2010 14th International Conference Information Visualisation

The Network Lens: Interactive Exploration of Multivariate Networks Using Visual Filtering

Ilir Jusufi, Yang Dingjie, and Andreas Kerren
School of Computer Science, Physics and Mathematics (DFM)
Linnaeus University
SE-351 95 Växjö, Sweden
Email (corresponding author): ilir.jusufi@lnu.se

***Abstract*—Networks are widely used in modeling relational data often comprised of thousands of nodes and edges. This kind of data alone implies a challenge for its visualization as it is hard to avoid clutter of network elements if using traditional node-link diagrams. Moreover, real-life network data sets usually represent objects with a large number of additional attributes that need to be visualized, such as in software engineering, social network analysis, or biochemistry. In this paper, we present a novel approach, called *Network Lens*, to visualize such attributes in context of the underlying network. Our implementation of the Network Lens is an interactive tool that extends the idea of so-called magic lenses in such a way that users can interactively build *and* combine various lenses by specifying different attributes and selecting suitable visual representations.**

***Keywords*-graph drawing; network analysis; multivariate network visualization; magic lenses; interaction techniques;**

I. Introduction

The visualization of complex and large networks is aimed to give insight into different patterns between relations of data. Most of these networks are represented following a node-link metaphor comprised of thousands of nodes and edges. Visual analysis tools have to deal with huge, complex and dynamic data sets and cope with different challenges, such as to avoid clutter and to increase the people's understanding of graphs, also known as readability of the network [1], [2]. In practice, each network object may additionally have a number of attributes that are important to be visualized in context of the overall network presentation.

Finding a good solution to visualize those attributes is an ongoing challenge in various network visualization domains, such as software engineering, social network analysis, or biochemistry. One of the most simple software engineering examples is the visualization of relationships between classes. Each class may have a number of methods, fields, and other properties. In addition, we could compute different measurements (software metrics) for such elements that are important to maintain and to improve the software engineering processes. Examples for popular software metrics are lines of code, number of classes, etc. From the perspective of the visualization community, the measured values of a whole software metric suite form a multivariate data set. To give an additional example from social network analysis research, we could imagine that the relationships between staff members of a large company should be analyzed. The set of all relationships forms a network. Of course, each staff member also has an individual set of own properties, such as age, gender, qualification, position, etc. The question is now: how can we visualize those attributes together with the network drawing?

There are a number of approaches to address this problem. The simplest one is to present a list of all attributes and their values in a separate view on the display. This could be a textual list or a more complex visual representation, such as parallel coordinates [3] or star plots [4], as the attributes form a multivariate data set. Another way is to use glyph-based approaches where we can represent attributes by using visual features of the glyphs, e.g., shape, orientation, color, or size [5]. A more detailed presentation of related work is given in Section II. In this paper, we will discuss an extension of the traditional magic lens idea (cf. Section II-C), called the *Network Lens*, applied to traditional node-link graph layouts. We have developed a prototype implementation of this Network Lens that enables users to interactively build various lenses by specifying different attributes and selecting different visual representations [6]. Each time we apply our Network Lens on a network element, it visualizes its attributes (or a subset of them) by using a specific visual representation, i.e, the standard node representation is replaced by a new visualization or diagram. A neat example would be the one of time-depended attributes, which is a standard problem in biochemical network analysis. A domain expert could analyze experimental data measured over time, which are attached and represented by the network nodes at time step t_i, for example. Without changing his/her current visualization setting, a specific Network Lens instance could be used to show the data at a time step t_{i-1} for a set of interesting nodes. In this way, our approach can support the visual analysis process of multivariate networks by having an additional generic tool that can be adapted to standard visualization tools and can extend already existing views to show node and edge attributes of the underlying network. Users are able to create individual lenses, to store

1550-6037/10 $26.00 © 2010 IEEE
DOI 10.1109/IV.2010.15

them for later analyses, and to combine them to analyze related attribute sets.

The remainder of the paper is organized as follows: Section II describes related work in the field of multivariate network visualization. The next Section III presents our approach and interaction possibilities. A typical use case is given in Section IV. We conclude the paper in Section V and give an outlook to future work.

II. Related Work

In the following, we describe related work starting with a brief discussion of graph drawing fundamentals, continue with approaches dealing with multivariate network visualization and illuminate the traditional magic lens approach which has inspired our work.

A. Graph Drawing (GD)

In this paper, we distinguish between *graphs* and *multivariate networks*. A (simple) *graph* $G = (V, E)$ consists of a finite set of vertices (or nodes) V and a set of edges $E \subseteq \{(u, v)|u, v \in V, u \neq v\}$. Whereas, a multivariate network consists of an underlying graph G plus additional attributes that are attached to the nodes and/or edges. Graph drawing algorithms compute a layout of the nodes and the edges, mainly based on so-called node-link diagrams [7]. They play a fundamental role in network visualization. Particular graph layout algorithms can give an insight into the topological structure of a network if properly chosen and implemented. The graph readability is affected by quantitative measurements called *aesthetic criteria* [2], such as minimization of edge crossings, displaying the symmetries of the graph drawing, constraining edge lengths, etc. [1]. Thus, graph drawing generally deals with the ways of drawing graphs according to the set of predefined aesthetic criteria [8]. Note that we focus on traditional GD approaches in this paper. There are further possibilities to represent graphs, such as matrix representations [9] or hybridizations between both approaches [10].

It is vital to have a good graph layout algorithm when doing any kind of network visualization. In our case, a sufficient layout algorithm would reduce the scalability problem, which is one of the ongoing challenges in information visualization [11]. Implementing good graph drawing algorithms is usually complicated and time consuming. Therefore, a number of different open source libraries were developed, such as JUNG (Java Universal Network/Graph Framework) [12] which is used in our approach.

B. Multivariate Network Visualization

A good drawing algorithm will not solely solve our problem to visualize multivariate networks. Several approaches can be found in the literature that attempt to offer a solution to this problem, i.e., *multiple and coordinated views*, *integrated approaches*, and *semantic substrates*. We will discuss these concepts in the following paragraphs.

Multiple and Coordinated Views: One possible solution to the problem stated above is to combine several views and present them together. This approach allows the use of the most powerful visualization techniques for each specific view and data set [5], [13].

Shanon et al. [14] present the application of this idea in the network visualization domain. They use two distinct views: one view shows a parallel coordinate approach [3], and the other view displays a node-link drawing of a graph. Their tool is equipped with a variety of visualization and interaction techniques, but both views must be coordinated by linking or brushing [15]. This might introduce a scalability problem with large networks. The drawback of multiple views is that they split the displayed data because of the spatial separation of the visual elements.

Integrated Approaches: To provide a combined picture, attributes and the underlying graph should be displayed in one view. "Integrated views can save space on a display and may decrease the time a user needs to find out relations; all data is displayed in one place." [5]. One example is described in Borisjuk's et al. [16] work on the visualization of experimental data in relation of a metabolic network. The authors used a straightforward approach by employing small diagrams instead of representing the nodes as simple circles or rectangles. Each diagram, e.g., a bar chart, shows experimental data that is related to the regarded node. This approach provides a view to all available information, but the embedding of the visualizations into the nodes causes the nodes to grow in size. This issue may affect the readability of the network due to the overlaps that may appear when the number of nodes and the attributes is high [2]. Thus, is does not scale well.

In general, the aforementioned example is an instance of embedding *glyphs* into networks. They are graphical entities that convey multiple data values via visual attributes, such as shape, color, position, or size [17]. Outside of network drawings, their orientation on the display is irrelevant. There is a considerable literature on glyphs and a lot of tools use them, e.g., Glyphmaker [18] or tools for visualizing software metrics [19].

Semantic Substrates: In order to avoid clutter in multivariate network visualization, some researchers realized the idea of so-called semantic substrates that "are non-overlapping regions in which node placement is based on node attributes". Shneiderman and Aris introduced this idea and combined it with sliders to control the edge visibility and thus to ensure comprehensibility of the edges' end nodes [20]. PivotGraph [21] uses a grid-layout to show the relationship between (node) attributes and edges. The tool aggregates nodes and edges with identical values for the selected attributes. The size of the resulting PivotGraph nodes and edges represent the degree of aggregation; color is used to code the attributes. Another approach was recently presented by Pretorius and van Wijk [22]. They arrange

edge labels in a list and place rectangular regions containing source and target nodes at each side. These regions are partitioned according to the node attributes and connected via straight lines with corresponding edge labels. One conceptual drawback of these approaches is that the underlying graph topology is not (completely) visible, which is tackled by our Network Lens even if it only works locally as described at the end of Section I.

C. Magic Lenses

Lenses, like magnifying or fisheye lenses, are widely used metaphors in visualization environments [23], [24]. Many of these so-called focus+context approaches use distortion techniques [15].

The most closely related work compared to our lens implementation is a magic lens developed by Bier et al. [25], [26]. They describe it as a user interface tool that combines a region on the screen with an operator that is used to change the view of object beneath that region. Bier et al. depict an analogy to a real magnifying lens over a newspaper when describing the interaction of their tool. Their magic lens could serve as simple magnification tool, but could also provide some sort of visual filtering of the object viewed through it. The lenses usually filter out some kind of graphical object or perform some image processing computations on the objects. They also introduce a possibility to combine different lenses and, in consequence, to get a new lens with inherited features of all the lenses being combined. In contrast, our approach is driven by attribute semantics and not focused on pure graphical filters.

EdgeLens is an interactive tool for managing edge congestion in graphs [27]. It locally improves the visual perception of graphs with high edge density. It lets edges flow around the focal point and makes it possible to read node labels for example. This approach is orthogonal to our ideas and could be combined with them.

Baudish et al. [23] made some efforts to study the usability of magic lens based techniques by applying the metaphor for their focus+context interaction interface and compared it with overview+detail and pan+zoom interfaces. Their results suggest a significant time saving in their experimental tasks and a higher subjective satisfaction. Gutwin et al. [28] performed a small comparative study based on three types of fisheye view interfaces in context of graph layout tasks. These studies suggest that lenses could be regarded as an advisable tool for network visualization environments even if there are differences between competing fisheye varieties.

III. The Network Lens

Our Network Lens tool facilitates the interactive exploration of complex networks using visual filtering. It offers a way to visualize additional network attributes, while preserving the overall network topology and context. On the one hand, users can gain insights into the whole network by exploring the overall visualization of the network, in terms of topology and connectivity of particular nodes of the network. On the other hand, they can get more details about *desired* attributes by focusing on specific node(s), i.e., instead of showing all attributes at the same time, the user has the possibility to choose a subset of all available attributes he/she is currently interested in *and* to interactively explore the network elements on the basis of the selected attributes. If wished, the remainder of the attributes could still be represented by standard node representations. Thus, our tool also avoids possible clutter and scalability issues if additional information is embedded. In the following, we discuss our approach, present our software prototype and describe a typical use case scenario.

A. Approach

Our idea is based on the magic lens approach of Bier et al. [25], [26], but extended in such a way that users can interactively build different lenses by selecting desired attributes and assigned visual representations in context of the network visualization. Each network element could have different quantitative (incl. ordinal) and nominal attributes that might be important to be visualized. Each attribute or group of attributes can be shown more or less efficiently by using different visualization approaches depending on the underlying nature of the data. Therefore it is important to have the flexibility to choose the way of showing various attributes. Additionally, users should be able to combine different lenses to simplify and speed up the process of creating new lenses by using drag and drop interaction. It is also desirable to have a possibility to set up and store a number of lenses for each working session as well as for later use. In this way, the users can create "custom-built" lenses and switch between them interactively during the exploration of the network visualization.

The GUI is divided into three parts, as shown in Fig. 1. The main area on the left hand side displays a traditional node-link network visualization and occupies the major part of the tool window. At first, an overview of the entire graph topology is displayed after loading the input network (using a GraphML specification [29]). Nodes can be drawn in various ways. Currently, the user can map the value of an arbitrary attribute to the color saturation of the rectangles representing the nodes. Additionally, edge weights are mapped to the thickness of the edge lines. Those edge weights depend on the application data and could be derived from the strength of the relationship between two node entities or similar. These graphical features are combined with the possibility to choose from five different graph layout algorithms and to modify the node positions manually. All together support the user to identify interesting parts of the network and to rearrange nodes by manual clustering (automatic clustering could be easily added). Then, the multivariate network can be explored further. Ideally, one

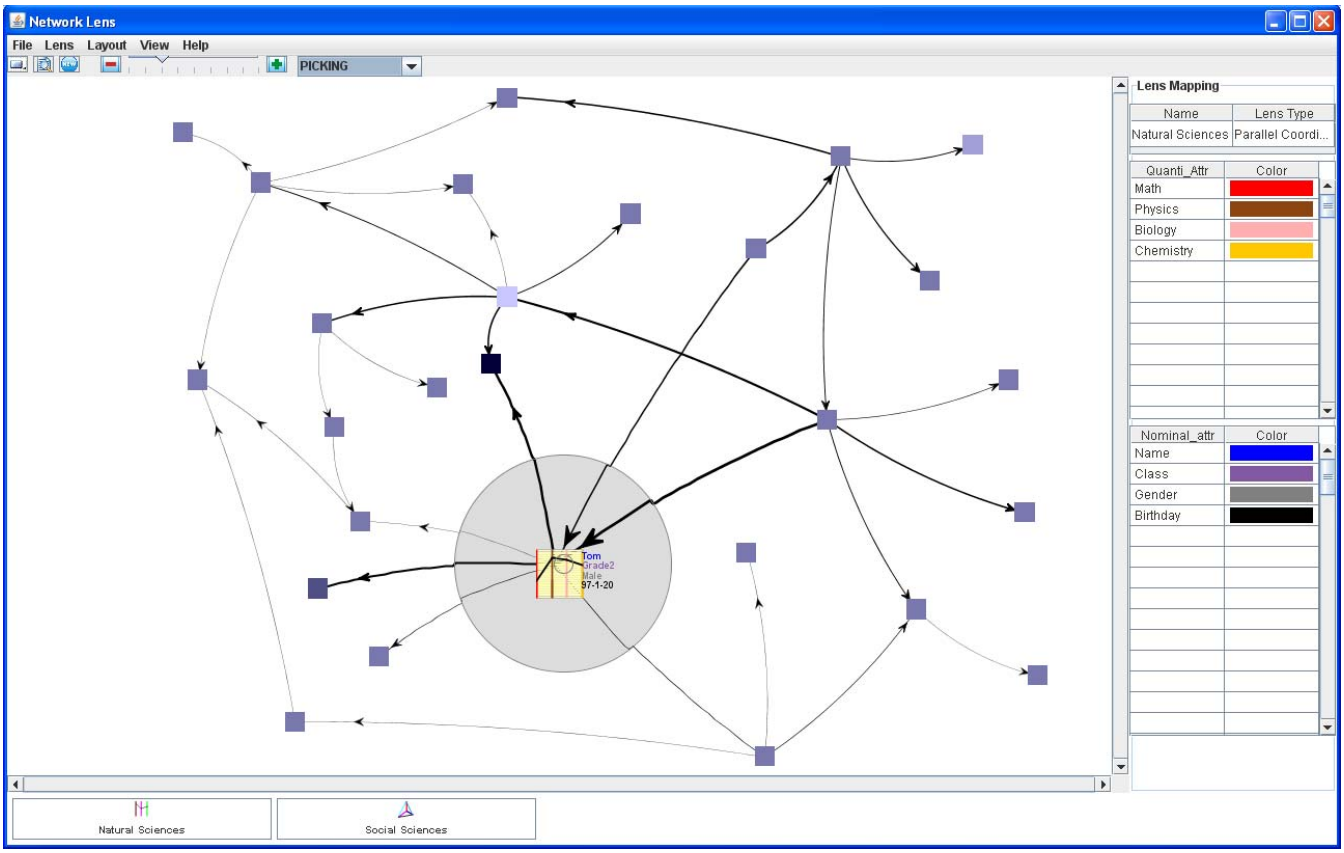


Figure 1. Overview of the Network Lens tool. The GUI is divided into three distinctive parts: the main network visualization area, the lens information area on the right hand side, which we call *Lens Mapping*, and the bottom part where all user-produced lenses are preserved. The multivariate network data is based on students ($\Rightarrow$ nodes) who share the same courses ($\Rightarrow$ edges) and their individual course grades and personal information ($\Rightarrow$ attributes).

would like to use different glyph representation or visual metaphors to visualize attributes that are related to nodes (Section II-B, *Integrated Approaches*). This feature is not implemented in our prototype yet, since it was not in our main focus at the beginning of the prototype development.

In the center of the network visualization, an active Network Lens "Natural Sciences" is displayed. It currently covers one node only and shows a small Parallel Coordinate diagram with four quantitative and four nominal attributes belonging to that node. The user is able to move the lens with the mouse or to translate the graph behind the lens. Of course, it is also possible to change the size of the lens itself and to zoom the lens content. Whereas the node representations remain undistorted, the edges are distorted if the lens is moved. But, the edges within the lens can be easily mapped to the original edges in the background, either by slightly moving of the lens or by following the edge lines along the lens rim. The right panel (called Lens Mapping) holds the legend of the lens attribute color mappings. The bottom part of the tool window keeps created lenses built by the user; more details on the creation of lenses is given in the following.

Network Lens Creation: The lens creation process is started by selecting the New Lens option from the Lens menu in the main window. At first, the user has to name the lens. Lens names should be self-descriptive in order to avoid forgetting the "nature" of the lens in case a lot of them are created and stored. For instance, if the user is exploring a data set based on students relationships and their grades for different subjects (see Fig. 1), he/she might want to create different lenses for different groups of classes and name them accordingly; for example, an *Art Lens* showing attributes of subjects like Painting, Sculpture, etc. or a *Science Lens* with subjects like Mathematics or Physics.

The next step is to specify different visual representations, attributes, as well as a suitable color mapping for the newly created lens by means of the form presented in Fig. 2. By selecting one of the options in the Select Lens Type list, users can assign a suitable visual representation of a lens concerning quantitative and ordinal data. Currently, there are two variations of star plots, one bar chart, and one parallel coordinate visualization to choose from, see Fig. 3. The Illustration icon at the top of the dialog box shows a sample view of the selected lens type.

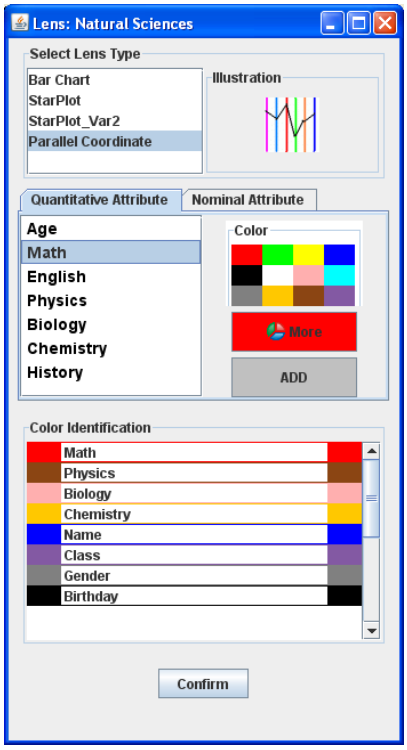
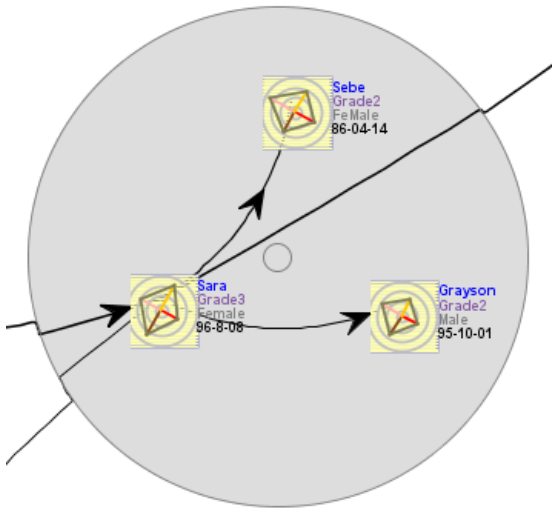


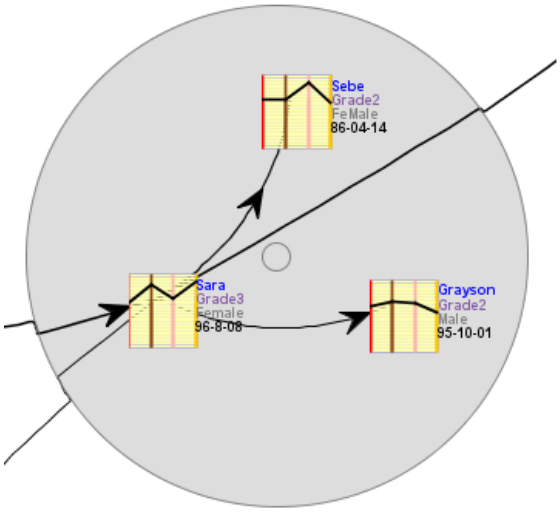
Figure 2. A dialog box used to create and edit lenses. Users can specify the type of the lens, select different attributes provided by the input data set and assign their color mapping.

Two tabs Quantitative Attribute and Nominal Attribute are used to specify quantitative and/or nominal attributes to be visualized by the lens. The user can select the attributes from a list and add them to the lens. Color mapping is done automatically in case the user does not specify the colors. We have provided 12 standard colors as suggested by C. Ware [7] to achieve a good visual perception. The final color mapping is shown in a separate list in the lower part of the dialog box. Analogically, the user can create several lenses repeating the steps described above. New lenses are added in form of buttons to the bottom part of the GUI. By clicking on a lens button, the corresponding lens is activated.

If the user moves a specified lens over the network, then the original node representations are replaced by the aforementioned visualizations and text labels as already described above. The text labels have a transparent background by default. Especially if the underlying graph has many edges, this can be irritating because of potential overlaps. To avoid this problem, the user is able to switch to an opaque background for text in our tool. However, this could lead to not very appealing lens experiences. Another way to improve this situation and a possibility for future work would be the combination with the EdgeLens approach discussed in Section II-C.



(a) Star plot diagram, 2nd variant



(b) Parallel coordinate representation

Figure 3. Two different visual representations used to display the attributes. Attributes can be color coded automatically, or the color can be specified by user. Nominal attributes, such as *Name* and others are represented by text labels on the right hand side of the small visualizations.

Combining Lenses: Inspired from optics, our approach provides a combination of already created lenses by "laying them one over another". In our case, this is a rather complex issue when using several lenses with different visualization metaphors for the same set of attributes: if one lens visualizes attributes using one representation, and another lens uses a different one, their combination will not be as straightforward as it could look in traditional, graphics-oriented magic lens approaches. We solved this problem in a pragmatical way: our users can simply drag and drop one lens button over the another one. Then, a new dialog box Combining Lenses appears as shown in Fig. 4.

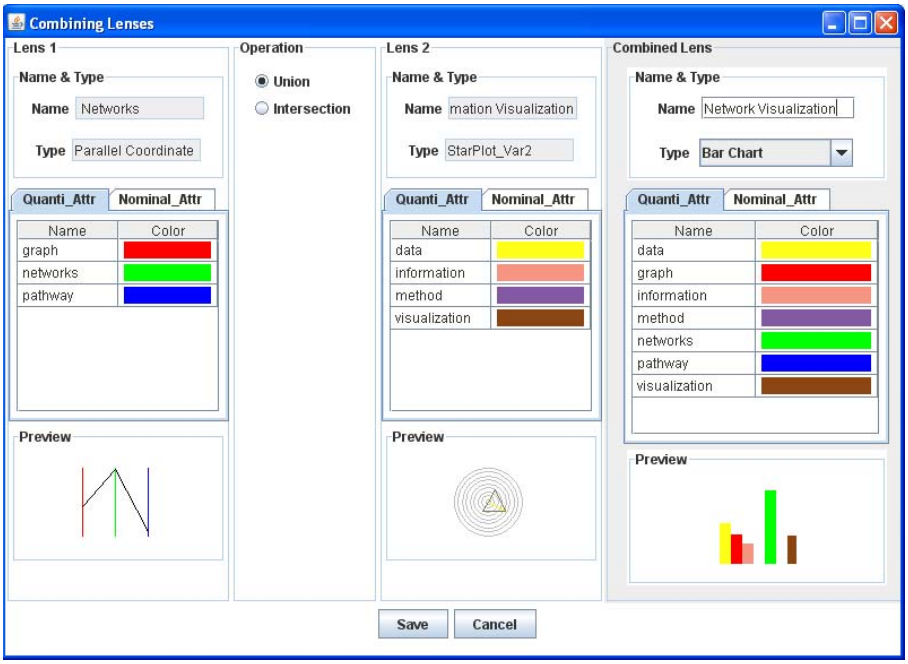


Figure 4. A dialog box used to combine different lenses.

There are four groups of controls in this dialog box. Lens 1 and Lens 2 show information about the lenses that are going to be combined. This information includes lens name and type, the list of attached attributes including their color coding, as well as a preview icon. In our current version, the user can choose between two basic set operations, *Union* and *Intersection*, which are used to create a new set of attributes from the given sets of Lens 1 and Lens 2. Two radio buttons within the Operation group allow the selection. The result of the chosen operation is shown in the Combined Lens group, that follows the same GUI layout as Lens 1 and Lens 2. Here, the user should name the combined lens and select the resulting lens type. If the input lenses have the same type, then the default type of the combined lenses corresponds to the input type. Additionally, it is possible to change the attribute colors of the combined lens. After saving, a new lens button appears again at the lower part of the main window, cp. Fig. 1.

IV. Application Scenario

We created a multivariate network dataset using various text documents (a set of research papers published by our group). Each node represents one document; its attributes stand for the occurrences of a specific word within the document. An edge between two nodes represents the *similarity* between them, respectively, the co-occurrence of a attribute set. The weight of the edge (shown as the thickness of the edge line) represents the degree of similarity calculated as the sum of the minimum values of the attributes. We pruned this input data using a blacklist of frequently used words and by setting a threshold for the minimum occurrence of words and for minimum edge weights. Fig. 5(a) shows a tool screenshot using this input data set.

Our data set is comprised of 24 text documents. Imagine that we would like to explore the content of these documents without reading them. For example, we are interested in documents whose content is related to “algorithms”. We map the color saturation of the nodes representations to the value of the attribute “algorithm”. Lighter colors represent higher values and vice versa as shown in our screenshot example. Then, we can immediately identify several documents. At this point, we would like to narrow our interest down to those documents with content related to “algorithms” and “networks” (or graphs). We create a new lens named Network and select attributes (keywords in this case) that would give insight to documents related to networks. We move the lens over the lighter colored nodes (those with content about algorithms), and we identify a couple of documents with relatively high values of specific attributes, see Fig. 5(b). These documents contain many occurrences of the words “graph”, “nodes”, and “pathways”. This could mean that these document might not be strictly related to visualization since they probably describe some computational algorithms related to biochemical pathways. Therefore, we import a previously configured lens named Visualization and combine it with our lens Network using the Union set operator in order to create a new lens Network Visualization. After exploring the documents once again, we get more insight about the documents connected to network visualization and algorithms. Finally, we discover the couple of document identified earlier fit our criteria as shown in Fig. 5(c).

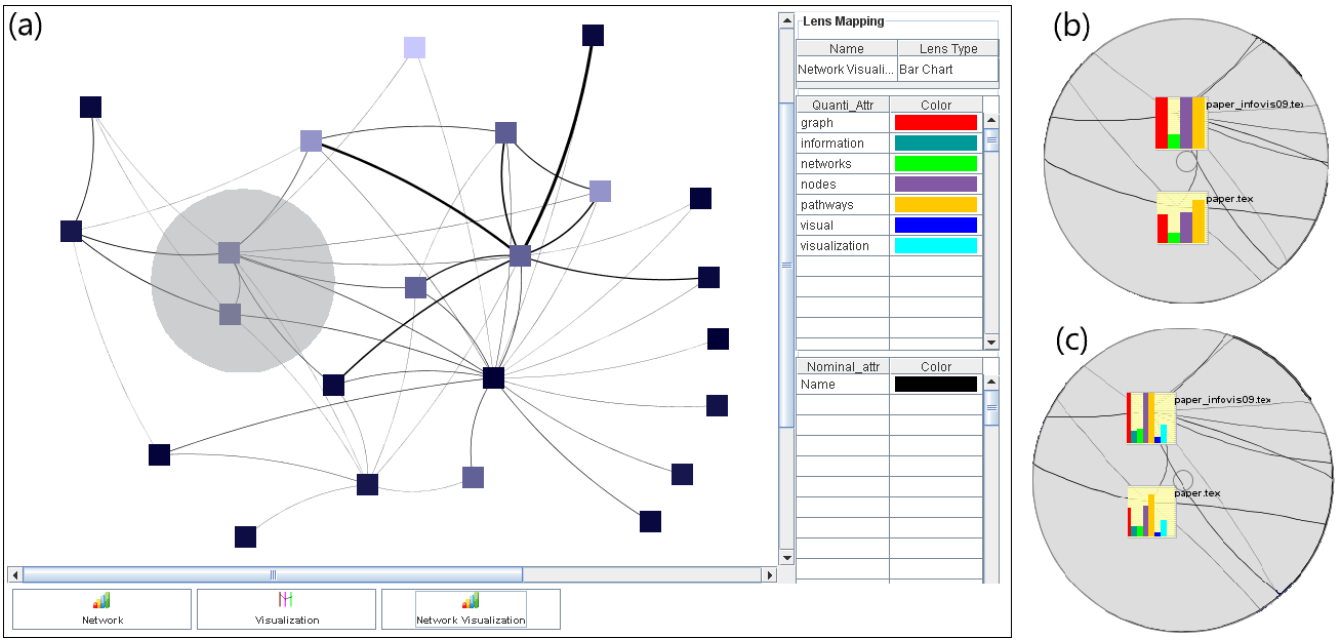


Figure 5. The transparent gray circular disk in the network visualization view (a) represents the focus of the lenses discussed in the following. The top right image (b) represents the view of the lens named Network, while the bottom right image (c) shows the view rendered by the Network Visualization lens.

Other scenarios can be easily identified, such as network lenses for representing data quality (Challenge #7 in [11]). The quality of node attributes could be described by ratings or statistical errors, which form a new set of attributes covered by a specific network lens. Those quality lenses could be applied in context of the original attributes.

V. Conclusion

We presented a novel approach for the interactive exploration of multivariate networks using an intuitive visual filtering method for the local representation of node attributes. Based on related work and our literature review, there has been no attempt to combine network visualization and magic lenses to address this problem. Our approach offers a solution while minimizing the side effects of earlier approaches, such as readability issues with respect to integrated graph drawing or display size in multiple view approaches. In more detail, our Network Lens joins the advantages of magic lens approaches and integrated graph drawing and reduces the overloading problem of the latter issue. It provides flexibility in attribute filtering and selection of suitable visual representations. Having a possibility to build various lenses and using them for various tasks in exploring multivariate networks makes our approach flexible and applicable to different application domains. Thus, domain experts can customize their visual filters based on their knowledge and expertise in order to gain insight into their relational data sets. Additionally, our users are able to combine different lenses in an intuitive way. This makes it easier to create new lenses for a deeper analysis of multivariate network data. Furthermore, our approach could be combined with other network visualizations as discussed in Section II.

We have implemented and presented a software prototype that demonstrates our approach. Our future work will be focused on designing a usability study, implementing more visual representations and experimenting with time-dependent data and data quality issues.

Acknowledgments

We would like to thank Daniel Cernea for carefully proofreading the final version of this paper.

References

[1] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[2] A. Kerren, A. Ebert, and J. Meyer, Eds., *Human-Centered Visualization Environments*, ser. LNCS Tutorial 4417. Heidelberg: Springer, 2007.

[3] A. Inselberg and B. Dimsdale, "Parallel coordinates: A tool for visualizing multi-dimensional geometry," in *IEEE Visualization*, 1990, pp. 361–378.

[4] J. Chambers, W. Cleveland, B. Kleiner, and P. Tukey, *Graphical Methods for Data Analysis*. Belmont, CA: Wadsworth, 1983.

[5] C. Görg, M. Pohl, E. Qeli, and K. Xu, "Visual Representations," in *Human-Centered Visualization Environments*, ser. LNCS Tutorial 4417, A. Kerren, A. Ebert, and J. Meyer, Eds. Heidelberg: Springer, 2007, pp. 163–230.

[6] Y. Dingjie, "The Network Lens," Master's thesis, Linnaeus University, Sweden, 2010 (to appear).

[7] C. Ware, *Information Visualization: Perception for Design*, 2nd ed. Morgan Kaufmann, 2004.

[8] C. Chen, *Information Visualization. Beyond the Horizon*, 2nd ed. London Berlin Heidelberg: Springer-Verlag, 2004.

[9] J. Abello and F. van Ham, "Matrix zoom: A visual interface to semi-external graphs," in *Proceedings of the IEEE Symposium on Information Visualization*. Los Alamitos, CA, USA: IEEE Computer Society, 2004, pp. 183–190.

[10] N. Henry, J.-D. Fekete, and M. J. McGuffin, "Nodetrix: a hybrid visualization of social networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, pp. 1302–1309, 2007.

[11] R. S. Laramee and R. Kosara, "Challenges and Unsolved Problems," in *Human-Centered Visualization Environments*, ser. LNCS Tutorial 4417, A. Kerren, A. Ebert, and J. Meyer, Eds. Heidelberg: Springer, 2006, pp. 231–254.

[12] J. O'Madadhain, D. Fisher, and T. Nelson, "JUNG - Java Universal Network/Graph Framework," 2009. [Online]. Available: http://jung.sourceforge.net/

[13] J. C. Roberts, "Exploratory visualization with multiple linked views," in *Exploring Geovisualization*, A. MacEachren, M.-J. Kraak, and J. Dykes, Eds. Amsterdam: Elseviers, December 2004. [Online]. Available: http://www.cs.kent.ac.uk/pubs/2004/1822

[14] R. Shannon, T. Holland, and A. Quigley, "Multivariate graph drawing using parallel coordinate visualisations," University College Dublin, School of Computer Science and Informatics, Tech. Rep. 2008-6, 2008. [Online]. Available: http://www.csi.ucd.ie/files/ucd-csi-2008-6.pdf

[15] R. Spence, *Information Visualization: Design for Interaction*, 2nd ed. Prentice Hall, 2007.

[16] L. Borisjuk, M. Hajirezaei, C. Klukas, H. Rolletschek, and F. Schreiber, "Integrating data from biological experiments into metabolic networks with the dbe information system." *In Silico Biol*, vol. 5, no. 2, pp. 93–102, 2005.

[17] M. O. Ward, "A taxonomy of glyph placement strategies for multidimensional data visualization," *Information Visualization*, vol. 1, no. 3/4, pp. 194–210, 2002.

[18] W. Ribarsky, E. Ayers, J. Eble, and S. Mukherjea, "Glyphmaker: Creating customized visualizations of complex data," *Computer*, vol. 27, no. 7, pp. 57–64, 1994.

[19] M. Pinzger, H. Gall, M. Fischer, and M. Lanza, "Visualizing Multiple Evolution Metrics," in *Proceedings of the ACM Symposium on Software Visualization (SoftVis '05)*, St. Louis, Missouri, 2005, pp. 354–362.

[20] B. Shneiderman and A. Aris, "Network visualization by semantic substrates," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, pp. 733–740, 2006.

[21] M. Wattenberg, "Visual exploration of multivariate graphs," in *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*. New York, NY, USA: ACM, 2006, pp. 811–819.

[22] A. J. Pretorius and J. J. van Wijk, "Visual inspection of multivariate graphs," *Comput. Graph. Forum*, vol. 27, no. 3, pp. 967–974, 2008.

[23] P. Baudisch, N. Good, V. Bellotti, and P. Schraedley, "Keeping things in context: a comparative evaluation of focus plus context screens, overviews, and zooming," in *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 2002, pp. 259–266.

[24] M. Sarkar and M. H. Brown, "Graphical fisheye views," *Commun. ACM*, vol. 37, no. 12, pp. 73–83, December 1994.

[25] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose, "Toolglass and magic lenses: the see-through interface," in *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1993, pp. 73–80.

[26] M. C. Stone, K. Fishkin, and E. A. Bier, "The movable filter as a user interface tool," in *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 1994, pp. 306–312.

[27] N. Wong, S. Carpendale, and S. Greenberg, "Edgelens: An interactive method for managing edge congestion in graphs," in *Proceedinngs of the IEEE Symposium on Information Visualization*. Los Alamitos, CA, USA: IEEE Computer Society, 2003, pp. 51–58.

[28] C. Gutwin and C. Fedak, "A comparison of fisheye lenses for interactive layout tasks," in *GI '04: Proceedings of Graphics Interface 2004*. Waterloo, Ontario, Canada: Canadian Human-Computer Communications Society, 2004, pp. 213–220.

[29] U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, and M. S. Marshall, "Graphml progress report (structural layer proposal)," in *Proceedings of the 9th International Symposium on Graph Drawing (GD '01)*, ser. LNCS, P. Mutzel, M. Jünger, and S. Leipert, Eds., vol. 2265. Springer, 2002, pp. 501–512. [Online]. Available: http://gdea.informatik.uni-koeln.de/524/

[30] A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North, Eds., *Information Visualization, Human-Centered Issues and Perspectives*, ser. Lecture Notes in Computer Science. Springer, 2008, vol. 4950.