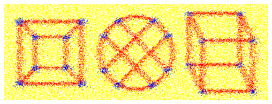

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 21, no. 1, pp. 5–27 (2017)
DOI: 10.7155/jgaa.00399

OnGraX: A Web-Based System for the Collaborative Visual Analysis of Graphs

Björn Zimmer[1] *Andreas Kerren*[1]

[1]Department of Computer Science, ISOVIS Group, Linnaeus University, Vejdes Plats 7, SE-35195 Växjö, Sweden

Abstract

The visual analysis of complex networks is a challenging task in many fields, such as systems biology or social sciences. Often, various domain experts work together to improve the analysis time or the quality of the analysis results. Collaborative visualization tools can facilitate the analysis process in such situations. We propose a new web-based visualization environment which supports distributed, synchronous and asynchronous collaboration. In addition to standard collaboration features like event tracking or synchronizing, our client/server-based system provides a rich set of visualization and interaction techniques for better navigation and overview of the input network. Changes made by specific analysts or even just visited network elements are highlighted on demand by heat maps. They enable us to visualize user behavior data without affecting the original graph visualization, are robust against layout changes, and are user-sensitive in a sense that the current analyst is able to perceive which changes were made by others in asynchronous collaboration. In case of synchronous collaboration, an analyst can see where and what others are currently analyzing in the network visualization. Thus, our approach addresses critical collaborative visualization challenges, for instance, awareness and coordination of user activities or pointing to interesting objects. We evaluated the usability of the heat map approach against two alternatives in a controlled user experiment. In addition, the results of a domain expert review are described in this article.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
October 2015	February 2016	April 2016	July 2016	July 2016
		Published: January 2017		
	Article type: Regular paper	Communicated by: E. Di Giacomo and A. Lubiw		

E-mail addresses: bjorn.zimmer@lnu.se (Björn Zimmer) kerren@acm.org (Andreas Kerren)

1 Introduction

With the growing size and availability of large and complex data, the cooperative analysis of such data sets is becoming an important new method for many data analysts as cooperation might improve the quality of the analysis process [25] and help to analyze data sets efficiently. One crucial observation is that collaborators—who are often spread across the globe—would like to seamlessly drop in and out of ongoing work [18]. On the one hand, the collaborative analysis process can take place in a joint online session where everybody is working simultaneously on one data set, discussing and changing it together in real-time to create better analysis results. Here, different experts might want to see what the others are doing, and if there are possibilities to coordinate their efforts and find a common ground [5, 11]. On the other hand, the experts work on the data set whenever they find the time (i.e., asynchronously) to avoid having to schedule and organize a virtual or physical meeting with a larger group of colleagues. Both situations cause specific problems that should be handled by tools which support collaborative work. For instance, while working independently, it would be helpful to see changes of the data performed by other analysts. Another interesting issue is to see which part of the data set has already been explored by others. Here, it is also interesting to know who changed the data: was an established expert working on a specific part of the data, or a new staff member who might not have the same experience as the expert?

To tackle the aforementioned problems in the context of collaborative network analyses, we have developed the visualization tool OnGraX [39, 40, 42]. Our system was designed for the *distributed asynchronous* and *synchronous* collaborative exploration of graphs in a modern web browser. Web-based visualization systems have the advantage of being accessible on the fly by just opening an URL in a browser and thus do not require the installation of additional software. This is a fundamental property for collaborative sensemaking, as analysts want to work together without too much setup overhead. One drawback of web-based visualizations is the performance while analyzing huge data sets. In such cases, native desktop applications are still the better choice. However, modern web browsers facilitate hardware accelerated rendering and, consequently, an increasing number of visualizations are implemented as web-based applications and published online. Note that we give a detailed explanation about the engineering aspects and technical challenges of implementing OnGraX as a web-based application in paper [42]. In contrast, we here propose interactive visualization techniques that

- help to coordinate work in a collaborative setting for *node-link diagrams* which may change their topology during the analysis process (referred to as dynamic graphs in the following) and
- assist analysts to identify previous activities performed by former users on these networks.

We exemplify our visualization approaches with the help of the collaborative analysis of metabolic networks from the Kyoto Encyclopedia of Genes and Genomes (KEGG) pathway database [23] due to our long lasting research collaborations with biologists/bioinformaticians at several research institutions [1, 19, 20, 21]. Building such biological networks is often based on complex experiments. In consequence, biologists of different domains and experience levels want to explore the resulting networks and check them for wrong entries or missing data and revise the networks wherever it is necessary. Usually, they only check parts of a network that are specific to their own field of expertise or interest. In this case it is important to know, what part of the network has already been checked and what part still needs attention. This can also be used as a kind of quality check: an area which has been investigated by many different experts is likely to have a higher quality than an area only investigated by one scientist. OnGraX supports such analysis tasks by providing methods for data awareness and coordination. Note that we retain this usage scenario in the rest of the paper except in the heat map evaluation (cf. Subsection 5.1) in order to attract a higher number of test subjects.

The remainder of this article is organized as follows. In the next section, we discuss related work in collaborative graph visualization. We describe our design decisions in Section 3 and explain OnGraX' interaction and visualization techniques for displaying user behavior in Section 4. A user experiment to evaluate our heat map approach for identifying previous user activities is discussed in Section 5 together with a domain expert review. We conclude our article in Section 6.

This article is an extended version of a paper presented at the *23rd International Symposium on Graph Drawing & Network Visualization* (GD '15) [41]. In this version, we added a short summary of the technical aspects of OnGraX in Subsection 3.2, as well as a discussion about an alternative coloring approach for the heat map visualization in Subsection 4.2. Figures 5 and 6 have been added to clarify features discussed in Section 4. Additionally, Section 5 has been extended by an expert review to assess our collaboration requirements (cf. Section 3) during a distributed synchronous analysis session in the domain of metabolic network analysis.

2 Related Work

Isenberg et al. [16] give a good overview of definitions, tasks and sample visualizations in the field of collaborative visualization. The authors define collaborative visualization as "the shared use of computer-supported, (interactive,) visual representations of data by more than one person with the common goal of contribution to joint information processing activities". They also provide an excellent summary of ongoing challenges in this field. The benefits of collaborative work were also discussed in an article on social navigation presented by Dieberger et al. [8]. Being able to see the usage history and annotations of former users might help analysts to filter and find relevant information more

quickly. In order to be able to work together during a synchronous session, users have to know each other's interactions and views on the data set, usually referred to as "common ground" [5, 11]. To find a common ground in node-link visualizations, we apply the techniques from the work of Gutwin and Greenberg [9]. They used secondary viewports and radar views to indicate other users' view areas and mouse cursor positions. We use a similar approach and show the viewports of other users as rectangles in the background of the graph visualization. Another work by Isenberg et al. [17] introduced the concept of collaborative brushing and linking, which "allows users to communicate implicitly, by sharing activities and progress between visualizations". The authors considered sharing activities during synchronous collaborations on a tabletop visualization for document collections. We adapt the concept and utilize it in node-link diagrams with the help of a heat map visualization for the exploration of interaction information in both asynchronous and synchronous, distributed sessions. An earlier article from Isenberg et al. [15] considers retrofitting existing visualization systems to enable co-located collaboration. They complemented the single-user graph visualization system NodeTrix [12] to support multiple mice inputs and one keyboard to host a collaborative session for multiple users in one room. However, they had to remove some of the features of the original system, since they were not designed to support multiple user input. Another tool for co-located collaboration was introduced by Mahyar and Tory [24]. Their tool CLIP supports collaborative work by sharing findings among a team of analysts in a co-located environment. Both systems are not suitable for our collaborative analysis problems, since they do not support asynchronous collaboration. There are other web-based developments such as ManyEyes [32] or Dashiki [27], which enable users to share data sets and analysis results online, but they do not support the interactive visualization of node-link diagrams in a web browser with real-time interactions for co-located as well as distributed collaboration.

During asynchronous sessions, users might want to share their thoughts, findings, or questions with subsequent collaborators while they explore a data set. Providing the possibility to add graphical and textual annotations is an important feature to communicate interesting parts of a data set directly within an existing visualization [4, 14]. These annotations should also keep their context if the data that they are referring to changes over time. Our tool OnGraX supports textual annotations that are tied to objects in node-link diagrams and can still be explored in context if the structure or the topology of a graph changes over time.

We utilize heat maps to analyze and identify highly frequented or edited parts of the graph based on user behavior. Patina [26] uses a similar approach but focuses on visualizing the usage of user interfaces, whereas our tool facilitates heat maps to visualize interactions of users with the data itself. To the best of our knowledge, heat map visualizations for representing data in combination with node-link diagrams are seldomly considered. Usually, they are used to visualize quantitative data in geovisualizations [30], as cluster heat maps [36], or for the visualization of eye tracking data to illustrate the quality of web site

designs, user interfaces, or graph layouts [28, 34], i.e., for evaluation purposes. One of the very few examples where heat maps are used in node-link diagrams is PLATO [35] which employs heat maps to visualize gameplay data.

3 Design Decisions

We carefully designed our system in terms of visual representations, interaction techniques, and analysis processes to support biologists/bioinformaticians in exploring and curating graphs from the KEGG pathway database. We decided to focus our work on node-link diagrams, since this is still the most accepted and preferred graph drawing metaphor, and our users are familiar with this kind of visualization.

3.1 Requirements

Our overall goal was to develop a visualization system that allows analysts spatially spread across multiple research labs or even countries to quickly start an analysis session and to work on large and complex networks together. A special problem that arises during the distributed analysis of graphs is that topology and structure of a graph are independent to the layout. Analysts might change the layout drastically during the analysis process, which complicates the task of keeping track of the graph objects and areas that users were most interested in. We also want our tool to support tracking and subsequent visualizing of all actions and graph changes performed by the users. This includes to keep track of the users' camera positions and use this data later to assist users in finding parts of a graph that were interesting to other analysts or have been edited a lot. The reason behind this is that users in a collaborative working environment do not always find the time to work together simultaneously. They would prefer to work on the data set whenever it is convenient for them. And in such a case, they would like to review changes that have been performed on the data set by other analysts in the past. Maybe, they also want to find out which part of the data set another analyst was looking at, since he/she might be an expert in the underlying application field and has another exploration pattern compared to less experienced users. Showing this data—the camera and mouse positions, the logged user views, and changes to specific objects—in the graph without changing the original node-link visualization was an important requirement for our users. Biologists are accustomed to existing layouts and drawing conventions of graphs from the KEGG pathway database. Thus, changing positions, color, or the shape of nodes to show the data which is collected during collaborations is not an option for our analysis tasks.

During their work, analysts would also like to share their thoughts, insights, and questions about specific nodes, edges or regions with other users. This could happen during a *synchronous* session where collaborators want to discuss their findings, or in an *asynchronous* session where users would like to share messages and pointers on specific nodes. Heer and Agrawala discuss these ideas

as "Common Ground and Awareness" and "Reference and Deixis" in their work on collaborative visual analytics [11]. In case of graphs that change their topology during the analysis process, single nodes or complete graph regions could be deleted from a graph, rendering old user annotations useless without the possibility to view them in their historical context. Thus, analysts need a way to quickly view the graph in a state when the annotation was originally written. Based on this discussion, we categorize our requirements as described in the following.

Collaboration Requirements (C-R)

1. Users should be aware of the position of other users in the same synchronous session.

2. Users should have possibilities to establish and keep a common ground with other users. Everyone should be aware of performed changes on the graph during a session.

3. They should have an option to discuss ongoing work through persistent chat channels and annotations.

Visualization Requirements (V-R)

1. Annotations should be viewable in their historical context. Thus, it should be possible for users to review old graph states.

2. Provide an easy and intuitive way for analysts to find out which regions of a graph were viewed and/or changed by former users.

3. Additionally, the visualization of this data should not interfere with the original node-link diagram.

3.2 General Setup and Technology

This subsection gives a brief overview of the technical aspects of OnGraX; a detailed discussion is available in [42]. OnGraX is based on a client-server architecture. The client is responsible for rendering the visualizations, providing the user interface, and sending action events and the user's current viewing area and mouse position to the server, which handles the data storage and distributes the events among all connected clients. Due to recent developments in web-based technologies, we decided to implement the client side by using HTML5 and JavaScript. The rendering of the node-link diagrams is achieved with WebGL [22]. WebGL is a JavaScript API for rendering 3D graphics on a GPU natively in a modern web browser without the need of additional plug-ins. In order to avoid low-level OpenGL programming, we use three.js [29]: a JavaScript library that builds on top of the WebGL specification.

The server is implemented in Java and runs as a web application on an Apache Tomcat server. It handles the real-time connections with WebSockets [37], stores the graphs, and keeps track of all performed user actions. This enables us to leave all computationally intensive parts on the server without affecting the visualization performance of the clients. One example of such a computationally expensive part is the heat map calculation which is described in Section 4. Whenever a user performs an action—such as joining or leaving a graph analysis session, or editing the graph—an action event is sent to the server. The server stores this action in a database and also forwards it to every connected client of this analysis session. All clients update their local graph visualization with the latest changes and add new actions to a list of recent events. Using a web application to store the graphs and all related data in a central place enables our users to start an analysis session whenever they want and also to seamlessly drop in and out of ongoing sessions.

4 Interaction and Visualization Techniques

Figure 1 shows an overview of OnGraX right after joining an ongoing graph analysis session. In this case, the user has joined a session where two other users, Bob and Sue, are already working in. Their viewports are represented as two dashed rectangles: Bob's view is shown in blue (bottom left) and Sue's view is shown in green (bottom right). All users in a session are listed as small icons at the left hand side of the screen. By clicking on one of the user icons, the camera moves to his/her current position in the graph. This feature provides a quick way to join and discuss another user's viewing area. Visualizing the viewports of other users helps us to tackle our first collaboration requirement (cf. C-R 1). An overview of the graph is rendered in the bottom right corner of the screen. Here, the user's camera position is shown as a blue rectangle. As in many other standard visualizations that use overview+detail [6], this rectangle can be dragged to another position in the overview in order to modify the detail view (the same can be done by clicking on the new position in the overview).

We use a standard node-link metaphor to visualize graphs in our system. The visualization uses tapered edges for directed graphs, as suggested by Holten and van Wijk [13], since they provide users with a faster way to find connected nodes as opposed to arrowhead edges. If another user selects one or more nodes, this will be visible to all other participants of the analysis session. An outline in the respective user color is added to a selected node; thereby the system adapts the outline shape to the corresponding node shape.

With the help of OnGraX, analysts are able to explore static graphs, and they can also edit the structure and topology of a graph. There are different kinds of use cases during the exploration and analysis of networks. Depending on the data, an analyst might want to keep the topology of a graph but completely change its layout, in order to find specific structures, hubs, or communities in the data. Also, mapping data attributes to the colors and size of nodes, or to

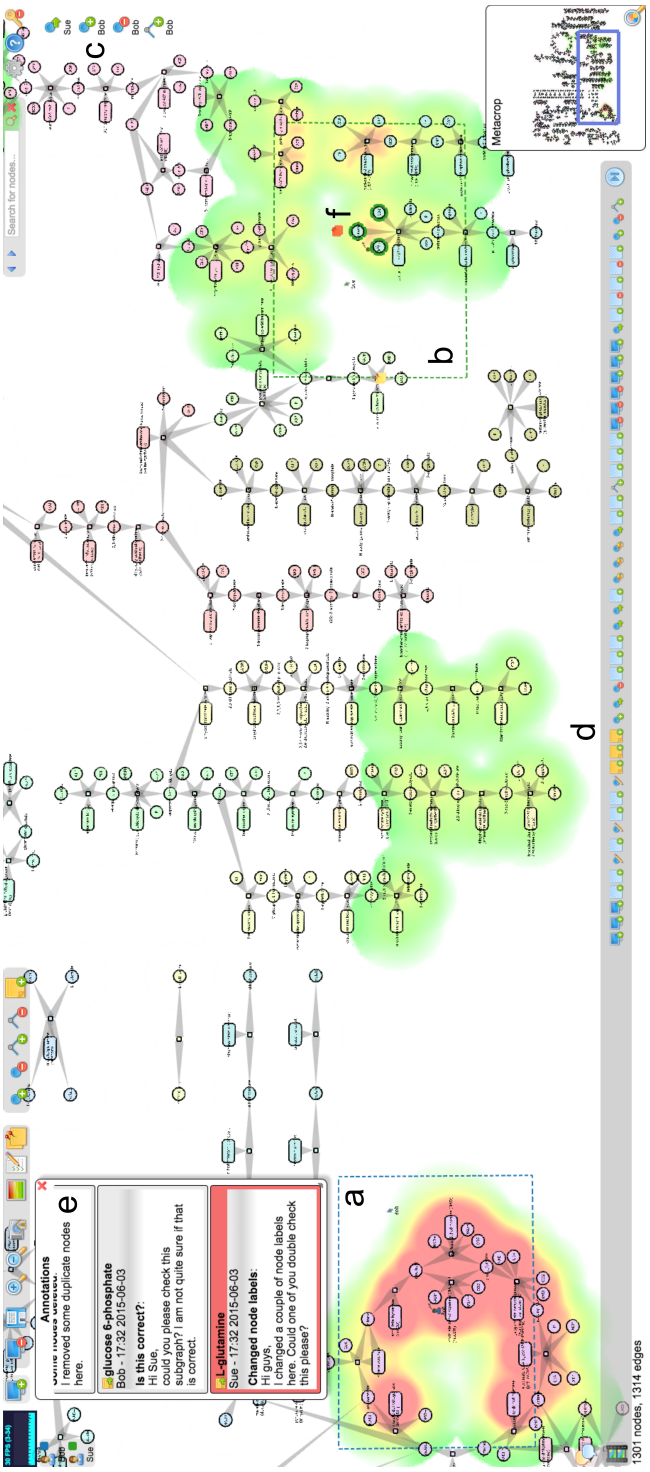


Figure 1: Overview of our system. The image shows a part of a biochemical network with 1,301 nodes and 1,314 edges. The blue and green dashed rectangles (see (a) and (b)) are the viewing areas (viewports) of two other users who are also exploring this graph simultaneously. In this concrete case, the underlying heat map highlights those nodes that were in the viewing area to all users during the last hour. Symbols in the top-right corner of the screen (c) assist analysts to keep track of recent actions performed by other users. The timeline (d) is used to temporarily revert the graph to a previous state and to replay applied changes. Analysts can pin text annotations to nodes and edges to discuss tasks, insights and questions with each other (e+f).

the width and length of edges could change the layout of a graph. In another situation, analysts could actually alter the topology of a graph by adding or removing edges and nodes. This could encompass a few nodes or edges, but it could also affect the graph on a bigger scale, for instance, if a complete subgraph is removed or added. And then, if the topology of a graph is altered on a huge scale, its layout could also change drastically. Comparing different states of a graph and keeping track of changes in such cases is a challenge which is addressed by Hascoët and Dragicevic [10]. Our use case in this work still includes altering the structure and layout of graphs, but on a smaller scale. The metabolic networks, which are analyzed and revised with OnGraX, already have an existing, handcrafted layout which is not changed by mapping different data attributes to the graph objects. Tasks for which OnGraX is used, usually only require adding or removing a few nodes and edges and/or changing their label, size or color manually. To make such graph changes more obvious—when they are performed by other users during a synchronous session, and to address the second collaboration requirement (cf. C-R 2)—we use short animations on the affected objects, similarly to the work of Gutwin and Greenberg [9]. For instance, the outlines for other users' node selections are animated shortly while they are added or removed, nodes are slowly moved to new positions instead of just jumping there after being moved by another user, and deleted nodes slowly vanish instead of just disappearing.

4.1 Annotations and Chat Links

In order to improve the communication among collaborators, our tool has a persistent chat channel for every graph session and offers the possibility to link chat messages to a position or a node in the graph. Users can use those chat links to move the camera to the linked object or position. A link to an arbitrary position might become obsolete after changes to the graph layout, but a message linked to a node or edge will always be valid as long as the object is not deleted. In addition, users can attach textual annotations directly to nodes or edges (cf. Figure 1, (e+f)). These annotations work as pointers from the graph visualization to text and vice versa. Clicking on an annotation in the graph visualization opens the annotation dialog and highlights the linked message. A click on an annotation in the dialog moves the camera to the object's position in the graph visualization. With the chat and annotation features, we address our last collaboration requirement (cf. C-R 3).

One problem with textual annotations and chat messages linked to objects is, that the original context in which an annotation or message was initially written could get lost if the respective graph region—where the link is pointing to—is changed during the course of a session or if the object with this link is deleted. We solve this problem by enabling analysts to temporarily revert the complete graph to an old state (similar to the timeline feature, see Section 4.3) by right clicking on a chat link or an annotation, giving them the possibility to view the graph in a state in which the annotation was originally written. This feature addresses our first visualization requirement (cf. V-R 1).

4.2 Visualizing User Behavior Data with Heat Maps

In order to provide users with a way to quickly find out which nodes or regions of a graph were viewed and/or changed by others (cf. V-R 2), we considered several options. It would be possible to map the corresponding data to the colors or the size of the nodes. Another option would be to use additional glyphs on/around the nodes which represent this data. Using glyphs would also allow us to show both the viewport data and the data for graph changes at the same time, as small bar charts for instance. The third option is a heat map-based visualization in the background of the graph visualization. We decided to omit mapping the data to the size of nodes, as this would interfere too much with the original graph layout and could introduce too many node overlaps. Additional options would have been to use contour lines [2] or bubble sets [7], but for our use case the focus usually lies on finding and marking single nodes and small groups instead of bigger regions in a graph. We also wanted to visualize the actual numerical values of the data, which is not possible with the aforementioned approaches. Finally three options are left over, and we exemplify them in Figure 2.

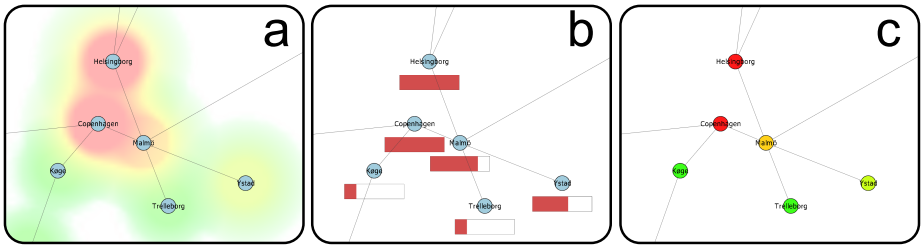


Figure 2: Heat map visualization (a) and two alternative approaches: glyphs (b) and node color (c). They are used to indicate which parts of the entire graph were viewed or changed by other users.

One disadvantage of glyphs in this context is the increased clutter in the graph visualization. Additionally, depending on the size of the glyphs, it could be hard to see the actual data values in highly zoomed-out views of the graph. Changing the color coding of nodes in a graph as alternative is in conflict with our last visualization requirement (cf. V-R 3), because the color coding can be already mapped to another attribute. Thus, heat maps could provide a good alternative to visualize additional data without directly changing the attributes of objects in a node-link diagram. We performed a user experiment (cf. Section 5) to assess how the heat map approach compares against glyphs and node colors. During the experiment, we used a three-color gradient for the heat map visualization. After getting diverse feedback about the preferred color gradient, we decided to give users the option to choose between three different gradient options: the default three-color gradient with colors ranging from green over yellow to red, a monochrome version, and a two-colored gradient for color blind users (cf. Figure 3). We believe that the colored gradients are better at indicating the actual values that are mapped to the heat map, whereas the grayscale

gradient could be used in cases where perceiving the original node colors is more important. The opacity of the heat map can also be adjusted to make it easier to perceive the actual colors of the nodes in case one of the colored heat map versions is used.

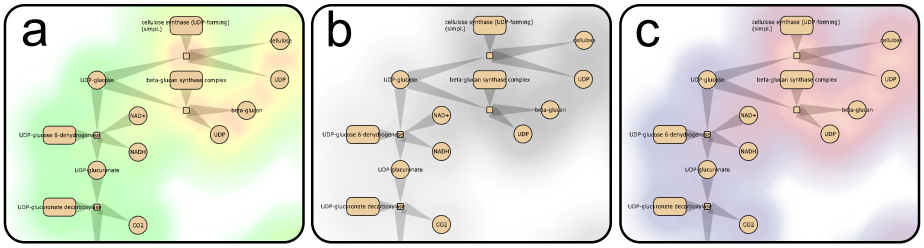


Figure 3: Three different approaches for the visual encoding of the heat map: a three-color gradient (a), a grayscale gradient (b), and a two-color gradient for color blind users (c).

If the heat map visualization is selected, all calculated values are transferred to the requesting client, where they are used to draw an alpha map on an off-screen canvas element. For every node, a circular gradient is drawn which is based on the size and position of the node. This creates an image with grayscale values ranging from 0 to 255. To create the actual colors for the heat map, each pixel value is used to lookup the color from a 1×256 pixel wide color gradient. Based on these values, an OpenGL texture is created and put on a mesh in the background of the graph visualization (cf. Figure 2, (a)). The actual values, which are mapped to the glyphs, node colors or heat map, can be computed based on two different data sources: viewports and graph changes.

Displaying Viewports In the first case, values are calculated based on the amount of seconds that nodes have been in the viewing areas of users (visitation rate). For aggregating this data, OnGraX stores each user's viewport together with the time spent on the position whenever the viewport is changed. Additionally, each time a node is moved, the old position is logged. The server correlates all logged user views and node positions to calculate the values, thus making them robust against changes in the layout of the graph. Figure 4 illustrates this approach. In this small example, three stored viewports of one user and two node movements from another user—whose viewports are ignored here—are taken into account. The user arrived at position A at exactly 10:00 AM, stayed there for 10 seconds, moved his viewport to position B for 5 seconds and finally stayed 16 seconds at position C. In viewport A, node 1 was visible for 10 seconds, but in viewport C, it was only visible for 12 seconds, as the node was only moved into the viewport 4 seconds after the user arrived at the position, resulting in a complete viewing time of 22 seconds for node 1. The viewing time of node 2 is only 2 seconds, as it was moved into viewport B 13 seconds after 10:00 AM, and the user arrived there at 10 seconds after 10:00 AM and left 5 seconds later.

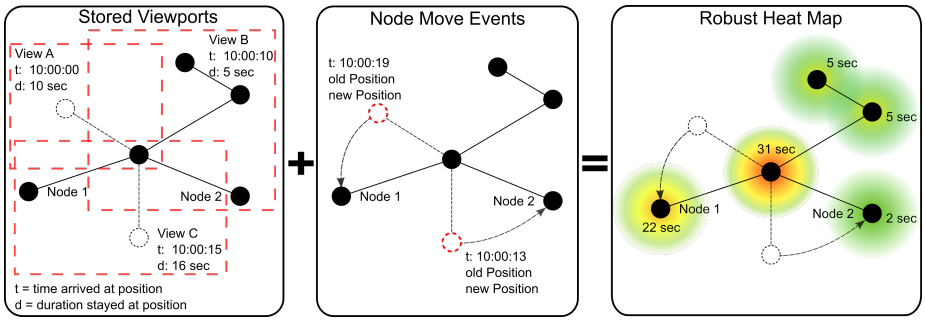


Figure 4: Illustration for the correlation of all stored viewports with all node move actions to create a heat map that is robust against layout changes of the graph.

For zoomed-out views that show a lot of nodes, it is clear that the user does not attend to all nodes in such a view. To solve this issue, users can adjust the settings to filter out these "big views" and only use zoomed-in views to calculate the heat map. Figure 5 illustrates this idea. In (a) and (b), all viewports are used to calculate the values, (c) and (d) only show viewports with a high zoom level. Views are also only tracked if the user is actively working on the graph: if a user switches to another window or tab, then the tracking is stopped. It is also stopped if the mouse is not moved for a while (currently 20 seconds) to avoid tracking views of inactive users. This approach does still include nodes in the views that might not have had attention by an active user, but it gives a better estimate about the viewed graph regions without asking a user to mark every inspected node manually or asking all users to use an eye tracker during the analysis process, for instance.

Displaying Graph Changes In the second case, OnGraX calculates values based on changes that have been performed on nodes. Seven actions (name changed, shape changed, node moved, node added, node selected, edge added, edge removed) are tracked and can be used to calculate the heat map values in this case. A multiplier is specified in a configuration dialog for each individual action type to give it more or less weight during the calculation. This enables analysts to highlight only nodes that were moved and had their names changed, for instance. The visualization can be configured to only show a specific user or to show the data for all users together (the selection of user groups would also be possible and could easily be added to the system). Furthermore, it is possible to select a time frame, for instance, the last five minutes of the current analysis session, or a specific start and end date. This enables an analyst to review changes done in a collaborative session during a specific time frame or to check the work of a single user.

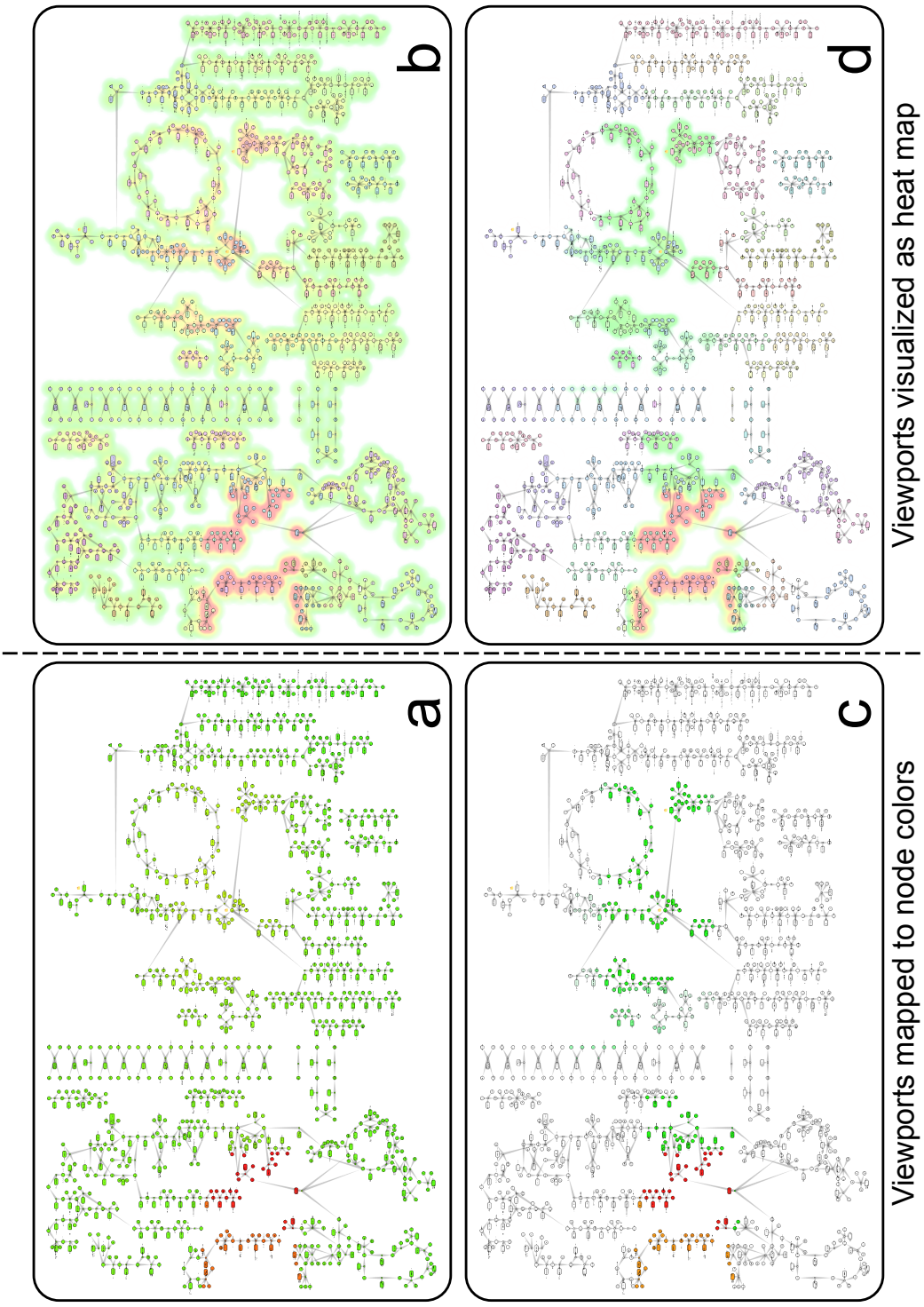


Figure 5: Example of filtering out user viewports with a low zoom level. (a) and (b) show all viewports, whereas (c) and (d) only show zoomed-in viewports of one selected user who was looking at a specific part of the graph. Here, (a) and (c) show the viewports mapped to *node colors*, whereas (b) and (d) show the same data visualized as *heat map* to preserve the original node colors.

4.3 Tracking and Replaying User Actions

Actions performed by other users during a *synchronous* session are shown at the right corner of the screen (cf. Figure 1, (c)) together with the name of the user who initiated the action. A right-click is used to dismiss a recent action and a left-click moves the camera to the location of the action in the graph. Another left-click on the same action moves the camera back to its original position. Thus, users can quickly check what their collaborators are doing and then return to their own work, without having to navigate to every performed action manually. To provide our users with the possibility to keep track of *all* actions that occurred in a session, we use a scrollable timeline at the bottom border of the screen that shows the complete action history of the graph session (cf. Figure 1, (d)). The mouse tooltip for the symbols in the timeline shows the action time and the name of the user who performed the action. The timeline can also be used to revisit old graph states and replay previous actions. If a user clicks on a symbol, all actions performed since this specific action are replayed in reverse order. The visualization will show the graph in a state before the action was performed. Shortly after the graph has been transformed to its old state, the clicked action is reapplied, animating the graph to the requested point in time.

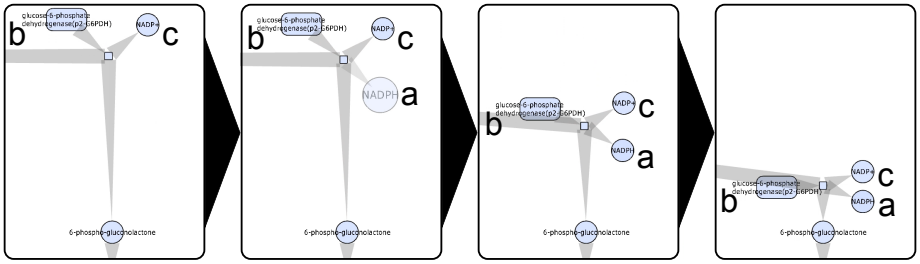


Figure 6: Transition to a previous graph state. The figure shows the *reverse animation* for a node delete action (a) and node move actions for the nodes marked with (b) and (c).

Figure 6 shows an example of such a transition. This feature gives users a tool to revisit old graph states and replay old actions allowing them to assess what work has been done by other collaborators. Clicking on the rightmost symbol reverts the graph back to its present state. While viewing an old graph state, it is *not* possible to apply any changes to the graph. We decided against this feature as it would open the possibility to create numerous new branches of different graph states. This is an interesting aspect and actively researched [33], but currently not the focus of our work.

It is also possible to inspect the complete action history of a graph session as a list in an additional dialog. To not get overwhelmed with uninteresting notifications in this dialog or the timeline, users are able to configure which types of actions are tracked or filtered out.

5 Formative Evaluation

We performed two studies: a user experiment and an expert review to get an impression of the usability of the most important aspects of OnGraX. The user experiment had the goal to evaluate the usefulness and acceptance of our heat map approach to visualize user behavior data in comparison to glyphs and node coloring. Here, the test subjects worked independently of each other, i.e., asynchronously. In contrast, the expert review aimed to assess OnGraX' features for collaboratively revising a graph during synchronous analysis sessions in the domain of metabolic network analysis.

5.1 Heat Map Evaluation

We recruited 15 participants (7 undergraduate students, 7 graduate students, and 1 post-graduate; average age = 28; 5 female, 10 male). Seven participants had a background in computer science and eight a background in media technology. Eight participants never worked with node-link diagrams before, but everyone was familiar with them.

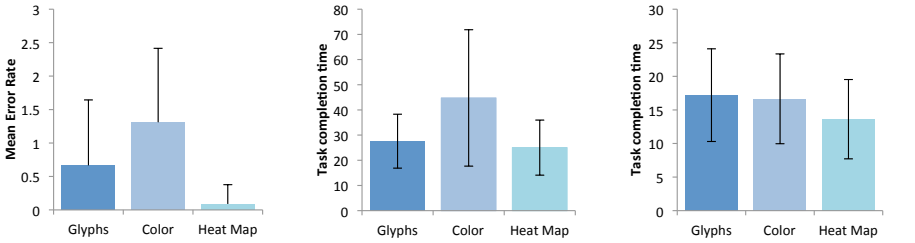
All 15 sessions were recorded on video and the participants were instructed to employ a think-aloud protocol. Before starting the actual tasks, the tool and the three visualization approaches for user behavior data (glyphs, node color, heat map) and their meaning were introduced by the experimenter, and each participant could explore a sample graph to get accustomed to the tool. Each session took about 25-30 minutes, and we asked the participants to solve each task as quickly as possible, but the time for the tasks was not limited by us. All participants had to solve two tasks for nine different graphs with the help of the three visualization approaches. Both tasks were described as follows:

Task 1 – explore graph changes: Find and count all nodes that were moved by a specific user (9-14 single marked nodes per graph).

Task 2 – explore viewports: Find all regions that a specific user was most interested in (1-3 marked regions per graph).

The experiment was conducted following a within-subjects design, and users were divided into three different groups. Every group explored all graphs in the same order but with a different sequence of visualization approaches. Six graphs were generated randomly: the first three graphs consisted of 1,000 nodes/edges and the following three of 2,000 nodes/edges. For the last three graphs, we used existing metabolic networks with 1,300 to 1,800 nodes/edges.

Quantitative Results We started measuring the task time in seconds for each task as soon as the visualization of the user behavioral data was enabled by the participants and stopped the time as soon as they reported a number. For Task 1, we show the number of nodes that were not found by the users (mean error rate). In Task 2, all participants found all marked regions, regardless of the visualization approach. Therefore, we only report the error rate for Task 1.



(a) Task 1, mean error rate

(b) Task 1, mean completion time in seconds

(c) Task 2, mean completion time in seconds

Figure 7: Analysis of the two tasks for the different visualization approaches.

Figure 7 shows the summarized results for all graphs. Initial Friedman tests showed that both tasks had statistically significant differences in task completion time. Task 1: $\chi^2 = 34.881$, $p < 0.001$. Task 2: $\chi^2 = 16.812$, $p < 0.001$. We conducted a post hoc analysis with Wilcoxon signed-rank tests for our not normally distributed data. For Task 1, the median interquartile range (IQR) task completion times were 26 (Glyphs), 39 (Node Colors), and 20 (Heat Map). Both, glyphs vs. heat maps ($Z = -3.678, p < 0.001$) and node colors vs. heat maps ($Z = -5.334, p < 0.001$) had a significant reduction in task completion time. For Task 2, the median (IQR) task completion times were 19 (Glyphs), 15 (Node Colors), and 13 (Heat Map). Here, the heat map approach also performed significantly better in comparison with glyphs ($Z = -3.678, p < 0.001$) and node colors ($Z = -2.406, p = 0.016$).

Qualitative Results We asked all participants which visualization approach they preferred. Everyone favored the heat map visualization. For them, the heat map was the easiest to perceive, and it also provided the most convenient way to find single nodes with high values, even at lower zoom levels. While performing the second task, four participants mentioned that the glyph approach introduced too much clutter in the view, especially for the metabolic networks. They said that glyphs were hard to distinguish from the actual nodes, because both the nodes and glyphs sometimes had a similar shape.

5.2 Expert Review

Expert reviews are an alternative to assess interface usability and find possible problems without having to perform a full scale user study [31]. Goal of this expert review was to get informal feedback about the usability of OnGraX' features for synchronous sessions from a small group of experts, who use the tool to revise metabolic networks collaboratively. More precisely, we wanted to get a first impression if OnGraX could address our three collaboration requirements (cf. C-R 1-3), i.e., (1) if the participants could keep track of each other's view position in the graph, (2) if they could keep a common ground while working

on the graph, and (3) if they could coordinate their work by using the chat and annotation features of OnGraX. The review was performed with an earlier prototype of our tool. Please note that this review has an informal character and is by no means a full scale user study, which would require more detailed preparation (cf. the work of Carpendale on evaluating information visualizations [3]).

Participants and Review Setup We asked three experts from the Bioinformatics research group at Monash University, Australia, to collaboratively work on a network in a *synchronous* session. All three experts were male. One of the experts was a full professor of Bioinformatics (45 years of age), and the two others were research fellows (one postdoctoral researcher, one very experienced research assistant; both around 30 years old) with degrees in Bioinformatics as well. The professor already knew OnGraX, because we asked him to provide some general feedback related to specifics in Bioinformatics (e.g., typical shapes of nodes or use of color) before the review took place. Consequently, he also introduced the task for the review to the two other experts. We asked the experts to revise the graph shown in Figure 1. All experts worked with this graph previously. For this expert review, however, a small part of the network was removed by the experimenters, and the experts had to manually edit and add the missing nodes and edges together. Every expert was sitting alone in his own office and was not supervised by us directly. Before they started, the participants had to watch a six minute introduction video for OnGraX and afterwards opened a web browser on their computers to join the graph session in OnGraX at the same time. The heat map visualization was turned off by default, as we did not aim to assess the heat map features in this review, but the feature was mentioned in the introduction video. We did not specify a time limit for the review, and the experts worked for about one hour on the graph. One experimenter also joined the session remotely to observe the experts during the review. His own viewport (zoomed out completely) was visible to the expert reviewers, and he could answer urgent questions via the online chat.

Results After the experiment, we asked the participants to write down their thoughts about the tool and if the available features helped to fulfill our three collaboration requirements. The first participant found it very easy to follow the work and positions of the other online users with the help of the viewports and tracking possibility of the mouse cursor, while the other two experts thought that the symbol for displaying the center of the other users' viewports together with all mouse cursors was a bit confusing. For them, it would have been enough to only show the colored rectangles and mouse cursors of specific users. Additionally, when clicking on a user icon in the top left corner of the screen (see Figure 1, (c)), they would have preferred to move to the respective user's mouse cursor rather than to the center of his view. We fixed these issues in the current version of OnGraX and also added the possibility to automatically follow another user's mouse and/or camera position in real time. The experts

thought that this would be a useful feature to allow collaborators to just follow one user's work on a graph while they discuss possible changes with each other.

All participants found the tool quite interesting, but they missed some features. In particular, everyone asked for an "undo" function, and they would have liked to be able to hide the timeline at the bottom of the screen. Both features have already been added in the current version of OnGraX. Even though the experts found the chat window helpful, they would have preferred to talk directly via an in-browser voice chat. They said it was easy to follow and track applied changes during the session, but discussing the changes that should actually be performed was quite cumbersome as they had to write down every question in the chat window. Alternatively, collaborators could use tools like Skype or Google Hangouts, but this would require more time during the start of a collaborative session. It would be more convenient if a voice chat functionality could be archived in the browser without having to rely on external programs or additional plugins. This could be addressed in a future version of OnGraX with the help of the new WebRTC standard [38] for build-in real-time communications in browsers.

The feedback which we got from this review indicated that being able to see the viewports and mouse cursors of a selection of users, helps analysts to keep a common ground in a synchronous session (cf. C-R 1). The experts also found the short animations when nodes and edges are added, deleted or modified to be helpful for following ongoing changes of other users (cf. C-R 2). Even though we did not ask them to use the heat map feature, two of the experts actually turned on this functionality and used it to track the nodes down that were added during the course of the review. The experts evaluated the heat map and chat feature as helpful, whereas the functions to replay user actions and add annotations were a bit less important for them. This is likely because they worked in a synchronous session and did not need to replay former actions or read old annotations (cf. C-R 3), since they could use the chat window to communicate with each other directly.

6 Conclusions

In this article, we presented a web-based collaborative system for visualizing graphs with several thousands of nodes and edges, see [42] for a more detailed discussion on performance and scalability. Our tool OnGraX provides visualization and interaction techniques for analyzing data sets synchronously *and* asynchronously in a distributed environment. Additionally, all actions performed during a session as well as the users' camera positions are tracked and can be visualized along with the graph data by using heat map representations.

We propose using heat maps to efficiently show additional data without affecting the original graph visualization. Based on a user experiment, we show that the heat map-based approach compares better against glyphs or changing the background color of nodes. As future work, we plan to evaluate the other aspects of OnGraX—such as those described in Sections 4.1 and 4.3—and to use

the tool in other contexts and application domains. For instance, some collaborators from biology want to use OnGraX for the education of their students. The idea is to give students existing metabolic pathways and ask to revise and edit those graphs. Afterwards, the docents could join the online session and discuss those changes with the students. We will use this opportunity to test our tool in another authentic environment and perform a detailed user study during collaborative work in an educational setting.

In our specific use case, graph changes are usually limited to a couple of nodes, thus the tracking of *all* actions and visualizing this data is not an issue here. This could become problematic if a graph or a subgraph is changed drastically. In this case, additional options to set the granularity for tracked events and alternative visualization techniques would be required including a newly designed evaluation.

Acknowledgements

The authors would like to thank Falk Schreiber and his colleagues at Monash University in Australia for constructive discussions and valuable feedback, as well as all students at Linnaeus University in Sweden for participation in our initial user studies and experiments.

References

[1] M. Albrecht, A. Kerren, K. Klein, O. Kohlbacher, P. Mutzel, W. Paul, F. Schreiber, and M. Wybrow. On Open Problems in Biological Network Visualization. In *Proceeding of the International Symposium on Graph Drawing (GD '09)*, volume 5849 of *LNCS*, pages 256–267. Springer, 2010. doi:10.1007/978-3-642-11805-0_25.

[2] B. E. Alper, N. Henry Riche, and T. Hollerer. Structuring the Space: A Study on Enriching Node-link Diagrams with Visual References. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*, pages 1825–1834, New York, NY, USA, 2014. ACM Press. doi:10.1145/2556288.2557112.

[3] S. Carpendale. Evaluating Information Visualizations. In A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North, editors, *Information Visualization: Human-Centered Issues and Perspectives*, pages 19–45, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. doi:10.1007/978-3-540-70956-5_2.

[4] H. Carr, P. Rheingans, and H. Schumann. Authoring Narrative Visualizations with Ellipsis. *Computer Graphics Forum*, 33(3):361–370, June 2014. doi:10.1111/cgf.12392.

[5] M. C. Chuah and S. F. Roth. Visualizing Common Ground. In *Proceedings of the International Conference on Information Visualization (IV '03)*, pages 365–372. IEEE, 2003. doi:10.1109/IV.2003.1218012.

[6] A. Cockburn, A. Karlson, and B. B. Bederson. A Review of Overview+Detail, Zooming, and Focus+Context Interfaces. *ACM Computing Surveys (CSUR)*, 41(1):2:1–2:31, Jan. 2009. doi:10.1145/1456650.1456652.

[7] C. Collins, G. Penn, and S. Carpendale. Bubble Sets: Revealing Set Relations with Isocontours over Existing Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1009–1016, Nov. 2009. doi:10.1109/TVCG.2009.122.

[8] A. Dieberger, P. Dourish, and K. Höök. Social Navigation: Techniques for Building more Usable Systems. *Interactions*, 7(6), Nov. 2000. doi:10.1145/352580.352587.

[9] C. Gutwin and S. Greenberg. Design for Individuals, Design for Groups: Tradeoffs Between Power and Workspace Awareness. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW '98)*, pages 207–216, New York, NY, USA, 1998. ACM. doi:10.1145/289444.289495.

[10] M. Hascoët and P. Dragicevic. Interactive Graph Matching and Visual Comparison of Graphs and Clustered Graphs. *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '12)*, pages 522–529, 2012. doi:10.1145/2254556.2254654.

[11] J. Heer and M. Agrawala. Design Considerations for Collaborative Visual Analytics. *Information Visualization*, 7(1):49–62, Feb. 2008. doi:10.1109/VAST.2007.4389011.

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

[13] D. Holten and J. J. van Wijk. A User Study on Visualizing Directed Edges in Graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*, pages 2299–2308, New York, NY, USA, 2009. ACM Press. doi:10.1145/1518701.1519054.

[14] J. Hullman, N. Diakopoulos, and E. Adar. Contextifier: Automatic Generation of Annotated Stock Visualizations. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*, pages 2707–2716, 2013. doi:10.1145/2470654.2481374.

[15] P. Isenberg, A. Bezerianos, N. Henry, M. S. J. Carpendale, and J.-D. Fekete. CoCoNutTrix: Collaborative Retrofitting for Information Visualization. *IEEE Computer Graphics and Applications*, 29(5):44–57, Sept. 2009. doi:10.1109/MCG.2009.78.

[16] P. Isenberg, N. Elmqvist, D. Cernea, J. Scholtz, K.-L. Ma, and H. Hagen. Collaborative Visualization: Definition, Challenges, and Research Agenda. *Information Visualization*, 10(4):310–326, 2011. doi:10.1177/1473871611412817.

[17] P. Isenberg and D. Fisher. Collaborative Brushing and Linking for Co-Located Visual Analytics of Document Collections. *Computer Graphics Forum*, 28(3):1031–1038, June 2009. doi:10.1111/j.1467-8659.2009.01444.x.

[18] P. Isenberg, A. Tang, and S. Carpendale. An Exploratory Study of Visual Information Analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*, pages 1217–1226, New York, NY, USA, 2008. ACM Press. doi:10.1145/1357054.1357245.

[19] I. Jusufi, C. Klukas, A. Kerren, and F. Schreiber. Guiding the Interactive Exploration of Metabolic Pathway Interconnections. *Information Visualization*, 11(2):136–150, 2012. doi:10.1177/1473871611405677.

[20] A. Kerren and F. Schreiber. Toward the Role of Interaction in Visual Analytics. In *Proceedings of the Winter Simulation Conference (WSC '12)*,

pages 420:1–420:13. IEEE Computer Society Press, 2012. doi:10.1109/WSC.2012.6465208.

[21] A. Kerren and F. Schreiber. Network Visualization for Integrative Bioinformatics. In M. Chen and R. Hofestädt, editors, *Approaches in Integrative Bioinformatics – Towards the Virtual Cell*, pages 173–202. Springer Heidelberg, 2014. doi:10.1007/978-3-642-41281-3_7.

[22] Khronos Group. WebGL Specification. Editor's Draft 1 July 2014. http://www.khronos.org/registry/webgl/specs/latest. Accessed July 2016.

[23] K. Laboratories. KEGG: Kyoto Encyclopedia of Genes and Genomes. http://www.genome.jp/kegg/. Accessed July 2016.

[24] N. Mahyar and M. Tory. Supporting Communication and Coordination in Collaborative Sensemaking. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1633–1642, Dec. 2014. doi:10.1109/TVCG.2014.2346573.

[25] G. Mark and A. Kobsa. The Effects of Collaboration and System Transparency on CIVE Usage: An Empirical Study and Model. *Presence: Teleoperators and Virtual Environments*, 14(1):60–80, Feb. 2005. doi:10.1162/1054746053890279.

[26] J. Matejka, T. Grossman, and G. Fitzmaurice. Patina: Dynamic Heatmaps for Visualizing Application Usage. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*, pages 3227–3236, New York, NY, USA, 2013. ACM Press. doi:10.1145/2470654.2466442.

[27] M. McKeon. Harnessing the Information Ecosystem with Wiki-Based Visualization Dashboards. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1081–1088, Nov. 2009. doi:10.1109/TVCG.2009.148.

[28] M. Pohl, M. Schmitt, and S. Diehl. Comparing the Readability of Graph Layouts using Eyetracking and Task-Oriented Analysis. In *Computational Aesthetics*, pages 49–56, 2009. doi:10.2312/COMPAESTH/COMPAESTH09/049-056.

[29] Ricardo Cabello. three.js. http://threejs.org. Accessed July 2016.

[30] T. A. Slocum, R. B. Mcmaster, F. C. Kessler, and H. H. Howard. *Thematic Cartography and Geovisualization*. Prentice Hall Series in Geographic Information Science. Prentice Hall, 3rd edition, Apr. 2008.

[31] M. Tory and T. Möller. Evaluating Visualizations: Do Expert Reviews Work? *IEEE Computer Graphics and Applications*, 25(5):8–11, Sept. 2005. doi:10.1109/MCG.2005.102.

[32] A. B. Viégas, M. Wattenberg, F. V. Ham, J. Kriss, and M. Mckeon. Many Eyes: A Site for Visualization at Internet Scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1121–1128, 2007. doi:10.1109/TVCG.2007.70577.

[33] T. von Landesberger, S. Fiebig, S. Bremm, A. Kuijper, and D. W. Fellner. Interaction Taxonomy for Tracking of User Actions in Visual Analytics Applications. In W. Huang, editor, *Handbook of Human Centric Visualization*, pages 653–670. Springer New York, 2014. doi:10.1007/978-1-4614-7485-2_26.

[34] O. Špakov and D. Miniotas. Visualization of Eye Gaze Data Using Heat Maps. *Electronics and Electrical Engineering*, 2(2), 2007.

[35] G. Wallner and S. Kriglstein. PLATO: A Visual Analytics System for Gameplay Data. *Computers & Graphics*, 38:341–356, Feb. 2014. doi:10.1016/j.cag.2013.11.010.

[36] L. Wilkinson and M. Friendly. The History of the Cluster Heat Map. *The American Statistician*, 63(2):179–184, 2009. doi:10.1198/tas.2009.0033.

[37] World Wide Web Consortium. The WebSocket API. http://www.w3.org/TR/websockets/. Accessed July 2016.

[38] World Wide Web Consortium. WebRTC. http://www.w3.org/TR/2015/WD-webrtc-20150210/. Accessed July 2016.

[39] B. Zimmer and A. Kerren. Applying Heat Maps in a Web-Based Collaborative Graph Visualization. In *Poster Abstracts, IEEE Information Visualization (InfoVis '14)*, Paris, France, 2014.

[40] B. Zimmer and A. Kerren. Sensemaking and Provenance in Distributed Collaborative Node-Link Visualizations. In *Abstract Papers, IEEE VIS 2014 Workshop: Provenance for Sensemaking*, Paris, France, 2014.

[41] B. Zimmer and A. Kerren. Displaying User Behavior in the Collaborative Graph Visualization System OnGraX. In E. D. Giacomo and A. Lubiw, editors, *Proceedings of the 23rd International Symposium on Graph Drawing & Network Visualization (GD '15)*, volume 9411 of *LNCS*, pages 247–259. Springer, 2015. doi:10.1007/978-3-319-27261-0_21.

[42] B. Zimmer and A. Kerren. Harnessing WebGL and WebSockets for a Web-Based Collaborative Graph Exploration Tool. In P. Cimiano, F. Frasincar, G.-J. Houben, and D. Schwabe, editors, *Proceedings of the 15th International Conference on Web Engineering (ICWE '15)*, volume 9114 of *LNCS*, pages 583–598. Springer, 2015. doi:10.1007/978-3-319-19890-3_37.