Navigation Recommendations for Exploring Hierarchical Graphs

Stefan Gladisch, Heidrun Schumann, Christian Tominski

Institute for Computer Science, University of Rostock, Germany

Abstract. Navigation is a key interaction when analyzing graphs by means of interactive visualization. Particularly for unknown graphs, the user often faces situations where it is not entirely clear where to go next. For hierarchical graphs, the user may also ponder whether it is useful to look at the data at a higher or lower level of abstraction.
In this paper, we present a novel approach for recommending places in a hierarchical graph that are worth visiting next. A flexible definition of interestingness based on the notion of a degree of interest (DOI) allows us to recommend horizontal navigation in terms of the graph layout and also vertical navigation in terms of the level of abstraction. The actual recommendation is communicated to the user through unobtrusive visual cues that are embedded into the visual representation of the graph. A proof-of-concept implementation has been integrated into an existing graph visualization system.

1 Introduction

When exploring unknown graphs, users need to switch between overview and detail representations and they need to navigate to different parts of the graph. These tasks are typically supported by a zoomable representation of the graph, where the graph is hierarchically structured to provide different levels of abstraction [1]. The user can zoom & pan to visit different parts of the graph, and can expand or collapse nodes to adjust the level of abstraction. There are several existing systems that implement this strategy [2], [3], [4]. A big plus of these systems is that users can freely choose the part of the data they are interested in and the level of abstraction that suits their needs.

However, a problem is that users may be overwhelmed with the seemingly infinite number of possibilities for navigation. According to Spence [5], a key question for the user is: *Where should I go now?* Figure 1 illustrates this problem. Considering a current position arrived at during the exploration, the user does not know where interesting data could be located.

In this sense, navigating in an unknown graph to find interesting data is often a tedious trial-and-error procedure. This prompted us to investigate some kind of navigational guidance to interesting data. The aim of such a guidance is to facilitate the user's navigation decisions (i.e., recommend navigation to interesting targets) and to mitigate the trial-and-error character of navigation (i.e., minimize unconscious navigation through regions with uninteresting data).

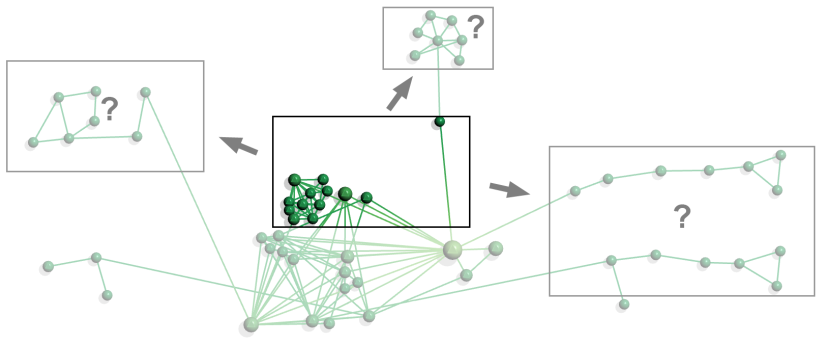

Fig. 1. The problem of navigation. Given a partial view on a graph layout (center rectangle) the user does not know where to navigate in order to find "interesting" data (rectangles with question mark).

In the following, we present a novel data-driven approach for navigation recommendations to support the exploration of hierarchical graphs. In Section 2 we describe in more detail the problem we are dealing with and briefly review existing state-of-the-art solutions. Section 3 introduces our novel approach, including means to define what the user is interested in, to compute navigation recommendations, and to communicate recommendations visually to the user. A demonstration of the proof-of-concept implementation is given in Section 4. Section 5 concludes our work and indicates directions for future work.

2 Problem Description and Related Work

Next we describe the problem addressed by our research, review existing work that is related to this problem, and identify gaps to be filled with our approach.

2.1 Problem Description

We consider hierarchical graphs as input data. A hierarchical graph is defined as a rooted tree whose leaves correspond to a graph at the finest level of granularity. Nodes and edges of the graph may be associated with data attributes. Inner nodes of the tree correspond to aggregations or abstractions of their associated child nodes [1]. We assume that a suitable layout of the graph can be computed with existing methods [6].

To allow users to explore the graph, its layout is visualized as a node-link diagram that is embedded in a zoomable space. The zoomable space enables what we call *horizontal* navigation: The user can pan to any rectangular partial view of the graph layout. A hierarchical graph allows for additional navigation on its hierarchical structure: The user can expand or collapse nodes in order to get to a lower or higher level of abstraction [7]. We call this *vertical* navigation. Figure 2 illustrates both types of navigation.

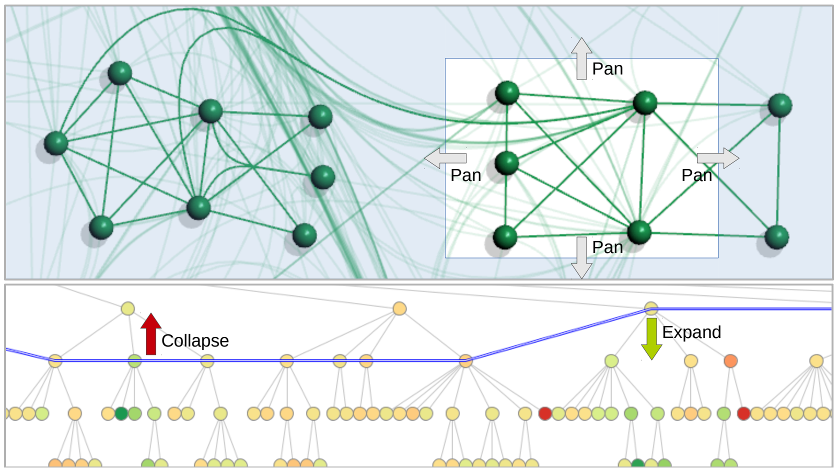

Fig. 2. Navigation in a hierarchical graph. Horizontal navigation means altering the partial view on the graph layout (e.g., by paning the view). Vertical navigation means adjusting the level of abstraction (blue line) along the graph hierarchy by expanding or collapsing individual nodes. (Colors visualize data attributes associated with nodes.)

One can easily imagine that the number of possible navigation steps is quite large. Should I pan this direction or the other to find some high-degree node? Which node should I expand to uncover a clique? Should I collapse these nodes to catch sight of the maximum-value node? In fact, the user can derive some more or less vague answers from the visual representation itself (e.g., navigate to where many edges connect). But we argue that a dedicated support to assist the user during navigation would be a promising addition to the user's analytical toolbox. With this thinking we are not alone, as documented in the next paragraphs.

2.2 Related Work

Looking at the literature one can find two categories of approaches to assist users in navigating graphs. On the one hand, there are approaches that focus on providing *orientation help* to keep users oriented. On the other hand, *navigation recommendation* approaches aim to actually suggest navigation steps to the user. In the following, we briefly review a few important examples.

Orientation help This kind of assistance helps users to orient themselves while navigating through the data. In the context of graph exploration, May et al. [8] present a technique that computes landmarks in the vicinity (context) of the current partial view (focus). The visualization is enhanced with labeled signposts that show directions to the determined landmarks. Jusufi et al. [9] investigate orientation guidance in graphs for which a complete partition is given. The approach is based on special glyphs that provide overviews of the subgraphs connected to a focus node. Plaisant et al. [10] also use special glyphs for user orientation. They enhance a tree visualization with preview icons that summarize the topology of subtrees. From a more general perspective, we can also consider off-screen visualization techniques (e.g., [11], [12], [13]) to be orientation help.

Navigation recommendations The main idea of navigation recommendations is to suggest navigation steps (e.g., a specific target or a direction). Van Ham and Perer [14] describe an exploration model for graphs that includes navigation recommendations. Based on an initial focus, the approach computes and shows the most interesting contexts. Visual hints help users to decide which nodes in the context to expand in order to navigate to the additional information. Crnovrsanin et al. [15] present a technique that recommends interesting nodes based on a set of selected nodes. Interestingness of nodes depends on data attributes, graph topology, and sequences of previous user interaction. Perer and Van Ham [16] introduce *querying and browsing* as a new paradigm for graph exploration. They propose a general model that determines an initial focus and its context on the graph based on a textual query. Special icons within a node-link diagram recommend to the user where to browse the context in order to find interesting information. Additionally, the approach computes and visualizes the shortest path from a focus node to a recommended node in the context.

Open research questions The reviewed examples from the literature demonstrate quite nicely how useful user assistance can be. A detailed look into the mechanisms behind the existing solutions reveals that most of them define a notion of a current *focus* that is associated with a *context*, where focus and context are defined exclusively on the graph structure. However, this implies that navigation recommendation can be given only for entities being connected to the focus in terms of the graph's topology. Interesting but disconnected nodes (e.g., in graphs with disconnected components) cannot be recommended, even if they are located close to the focus in the graph's layout (which is what users see on the display). Our novel solution addresses this limitation by utilizing a broader and more general notion of focus and associated context.

Another aspect common to the reviewed solutions is that they address only horizontal navigation in plain graphs. Hierarchical graphs have not been considered in connection with navigation recommendations so far. Our approach closes this gap by including vertical navigation along the axis of the level of abstraction. In other words, we address both horizontal navigation *and* vertical navigation. The next section will introduce our approach for navigation recommendations for hierarchical graphs.

3 Navigation recommendations for hierarchical graphs

As described earlier, the scenario is that users explore an unknown hierarchical graph by means of a zoomable visualization that shows a layout of the graph and an encoding of associated data attributes. Our goal is to support the user in deciding which navigation step to take next to arrive at interesting data. To this end, we need to address the following key issues:

Determining recommendation candidates Given a hierarchical graph and the current state of the visual exploration process we need to derive a set of recommendation *candidates*.

Selecting interesting recommendations In order to compile a set of navigation *recommendations* we need to rank the candidates according to their *interestingness* and select those that are worth visiting next.

Communicating recommendations visually The selected navigation recommendations need to be *communicated* to the user in an unobtrusive fashion with as little distraction from the actual visualization as possible.

Following this line of thinking, we will next describe in more detail how our approach handles these issues. But first of all, we need to define what the targets for navigation recommendations could be. In general, one could recommend navigation to any entity related to a hierarchical graph, for example, nodes, edges, connected components, cliques, or any other semantically meaningful subset of nodes and edges. For the sake of simplicity, we restrict our considerations to nodes as the targets for navigation recommendations.

3.1 Determining recommendation candidates

As commonly accepted, the starting point for determining candidates is the user's current focus. Based on the focus we define a context, which contains the candidates. The context must include a sufficiently large number of candidates to choose from, and it must be sufficiently small to stay focused and to avoid computations on a huge search space. The size of the context and hence the number of candidates is controlled by means of a distance measure. In summary, we use three components: (1) a set of focus nodes to start with, (2) a sufficiently sized set of context nodes – the candidates, and (3) a distance measure to control the size of the context. Figure 3 illustrates how these components can be realized.

An intuitive and often used definition of these components is based on the graph structure. A set of focus nodes is selected by the user, and the context is defined by the k-neighborhood of the focus nodes. Here k is the parameter to be adjusted to control the size of the context.

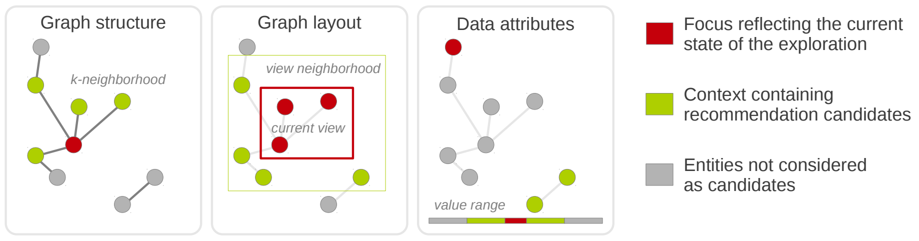
Graph structure
k-neighborhood
Graph layout
view neighborhood
current view
Data attributes
value range
Focus reflecting the current state of the exploration
Context containing recommendation candidates
Entities not considered as candidates

Fig. 3. Different definition of focus and context in terms of the graph structure, the graph layout, and the data attributes yield different candidates for recommendation.

As already indicated, we generalize focus and context to a broader definition. So, as a second facet, we additionally consider the user's current view on the graph layout. That is, all nodes that are currently visible on the display are considered to be the focus. The context is again defined in terms of a neighborhood, but this time a neighborhood in terms of the view space. The size of the context is again controllable.

So far we have not yet taken into account the data attributes that might be associated with the nodes. Consequently, we allow for a focus in attribute space. This focus can be determined in different ways, for example by dynamic filtering sliders or by fixing the value range of what is currently visible on the display. The context can then be defined as a range of values enclosing the focus, where the range's size can be set as needed.

With this general definition we can better capture the different aspects being relevant when exploring graphs – the graph structure, the graph layout, and the data attributes. The broader definition also allows us to circumvent problems that occur when considering either of the aspects alone. An example are nodes that are close to the focus in the layout, but that are far away in terms of the graph structure. In contrast to existing solutions, our approach is able to recommend navigation to such nodes.

3.2 Selecting interesting recommendations

Given our definition of candidates in the context, the next step is to assign an interestingness to each candidate. As we want to recommend interesting nodes to navigate to, we need a concept that describes how interesting a recommendation candidate is. Given the unpredictability of the visual exploration process, the concept must be capable of handling varying interestingness.

An established and widely-applied concept is the *degree of interest* (DOI), a numerical interestingness computed by means of a DOI function. Originally, Furnas [17] introduced the DOI for trees only. Van Ham and Perer [14] generalized it for graphs. Their weighted DOI function considers an a priori interest of the nodes (API), a distance to a focus ($DIST$) and a user interest (UI):

$$DOI(x, F, \mathfrak{s}) = \alpha \cdot API(x) + \beta \cdot UI(x, \mathfrak{s}) + \gamma \cdot DIST(x, F)$$

where x is the node to be assigned an interestingness, F is the current focus, $\mathfrak{s}$ are search criteria that describe what the user is currently interested in, and α, β, γ are real-valued weights. With this DOI definition, we already have a quite flexible mechanism to incorporate the user's interest into the recommendation computation.

Additionally, it can be important to know which data elements have already been visited (i.e., have been visible or have explicitly been marked as explored) in the course of the exploration. We propose to use an additional weighted $KNOW$ component for the DOI function that considers the interestingness of a node according to its exploration state:

$$DOI(x, F, \mathfrak{s}) = \alpha \cdot API(x) + \beta \cdot UI(x, \mathfrak{s}) + \gamma \cdot DIST(x, F) + \delta \cdot KNOW(x)$$

By considering the exploration state of a node, we can penalize already explored data or, on the contrary, favor them. Which option to use depends on the user's goal. Visited nodes can be considered less interesting because they do not provide any new information. On the other hand, they could be particularly of interest for comparison tasks.

Given our specification of the *DOI* function, the question that remains to be answered is how to instantiate it and its components. We follow the accepted way of previous *DOI*-related approaches and provide interactive means for the user to specify and adjust the settings. To ease the specification procedure, we use template functions that can be parameterized using classic GUI elements. Which template functions to apply (i.e., what is interesting?) and how to parameterize them depends on the application domain, the use case, and the analyzed graph.

Given an appropriate *DOI* specification, we compute the interestingness of the nodes of a graph. It is worth mentioning that we do so only for the recommendation candidates in the context of the current focus. This spares us computing interestingness values for all nodes of the whole dataset.

For a hierarchical graph, we differentiate between two alternative ways of computing the interestingness. The first is that we compute interestingness for the finest level of granularity and aggregate interestingness along the hierarchy. The second alternative is to compute interestingness explicitly for each candidate irrespective of whether it is a leave node or an inner node. Again, the application scenario and the nature of the data decide on which alternative to apply.

Now that every candidate has a *DOI* value they can be sorted according to their interestingness. The result is a ranking of the recommendation candidates. Since we only want to recommend the *most interesting* nodes, we choose the first m nodes of the ranking as targets for the navigation recommendations, where m should be kept small to avoid overloading the user with too many recommendations. From our experience with test datasets, we suggest recommending $m < 10$ interesting navigation targets.

3.3 Communicating recommendations visually

The last step is to create an adequate visualization for the navigation recommendations. Given a potentially already visually rich graph visualization, how can we enhance it in order to communicate navigation recommendations to the users without interfering too much with the ongoing visual exploration? Our answer to this question is to embed specifically designed visual navigation cues into the existing node-link visualization. Depending on the type of navigation and on where the target of a recommendation is located, we use different visual cues. The type of navigation can be either *horizontal* or *vertical*.

Recommendation for horizontal navigation For horizontal navigation, we distinguish navigation to nodes that are *on-screen* and nodes that are *off-screen*. Recommendations to on-screen nodes are visualized via subtle highlighting rings that encode how interesting a node is according to its *DOI* value.

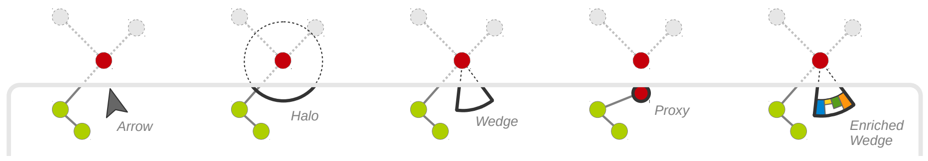

Fig. 4. Techniques for recommending navigation to off-screen nodes. Green nodes are on-screen, dashed elements are off-screen, and the red node is the recommend target.

For recommendations to off-screen nodes, we need a visual encoding that communicates at least the target's direction and better still, the distance to the target as well. For this purpose, we consider known techniques for off-screen visualization, such as arrows, *halos* [11],*wedges* [12], or *proxies* [13]. Arrows are easy to interpret, but communicate navigation direction only. Halos and wedges have the advantage that they encode direction and distance to a recommended navigation target. Further, wedges can be arranged to reduced overlap [12]. Proxies focus not so much on target distance, but more on communicating additional information about the target by means of shape, color, or labels.

Inspired by these approaches, we designed a new solution that combines the advantages of the existing ones. What we call *enriched wedge* is a visual cue that encodes direction and distance, and also additional information about *why* the recommendation was given. This is accomplished by embedding a bar chart into a wedge. The wedge visualizes direction and distance, and each bar visualizes the partial interestingness of a recommended node according to the individual components of the *DOI* function (i.e., *API*, *UI*, *DIST*, *KNOW*). Using enriched wedges can positively influence the user in arriving at a navigation decision (i.e., choosing the "right" navigation target). Figure 4 illustrates the enriched wedge in comparison to existing off-screen techniques.

Of course, the idea of enriching the navigation recommendations with a visualization of individual *DOI* values is not restricted to wedges pointing to off-screen targets. The highlighting of on-screen targets can be enhanced in a similar manner to provide information about interestingness at a glance.

Recommendation for vertical navigation A vertical navigation is necessary when the recommended target is not contained in the currently visualized level of abstraction. As the target is definitely not visible, we need to pick a suitable anchor to attach the navigation recommendation to. We decided to visually highlight the nodes whose expansion (or collapse) would bring the recommended target to the display. For example, if a target is below the current level of abstraction, we highlight the target's ancestor that is contained in the current level of abstraction and whose expansion will uncover the target. If the ancestor is off-screen we can again apply one of the off-screen techniques described before.

In order to differentiate the highlighting for vertical navigation from that for horizontal navigation, and further the one for node expansion from that of node collapse, we resort to animated rings around nodes. In accordance with our goal

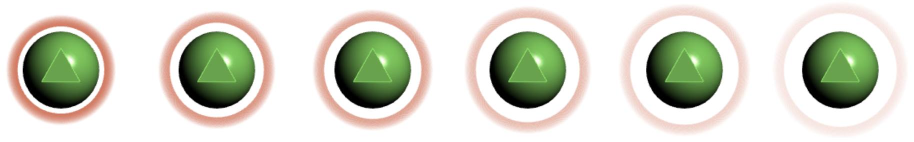

Fig. 5. Snapshots of the animation that indicates nodes to be expanded to arrive at a recommended navigation target.

to generate an unobtrusive visual embedding, the highlighting is designed as subtly pulsing animations with a specific direction. The animated rings appear to shrink when collapse navigation is recommended and to grow for recommended expansion. Figure 5 shows snapshots of an animation indicating an expand recommendation.

3.4 Summary and additional concerns

With the aforementioned mechanisms, we can select interesting nodes and recommend them visually to the user as potentially worthy steps for navigation. A key issue of our approach is balancing it appropriately. Interests vary and also the visual presence of navigation cues will be perceived differently by different users in different stages of the exploration process. Therefore, it is critical to adjust the computation of recommendations and their visualization to the application scenario and to the preferences of the user. Our approach provides the required flexibility to do so. Further, we should recall the on-demand character of our approach. That is, only if users feel that they need assistance they will activate the navigation recommendations.

Two additional concerns need to be addressed in the context of navigational guidance: (1) a good initial view to start with and (2) a visual encoding of the exploration state. Both are not trivial question and we have not dealt with them in depth. Yet we give some ideas how to address them.

Ideally, a good initial view on the data provides an expressive overview of the data and offers a suitable number of options for further exploration. For determining such an initial view, different criteria can matter. For example, the number of nodes can be considered. Huang et al. [18] state that 20 to 100 nodes are suitable for an overview. Moreover, in specific applications, there may exist data elements being semantically more relevant than others. In such cases, including graph elements of higher relevance (e.g., outliers) can lead to a more appropriate initial view. One could also favor nodes with high degree as they potentially lead to more options for navigation along the graph structure. Despite these initial suggestions, creating a good initial view remains a difficult and largely context-dependent task.

The second concern regards the dependency of interestingness on the exploration state. To make this dependency clear to the user it makes sense to visualize a node's exploration state as well, because it may influence navigation decisions. When exploring hierarchical graphs the user might want to know which subtrees have already been explored. Cramming this additional information into the visualization as well is difficult. Therefore, we experimented with on-demand labeling that classifies nodes into *unexplored*, *partially explored*, and *explored*. Such on-demand labels can help users to decide where to explore further and where no further exploration is necessary.

4 Proof-of-Concept Implementation

To test our approach, we developed a proof-of-concept implementation. As the underlying zoomable graph visualization, we use the *CGV* system [4]. We implemented a plausible default preset for the interestingness specification (including maximum attribute values and attribute outliers), which enables us to give recommendations at all times, even in cases where the user has not yet made the interests known to the system. The *DOI* function and its components can be altered interactively via a simple graphical user interface. We implemented arrow-based recommendation cues and our *enriched wedge*.

We tested the system with several hierarchical graphs. Here we illustrate its application with a graph that contains search queries as nodes and relations between the queries as edges. The graph is of moderate size with 695 nodes and 4073 edges. Figure 6 shows a partial view on the graph as the user may see it during exploration. Note that for the purpose of demonstration we use a visual encoding that might not appear as gentle and subtle as one would use it in a real application. In the figure, we can see recommendations to investigate on-screen targets, indicated by red circles around some nodes. Enriched wedges at the border of the screen recommend navigation to off-screen targets. Once the user has decided on the next navigation target, it can be visited by following the navigation recommendations manually. For example, the user can pan the view in the direction of an enriched wedge until the target falls into view. The target is then highlighted using the red circle for on-screen targets. As an alternative to

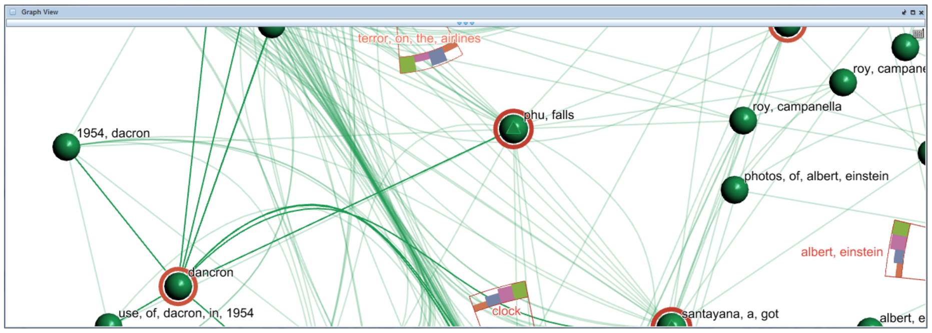

Fig. 6. Navigation recommendation in the proof-of-concept implementation. Red circles indicate on-screen targets worth investigating next. Enriched wedges indicate off-screen targets that might be of interest to the user as well.

manual navigation, we utilize *CGV's* animation facilities to provide automatic animated traveling to the selected target. To this end, the user simply clicks an enriched wedge to trigger the animation.

During the exploration, the recommendations are constantly updated according to the current focus and the specification of the user's interest. The system also keeps track of which nodes have already been visited, where we rely on users explicitly marking a node as explored.

In summary, provided that users are able to express their current interest, our implementation can help in locating interesting nodes in the local context quickly.

5 Conclusion

In this work, we developed a data-driven approach for navigation recommendations to interesting information. Our solution is based on three basic steps: (1) collection of a set of recommendation candidates based on a compound focus, (2) selection of navigation recommendation based on user interests, and (3) visualization of navigation recommendations via visual cues embedded into an existing graph visualization.

Our solution extends the body of existing work in several aspects. We addressed hierarchical graphs, which require both horizontal and vertical navigation, an issue not studied in previous work. In terms of computing navigation recommendations, we generalized the notion of focus and context to incorporate the graph structure, the graph layout, and the data attributes. Further, we extended the widely-accepted *DOI* concept by the component *KNOW*, which captures the exploration state of data elements. For communicating navigation recommendations, we suggest several visual encodings, including the novel *enriched wedge*. A proof-of-concept implementation has been developed.

The mechanisms behind our concept work quite well in the proof-of-concept implementation. However, in the future, we need to develop a better interface for specifying the users' interests. Our current approach with classic GUI elements needs to be revised in order to make the overall solution more accessible for users. Further it makes sense to keep a history of what users have already marked as interesting. This would allow us to find better starting points for exploration.

A pressing issue is that the degree to which our solution reduces the trial-and-error character of visual exploration has not yet been quantified. And unfortunately we believe that it will be hard to do so due to the many influencing factors. Therefore, we invite evaluation experts to contact us and we will gladly collaborate and provide our implementation for in depth usability studies.

Acknowledgements

This research has been supported by the German Research Foundation (DFG) in the context of the project GEMS – graph exploration and manipulation on interactive surfaces.

References

1. Herman, I., Melançon, G., Marshall, M.S.: Graph Visualization and Navigation in Information Visualization: a Survey. IEEE Transactions on Visualization and Computer Graphics **6** (2000) 24–43
2. Auber, D., Archambault, D., Bourqui, R., Lambert, A., Mathiaut, M., Mary, P., Delest, M., Dubois, J., Melançon, G.: The Tulip 3 Framework: A Scalable Software Library for Information Visualization Applications Based on Relational Data. Research Report RR-7860, INRIA (2012)
3. Mathieu, B., Heymann, S., Jacomy, M.: Gephi: An Open Source Software for Exploring and Manipulating Networks. In: Proceedings of the International Conference on Weblogs and Social Media (ICWSM), Association for the Advancement of Artificial Intelligence (2009) 361–362
4. Tominski, C., Abello, J., Schumann, H.: CGV – An Interactive Graph Visualization System. Computers & Graphics **33** (2009) 660–678
5. Spence, R.: Information Visualization: Design for Interaction. 2nd edn. Pearson/Prentice Hall (2007)
6. Abello, J., van Ham, F., Krishnan, N.: ASK-GraphView: A Large Scale Graph Visualization System. IEEE Transactions on Visualization and Computer Graphics **12** (2006) 669–676
7. Elmqvist, N., Fekete, J.D.: Hierarchical Aggregation for Information Visualization: Overview, Techniques, and Design Guidelines. IEEE Transactions on Visualization and Computer Graphics **16** (2010) 439–454
8. May, T., Steiger, M., Kohlhammer, J.D.J.: Using Signposts for Navigation in Large Graphs. Computer Graphics Forum **31** (2012) 985–994
9. Jusufi, I., Klukas, C., Kerren, A., Schreiber, F.: Guiding the Interactive Exploration of Metabolic Pathway Interconnections. Information Visualization **11** (2012) 136–150
10. Plaisant, C., Grosjean, J., Bederson, B.: SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation. In: Proceedings of the IEEE Symposium on Information Visualization (InfoVis), IEEE Computer Society (2002) 57–64
11. Baudisch, P., Rosenholtz, R.: Halo: A Technique for Visualizing Off-Screen Objects. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI), ACM Press (2003) 481–488
12. Gustafson, S., Baudisch, P., Gutwin, C., Irani, P.: Wedge: Clutter-Free Visualization of Off-Screen Locations. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI), ACM Press (2008) 787–796
13. Frisch, M., Dachselt, R.: Visualizing offscreen elements of node-link diagrams. Information Visualization **12** (2013) 133–162
14. van Ham, F., Perer, A.: Search, Show Context, Expand on Demand: Supporting Large Graph Exploration with Degree-of-Interest. IEEE Transactions on Visualization and Computer Graphics **15** (2009) 953–960
15. Crnovrsanin, T., Liao, I., Wu, Y., Ma, K.L.: Visual Recommendations for Network Navigation. Computer Graphics Forum **30** (2011) 1081–1090
16. Perer, A., van Ham, F.: Integrating Querying and Browsing in Partial Graph Visualizations. Technical report, IBM Research (2011)
17. Furnas, G.W.: Generalized Fisheye Views. ACM SIGCHI Bulletin **17** (1986) 16–23
18. Huang, M.L., Eades, P., Wang, J.: On-line Animated Visualization of Huge Graphs Using a Modified Spring Algorithm. Journal of Visual Languages and Computing **9** (1998) 623–645