visual graph analysis in the field of software engineering. In software systems, it is important to analyze properties like coupling and dependencies. These systems are often modeled as directed call graphs where vertices represent full-qualified method names. Edges between two vertices represent the source method calling the target method.

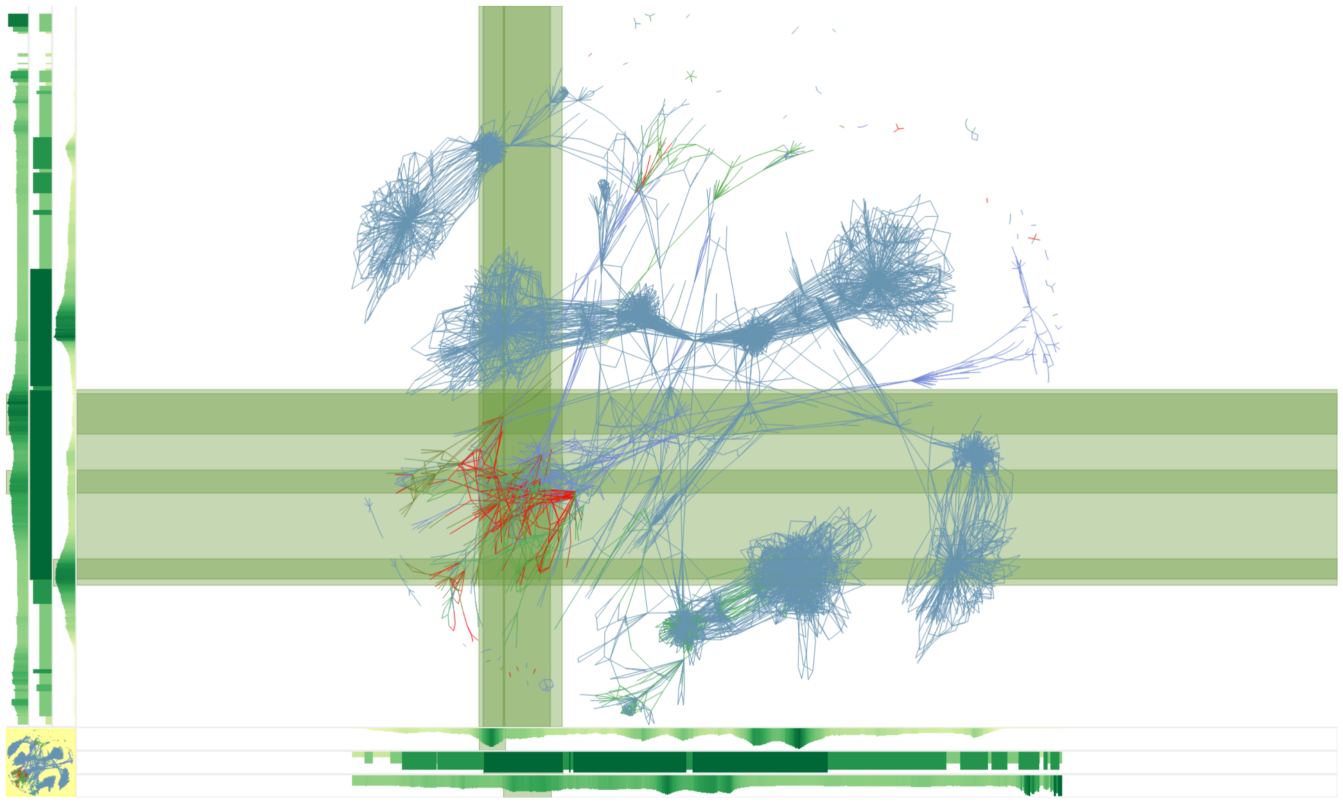

Figure 3. A software call graph depicting the frequency of method calls. The edge weight was mapped to a topological color table, thus red links indicate the most frequent calls. The metric views show (from outermost to innermost metric) the average and maximum package distance, as well as the number of edges. Through selection in the metric views, the interesting region in the node-link diagram is visually linked to the software metrics.

Figure 3 shows such a call graph of Cobertura[1], a code coverage tool for Java. At this stage, the analyst does not know details about the data set, except for very basic high-level information: It is a directed graph with 5,000 vertices and 11,319 edges, each vertex contains the name of a method and its package, and edge weights represent the number of times a method called another one. Now, the analyst wants to obtain insights into that data set, following the Information Seeking Mantra [3]. We use a Fruchterman-Reingold [10] layout to find methods that call many other methods; such dense areas indicate high coupling, which is unfavorable in software engineering. Yet, from the initial layout it is unclear from which packages these methods originate, i.e., from the Java SDK, the Cobertura developers, or some third party dependencies. Though this information is important, it cannot be embedded into the graph layout easily.

We use a topographic color coding, mapping low edge weights to blue, medium weights to green, and high weights to red. This way, we can immediately spot an area with many red links (see Figure 3), which indicate the highest method call frequency and should be investigated first. To explore this area we attach a data-specific metric that shows the distance in the package hierarchy between two nodes as property of the links. We use both the average as well as maximum package distance to reveal outliers that might be concealed in the average (see the outer metrics in Figure 3). This reveals some interesting patterns in the metrics, for example two peaks in the average package distance near the region with the red links. To find out whether these peaks are due to the high density of links in that area, we attach a layout-related metric for the number of edges (see the innermost metric in Figure 3). We see that the peaks are in fact within regions with few links and thus few method calls. We assume that these must be very specific methods that gather information from many places, since they have high package distance and call each other frequently, but not as often as other methods.

To further investigate this, we highlight the peaks in the metric views to correlate them to the node-link diagram. Then we zoom into this region, which updates the metric views, revealing new peaks. We repeat this process until we reach a view of the graph where the second metric, the maximum package distance, is constant for the visible part of the node-link diagram (see Figure 4(a)). Subsequently we remove that metric view, since it serves no purpose at this point. The metric values are very similar in this region, so we filter on the metric views to only show regions where

[1]http://cobertura.github.io/cobertura/

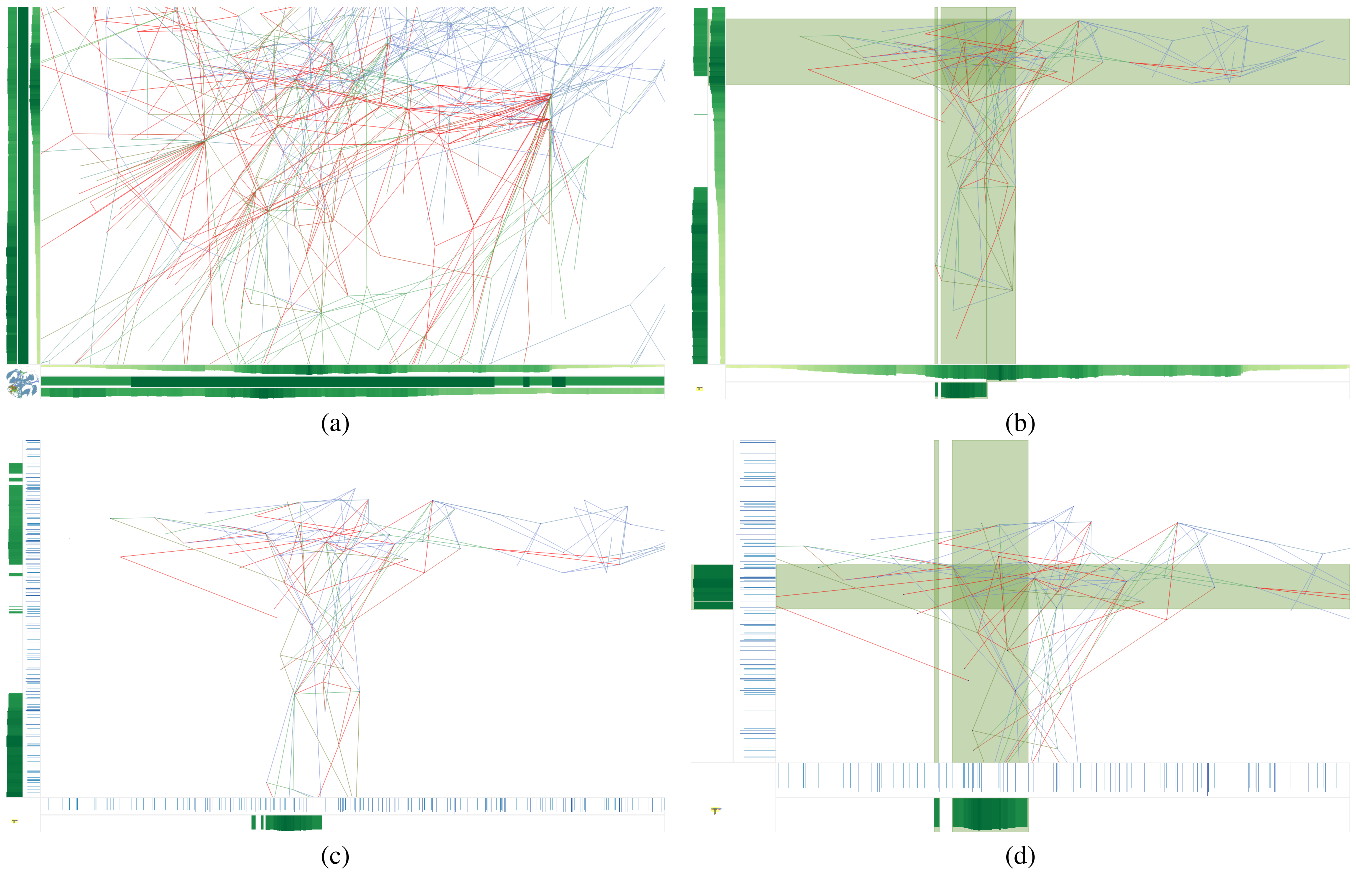

Figure 4. Exploration of the region with the most frequent method calls. (a) The visible part of the node-link diagram mostly contains calls with the highest package distance. (b) Filtering on the metric views allows us to find the regions in the node-link diagram that contain the highest average package distances. Parts of the node-link diagram that were not selected have been hidden. (c) A metric has been attached to show the package depth of the methods. (d) The data set has been reduced to few methods which can be investigated manually and looked up in the source code.

the average package distance is above a threshold. To further improve the view on the graph we hide all nodes and links that are not highlighted by any metric view (see Figure 4(b)).

Our focus is now on a region with high average package distance as well as few links (with respect to the rest of the node-link diagram). We now replace the edge count metric with the node metric that represents the maximum hierarchy level of methods (see Figure 4(c)). We spot an outlier (a singular method with the highest hierarchy level in this part of the node-link diagram) and can easily identify the method as CopyFiles.copy in the package net.sourceforge.cobertura.reporting.html.files, which is not called often. Finally, we increase the thresholds gradually on the metric views and mark the new peaks. Now only few methods are left for consideration (see Figure 4(d)), which can be inspected manually through their tooltips. At this point the analyst can look up these methods in the source code to understand their behavior.

In conclusion, our approach allows for top-down interaction to match structures of the node-link diagram to the metric views as well as vice versa. This interaction between metric views and node-link diagram allows for a drill-down into large data sets. Visual signatures and prevalent structures can be found in the node-link diagram (cluster-like structures) and correlated to the metric views (intervals with high values). Furthermore, outliers are visible as peaks in the metric views as well as in the node-link diagram as links with substantially different colors than the others. Most importantly, our metric views show attributes of the data that were not visible before.

V. Limitations

A problem of our approach is rooted in the position of our metric views. Let us assume a node-link diagram with two dense regions in the same vertical area. While a vertical metric view will accumulate the values correctly, it is impossible to discern the contributions of the clusters, since they are in the same region of the metric. This ambiguity can already be resolved, for example by moving one cluster out of the graph view, so that their contribution to the metric can be seen separately. Alternatively one can add the same metric horizontally and highlight the affected regions; note though that this fails if there are also nodes or links in that region that are not part of the cluster. In essence, this is a drawback we need to investigate further, possibly employing a user study with different scenarios.

The metric views are drawn at the sides of the graph view by design, but this is also problematic in some ways. For one, metric views become harder to read with each additional metric view for a given side of the graph view. Then, resizing those metric views to improve the readability reduces the available screen space for the graph. This is a trade-off between available space on the screen and the usability of our approach. We need to investigate this phenomenon in a user study to determine when the usability falls off and the metric views become impractical.

As we detailed in Section III-B, some metrics are expensive, for example calculating link intersections (quadratic complexity) and finding shortest paths for each node. This disrupts the user experience and slows the graph exploration down due to an unresponsive user interface. On the one hand, users can disable offending metrics and enable them later on. On the other hand, it is not obvious which metrics to disable, since their slowness might depend on the graph, its underlying data, and the metric itself. Additionally, the overall user experience is influenced by the compute-intensity and number of active metrics. A reasonable approach could be to monitor the runtime for each metric calculation and disable those above a threshold when many updates or interactions are queued.

VI. Summary and Future Work

In this paper, we have introduced *Graph Metric Views*: supplemental views to a node-link diagram that show various metrics related to network data. The main advantage of our approach is that we show additional information of such data sets that cannot be integrated into the node-link diagram due to visual clutter and exhaustion of visual channels. Thus we can reveal patterns that would otherwise be hidden in hairball-like structures or are not visible at all. We complement existing graph drawing techniques by basing our metrics on layout-related attributes of the node-link diagram, graph-theoretic properties, and data-inherent information. Using the metric views, we can explore aesthetics criteria for graph visualization as well as drill into unknown data sets easily. Our coordinated views are connected through brushing and linking, resulting in an interactive technique for analysis of local and global properties of graphs. We have illustrated the usefulness of this graph analysis approach by exploring a software call graph.

In future work, we will evaluate our technique in a controlled user study with typical tasks that are required when reading and understanding graph structures. We also want to extend our approach to dynamic graphs, i.e., those that change over time, and investigate how it applies to further layouts and non-traditional graph visualizations. Finally, we plan to look into further, more complex metrics as well as alternative visualizations of the metrics themselves.

Acknowledgements

This work was partially funded by the Federal Ministry of Education and Research of Germany (BMBF) as part of the FeToL project.

References

[1] L. Euler, "Solutio problematis ad geometriam situs pertinentis," *Commentarii academiae scientiarum Petropolitanae*, vol. 8, p. 128–140, 1741.

[2] R. Rosenholtz, Y. Li, J. Mansfield, and Z. Jin, "Feature congestion: a measure of display clutter," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '05. ACM, 2005, p. 761–770.

[3] B. Shneiderman, "The eyes have it: a task by data type taxonomy for information visualizations," in *Proceedings of the IEEE Symposium on Visual Languages*, 1996, pp. 336–343.

[4] J. Roberts, "State of the art: Coordinated multiple views in exploratory visualization," in *Proceedings of the Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization*, 2007, pp. 61–71.

[5] M. Scherr, "Multiple and coordinated views in information visualization," *Trends in Information Visualization*, pp. 37–45, 2010.

[6] D. A. Keim, "Information visualization and visual data mining," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 1–8, 2002.

[7] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1998.

[8] M. Kaufmann and D. Wagner, Eds., *Drawing Graphs–Methods and Models*, ser. Lecture Notes in Computer Science. Springer, 2001, vol. 2025.

[9] D. Archambault, J. Abello, J. Kennedy, S. Kobourov, K.-L. Ma, S. Miksch, C. Muelder, and A. C. Telea, *Temporal Multivariate Networks*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2014, no. 8380, pp. 151–174.

[10] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Software: Practice and Experience*, vol. 21, no. 11, p. 1129–1164, 1991.

[11] E. R. Gansner, Y. Koren, and S. North, *Graph Drawing by Stress Majorization*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, no. 3383, pp. 239–250.

[12] H. C. Purchase, R. F. Cohen, and M. James, "Validating graph drawing aesthetics," in *Proceedings of the Symposium on Graph Drawing*, ser. GD '95. Springer, 1996, p. 435–446.

[13] C. Ware, H. Purchase, L. Colpoys, and M. McGill, "Cognitive measurements of graph aesthetics," *Information Visualization*, vol. 1, no. 2, pp. 103–110, 2002.

[14] G. Ellis and A. Dix, "A taxonomy of clutter reduction for information visualisation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1216–1223, 2007.

[15] D. Holten, "Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 741–748, 2006.

[16] D. Holten and J. J. van Wijk, "Force-directed edge bundling for graph visualization," *Computer Graphics Forum*, vol. 28, no. 3, pp. 983–990, 2009.

[17] W. Cui, H. Zhou, H. Qu, P. C. Wong, and X. Li, "Geometry-based edge clustering for graph visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1277–1284, 2008.

[18] R. van Liere and W. de Leeuw, "GraphSplatting: visualizing graphs as continuous fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 2, pp. 206–212, 2003.

[19] 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*, vol. 17, no. 12, pp. 2344–2353, 2011.

[20] C. Hurter, O. Ersoy, and A. Telea, "Graph bundling by kernel density estimation," *Computer Graphics Forum*, vol. 31, p. 865–874, 2012.

[21] M. Zinsmaier, U. Brandes, O. Deussen, and H. Strobelt, "Interactive level-of-detail rendering of large graphs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2486–2495, 2012.

[22] D. Holten and J. J. v. Wijk, "A user study on visualizing directed edges in graphs," in *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, ser. CHI '09. ACM, 2009, pp. 2299–2308.

[23] M. Burch, C. Vehlow, N. Konevtsova, and D. Weiskopf, "Evaluating partially drawn links for directed graph edges," in *Graph Drawing*, ser. Lecture Notes in Computer Science, M. v. Kreveld and B. Speckmann, Eds. Springer Berlin Heidelberg, 2012, no. 7034, pp. 226–237.

[24] T. J. Jankun-Kelly and K.-L. Ma, "MoireGraphs: radial focus+context visualization and interaction for graphs with visual nodes," in *Proceedings of the IEEE Symposium on Information Visualization*, 2003, pp. 59–66.

[25] N. Wong, S. Carpendale, and S. Greenberg, "EdgeLens: an interactive method for managing edge congestion in graphs," in *Proceedings of the IEEE Symposium on Information Visualization*, 2003, pp. 51–58.

[26] C. Tominski, J. Abello, F. van Ham, and H. Schumann, "Fisheye tree views and lenses for graph visualization," in *Proceedings of the International Conference on Information Visualization*, 2006, pp. 17–24.

[27] A. Panagiotidis, H. Bosch, S. Koch, and T. Ertl, "EdgeAnalyzer: exploratory analysis through advanced edge interaction," in *44th Hawaii International Conference on System Sciences (HICSS)*. IEEE Computer Society, 2011, pp. 1–10.

[28] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker, "Cytoscape: A software environment for integrated models of biomolecular interaction networks," *Genome Research*, vol. 13, no. 11, pp. 2498–2504, 2003.

[29] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *Proceedings of the Third International AAAI Conference on Weblogs and Social Media*, 2009, pp. 361–362.

[30] H. Kang, C. Plaisant, B. Lee, and B. Bederson, "NetLens: iterative exploration of content-actor network data," in *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, 2006, pp. 91–98.

[31] A. Kerren, H. Köstinger, and B. Zimmer, "VINCENT–Visualization of network centralities," in *Proceedings of the International Conference on Information Visualization Theory and Applications*, 2012, pp. 703–712.

[32] B. Silverman, *Density Estimation for Statistics and Data Analysis*, ser. Monographs on Statistics & Applied Probability. Chapman and Hall, 1986, vol. 26.

[33] G. Csárdi and T. Nepusz, "The igraph software package for complex network research," *InterJournal Complex Systems*, vol. 1695, 2006.