Parallel Distances
Analyzing Multi-Level Relationships in Networks

Stephan Pajer
Supervised by: Harald Piringer

VRVis Research Center
Vienna / Austria

Abstract

This paper introduces a new type of visualization that supports the study of queries concerning relationships between different groups of nodes in a network. It allows a user performing a search for a multi-leveled relationship within a graph to see how each part of the query affects the resulting set of nodes. This view can be efficiently utilized in a linked-view environment to supplement well-established visualizations. The paper finishes off with a case study that exemplifies how to apply the parallel distance view to perform a query on a dataset.

Keywords: Information Visualization, Graph Visualization, Linked Views

1 Introduction

With the recent boom in the popularity of social networks, the importance of extracting information from those networks has undergone a similarly staggering increase. These networks differ from normal network in the distribution of their connections. The maximum distance between nodes is quite small [Wat04] even though most nodes only have a small number of connections [JHGH08]. Additionally, these networks are generally multivariate, which means that there is a multitude of data available for each node of the networks, e.g. age, income, location, etc.

When investigating a social network, one common task users want to accomplish is to search the network for a specific relation between different nodes, each characterized by certain attributes. The nodes can then be grouped together by such attributes as age or income. The user can then create a query to find out how many persons of a group know someone from another group. Such a query might search for persons with a low income that directly know politicians. The query could also be expanded to include more than just two groups. This paper presents 'Parallel Distances', a view that is tailored to assist the user in studying the effects of each part of such a complex relation between groups of nodes in a network.

After briefly presenting related work, this paper will explain this new visualization. We will then go on to explain the interactions we have implemented for this view before describing the necessary computations. The paper continues by giving a usage example of our new visualization before finishing with a discussion and propositions for future work that could be done to improve this view.

2 Related Work

The most widely known visualization of a network is the node-link view. To cope with the size of some networks, several systems that cluster multiple nodes together have been proposed, e.g. [AvHK06] and [AMA08]. This way, node-link views have been successfully utilized in analyzing social networks in [ACJM03] and [PS06]. Others have decided to combine matrix representations [FK46] with node-link diagrams [HF07], [HFM07]. These two visualizations have also been used together in a multi-view environment in [HF06].

Some approaches that also use a node-link diagram to visualize the network but bin the nodes in groups [BMGK08] and [SA06]. Such a binning of multivariate data has also been utilized several times in the recent years to allow the inspection of connections between groups, e.g. [PW06], [PvW08] and [Wat06].

In addition to more traditional network visualization approaches, the Parallel Distance view is also heavily based on Parallel Sets [BKH05] which visualize connections between sets.

3 Parallel Distances

3.1 Visualization

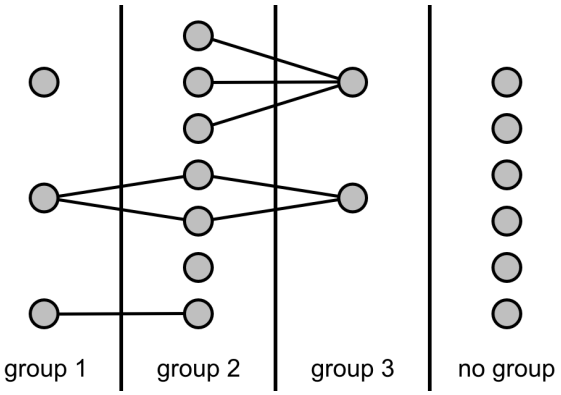


Figure 1: The network visualized in this section.

It should be noted that all figures in this subsection are created from the network pictured in Figure 1. Even though nodes can be members of multiple groups, this example only uses disjunct groups. While all figures are explained in the text, the reader might find it helpful to compare the resulting visualization to the network they are made of.

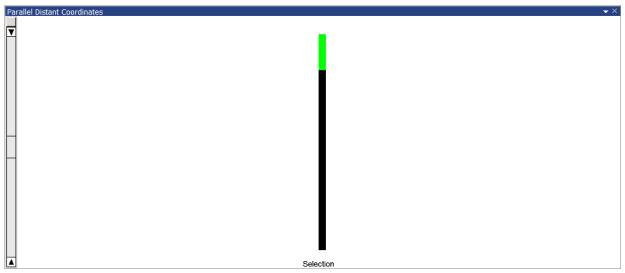


Figure 2: A single axis

First, we will explain how the view looks when just a single group of nodes has been added to the view. An example image of this can be found in Figure 2. A single axis representing the group can be seen. This axis is divided into two parts. The upper part, displayed in green, represents the nodes that are in the group. The lower black part represents the nodes that are not selected. Both parts are scaled according to the number of nodes they represent.

Now we will explain what happens when a second group is added to the view. Figure 3 displays the look of the view with two axes. Two axes is the minimum size for a query to be performed on the groups. Each axis is now divided into 3 parts. While the black part at the bottom has remained unchanged, the top green part has now been divided into two. The red part at the very top shows how many nodes in the group fulfill all requirements of the query. The remaining green part in the middle represents the nodes that are selected in the group but do not meet all requirements of the query.

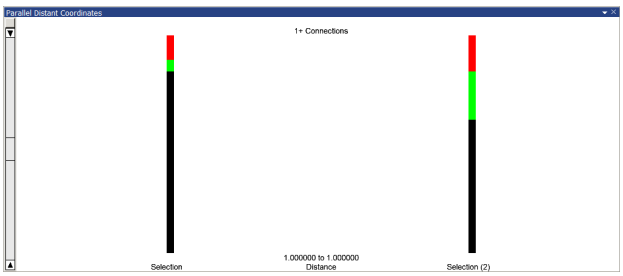
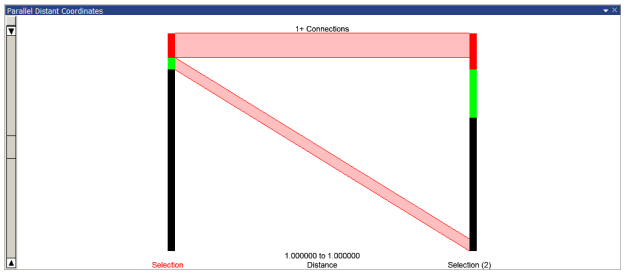


Figure 3: Two axes

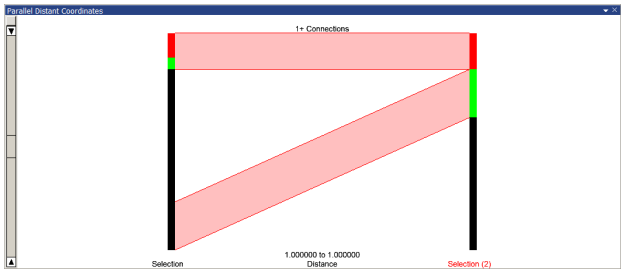
The query is performed from left to right, with a textual representation of the parameters displayed between the axes. Between the axes on the top of the view we display how many connections are required for a node on the left. A node from the group of the left axis that is connected to *at least* this many nodes of the right axis' group is a valid result of this query. Below the number of required connections, on the bottom of the view, we display how large the distance between a node from the left group and a node from the right group has to be. This can be very useful for networks in which each node has a specific weight that represents a value like physical distance between nodes. If, for example, each node represents a city, this value can represent the distance a person has to travel from one city to reach the other one. This way, only cities within a certain distance would be considered for the query.

If the user has selected a node as described in subsection 3.2, the space between the axes is used to show how the connection requirement filters out the nodes that do not fulfill these requirements. See Figure 4 for reference. This is in accordance to the visual information seeking mantra of only offering details on demand [Shn96]. Between the axes are now two connecting segments that represent the nodes in the selected group. Next to the selected axis, both connections are at the top. On the opposite side, one segment stays at the top while another segment points toward the bottom. The segment that stays at the top represents the nodes that fulfill the requirements of the query while the segment that points toward the bottom represents those that do not fulfill the query. Thus, the height of each segment is equal to the respective height on the axis.

Figure 5 shows how the view looks after a third group has been added. To fulfill the query, a node from the left group now has to be connected to the specified amount

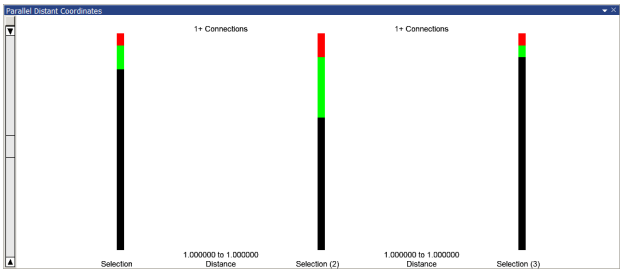


(a) Left axis selected

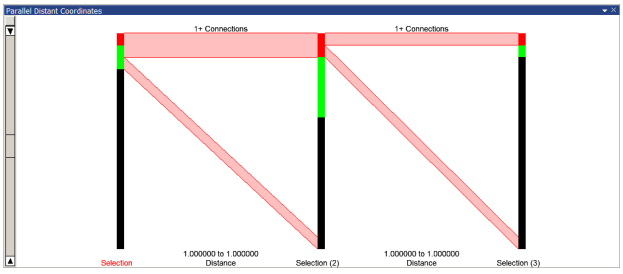


(b) Right axis selected

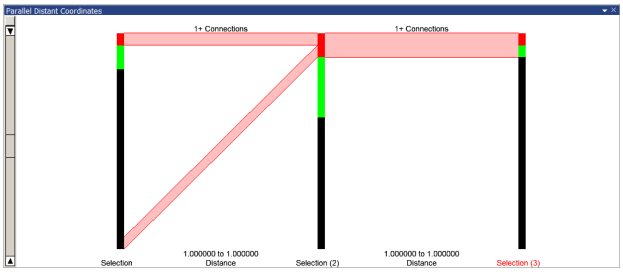
Figure 4: Two axes with one selected



(a) No axis selected



(b) Leftmost axis selected



(c) Rightmost axis selected

Figure 5: Three axes

of nodes from the middle group that in turn have to be connected to the specified amount of nodes from the right group. Let us now assume that the user has selected either the left or the right axis. The connecting segments adjacent to the selected axis act as if only the selected axis and the middle axis existed. The connecting segments between the other two axes show the number of nodes that fulfill the whole query staying at the top with the nodes that fulfill the first part of the query but fail the second part point toward the bottom. The size of the segment pointing toward the bottom thus equals the difference between the two segments staying at the top. This allows the user to see which part of the selected group fails the requirements at each part of the query.

The connections look somewhat different if the user selects the group in the middle. This change is shown in Figure 6. Each connecting segment has been further split into two. The red segment at the very top represents the amount of nodes that fulfill the complete query. The black segment pointing to the top represents nodes that fulfill this connections requirements but fail the requirements between the middle axis and the opposite axis. The black segment pointing toward the bottom stands for nodes that fulfill neither this requirement nor the one from the other side. The second red segment, between the first red segment and the black segment pointing to the top, that points toward the bottom represents the nodes that fail the requirements between these axes but fulfill the requirements between the other two axes. Thus the red segments represent the nodes that fulfill the requirements of the *other* direction while the black segments fail the other direction.

3.2 Interaction

Since this system is designed to assist the user in exploring the dataset, interaction is very important. First, we will explain how a user can interact with our view, then explain the offered interactions of our view with a linked-view system.

The user may select an axis by clicking on or in the close proximity of it. Selecting an axis updates all connecting segments to reflect the amount of filtered and non-filtered nodes of the newly selected group as described in subsection 3.1.

Besides the main window, an additional control window allows the user to adjust the currently set groups (shown in Figure 7) and also allows the selection of an axis. It also enables to change the order of the axes as well to remove an axis. A dialog to adjust the constraints from the currently selected axis to the next one can also be opened from this control window.

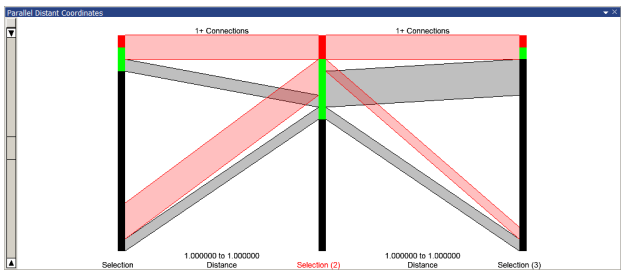
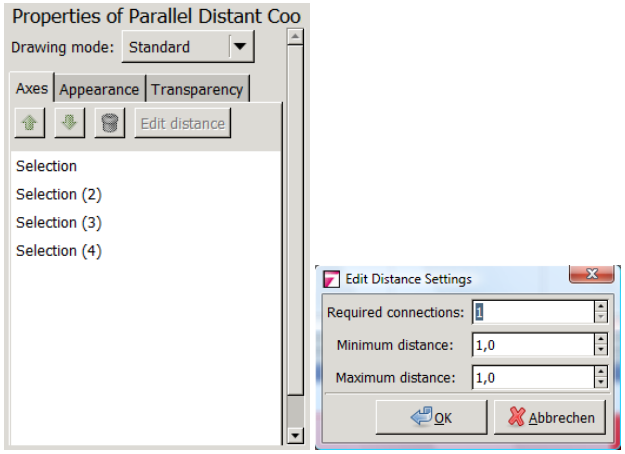


Figure 6: Three axes with the middle axis selected

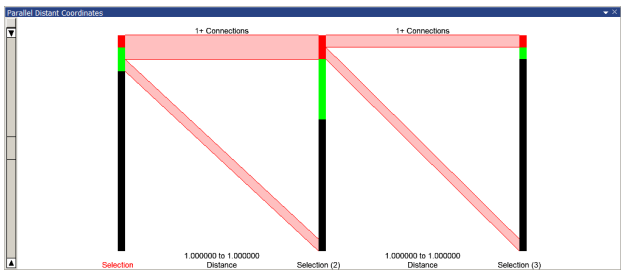


(a) Control window (b) Connection settings

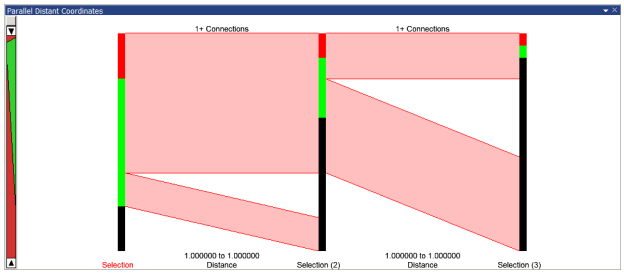
Figure 7: Additional windows and dialogs.

Our view allows the user to zoom into any region he is interested in. Zooming is depicted in Figure 8. This is implemented as a bar to the far left of the window. To zoom, the user simply has to click the widget and drag his mouse. Zoom also affects the displayed connections that originate from this axis. Such a zoom operation is not reset after the user selects a new axis.

There is also interaction between this view and the linked-view system. The user can select any subset of nodes (passes the query, does not pass but is in group, not in group) from any axis. Such a selection affects the whole system and can be utilized in almost any view. Selection also enables the user to transfer the results of a query from this view to other linked views. It is performed by pressing the left mouse button over an axis and highlights the subset of graph nodes that is visually represented below the cursor in all linked views. The framework this view was implemented in supports composite queries based on set operations. This way, selecting the results of the query from all groups is as simple as choosing the unison operation and selecting the desired subset of each distance separately.



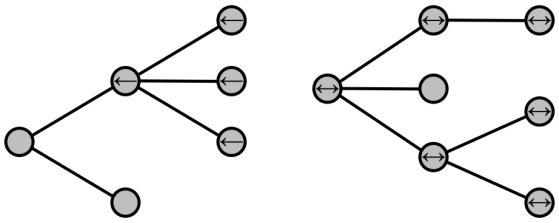
(a)



(b)

Figure 8: The same visualization with (b) and without zoom (a).

4 Involved Query Computations



(a) (b)

Figure 9: Example markings of the algorithm with 2+ connections to the left and 1+ connections to the right.

This section explains the algorithm required to calculate the displayed values of our approach. For this purpose, this algorithm solves the presented query. In addition to the main query, it is also necessary to calculate reduced queries to display the connecting segments for longer (more than 2 axes) queries to display the fraction of nodes that fail the requirements at each axis. These queries differ from the main query in the fact that they ignore axes from one or both sides. For each combination of groups removed from the left and/or right that still has at least two groups left, the algorithm has to run once to solve it. To obtain the number of nodes of a group that fails at a particular distance, the results from the queries with adjacent cutoffs are compared. These results are the same results as the ones that would be obtained from the queries just before and after adding the group. Thus, they

Listing 1 Query Calculation

```
//backward pass
mark all nodes of rightmost axis that are in the group as backward selected
for each segment between axes starting from right
   for each selected node in the left axis
      evaluate connection with backward selected nodes of the right axis
      if the requirements are fulfilled
         mark backward selected

//forward pass
mark all nodes of leftmost axis that are backward selected as forward selected
for each segment between axes starting from left
   for each forward selected node in the left axis
      for each connected and backward selected node of the right axis
         mark forward selected
```

represent the difference in nodes that pass the query this group causes.

The developed algorithm works by performing two passes over every segment between axes. Pseudocode for the algorithm is presented in Listing 1 with an additional illustration showing an example marking in Figure 9. The first pass is a backward pass marking valid nodes starting on the rightmost axis. The second pass starts at the leftmost axis and marks the nodes that pass all requirements and thus fulfill the complete query. For calculations that do not involve all groups in the view, the backward pass information only has to be calculated once per first group on the right and can be reused otherwise.

In the following evaluation, V stands for the number of nodes while S stands for the number of segments between groups (the number of groups minus one). The algorithm has a complexity of $O(V^2)$ for each segment, resulting in a complexity of $O(SV^2)$ for the query and a complexity of $O(S^2V^2)$ for all calculations that have to be performed. In the backward pass, for each segment each selected node on the left side has to be checked against all selected nodes on the right side, resulting in the complexity of V^2. Since S is generally quite small, the major factor influencing the experienced performance of our approach is the size of the groups.

5 Usage Example

This section demonstrates how Parallel Distances can be utilized to analyze a dataset. We utilize our view to perform the search for structure A from the 2nd mini-challenge of the VAST challenge 2009[1]. This challenge asks the user to search the given dataset for an employee at an embassy who communicates with exactly three 'handlers' who pass on his information to a common middle man who then forwards it to the mastermind of a criminal organization. It is further known that the employee has about 40 contacts, each handler has 30–40 contacts, the middle man has 4–5 contacts and the mastermind 'Fearless Leader' has more then 100 contacts including international ones. The handlers are also not allowed to communicate with each other.

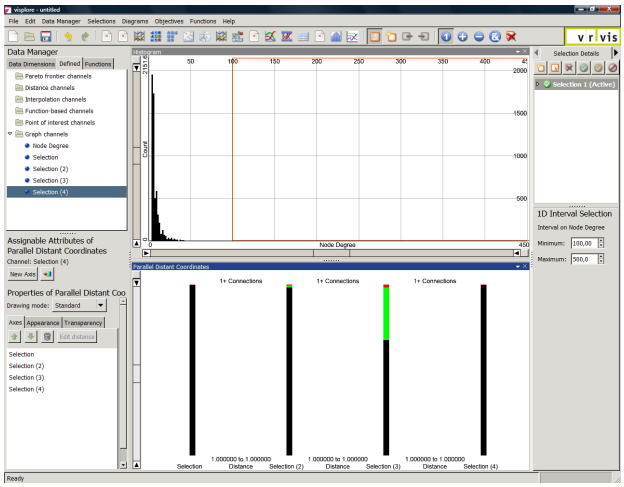


Figure 10: Groups created and added to view.

After loading the data into the system, we start off by defining the groups. We define the employee group as users with 38–42 contacts to incorporate the expressed uncertainty. The handler group is defined as users with 30–40 contacts. It can be seen that these groups overlap. We define the middle man group as persons with 4–5 contacts and add all users with at least 100 contacts to the mastermind group.

[1] http://hcil.cs.umd.edu/localphp/hcil/vast/index.php/taskdesc/index/mini_challenge_2

Next, we open up an instance of our view and add each of those groups in the order employee–handler–middle man–mastermind. A picture of the system at this stage can be found in Figure 10. Now we set the parameters of the connections between the groups. Since a direct connection is the default setting and just the employee needs more than one connected node, we simply set the number of connections between the first two axes (employees and handlers) to three.

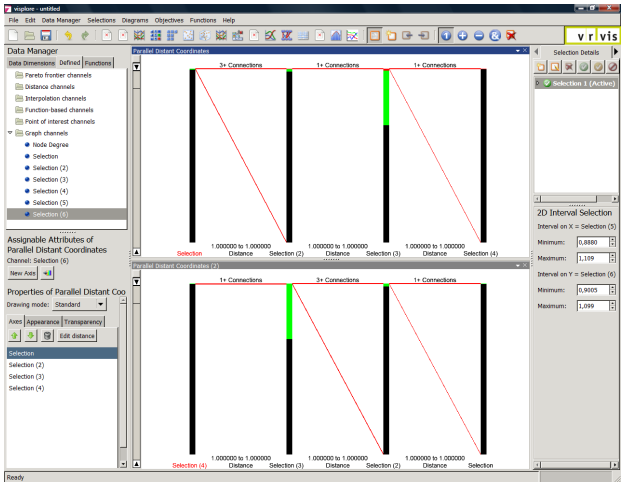


Figure 11: Views and settings needed to solve the problem.

The only requirement we have not yet incorporated is that all three handlers need to be connected to a single middle man. To specify this requirement, we add a second instance of our view to the system. This instance also contains all groups, but in reverse order: mastermind—middle man—handler—employee. We set the required number of required connections between the second and third axis (middle man and handler) to three. This is shown in Figure 11.

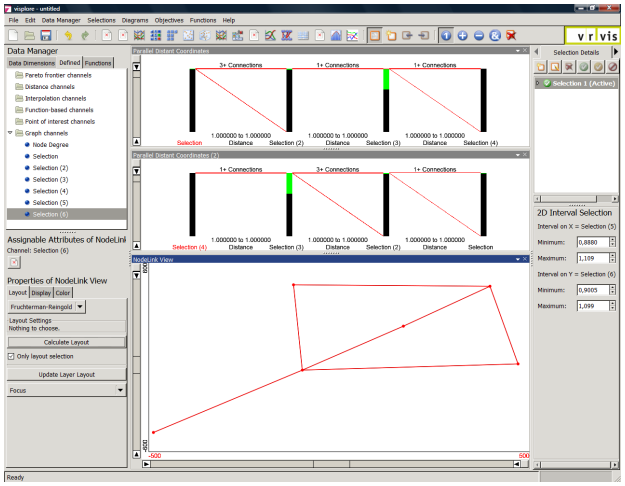


Figure 12: Visualization of the result.

An intersection between the same groups from the two views now yields the nodes that fulfill the requirements of both views. Since there could still be nodes that fulfill the requirements of the two views with different neighbor nodes, we add a node-link view to the system and visualize the results. Such nodes can be easily identified in such a view, since such neighbor nodes are already filtered out and the resulting structure does not match the requested one. In fact, such nodes will most probably have no neighbors or just a single neighbor with no further connections after such filtering. Finally, we still have to verify that the handlers do not communicate among themselves. Thankfully, there are only six nodes not filtered out, and they match the requested layout, so those steps can be skipped. The final state of the system can be seen in Figure 12.

6 Discussion and Future Work

While the presented view succeeds in offering the user the intended information, there are still a number of points that could be improved. As described in section 5, it is currently not possible to specify that a node needs to be connected to multiple nodes that are further all connected to the *same* node (i.e. a person in group A knows two persons in group B that *both* know the *same* person in group C). In the current version, this would require two views where the second instance contains the same groups as the first one but in reverse order. It would further require the user to check the result for fragments of solutions where one part managed to fulfill all requirements while the nodes used to fulfill them did not pass. However, it is easily possible to imagine a constellation that can not be answered with two instances of this view by requesting such a constellation multiple times in the same query. A possibility to exactly specify the required layout would be a significant improvement for this visualization.

Another extension concerns the support of fuzzy groups. If the given group values would be treated as a percentage of group membership instead of a binary value, the user could dynamically adjust the threshold and thus alter the group membership (i.e. for a group defined by age, the user could see if reducing the threshold by a single year would cause a significant change in the result of the query). This would reduce the time required to modify groups significant. It would further be possible to change the binary group membership to show one or more colored sections that reflect how close these sections are to fulfilling the group requirements. This would enable the user to assess the impact of changes to the group membership.

7 Conclusions

This paper presented a novel view for systems that utilize multiple linked views with the goal of giving users that are interested in searching for complex relations within a multivariate network using a multiple linked view environment. Our approach supports the study of queries concerning relationships between groups in a network. To further investigate the results of such queries, this view has been integrated into a system containing views that can be used to further investigate the results. A useful combination with our view that is available in this system is in the form of an already existing node-link view. This combination has proven to be effective for both small networks and cases where the user already has at least a basic idea of the kind of relations he is interested in investigating.

8 Acknowledgments

This work was done at the VRVis Research Center in Vienna, Austria. Special thanks go to my supervisor Harald Piringer as well as Wolfgang Berger for the discussions we had while planning the view.

References

[ACJM03] AUBER D., CHIRICOTA Y., JOURDAN F., MELANÇON G.: Multiscale visualization of small world networks. In *INFOVIS* (2003), pp. 75–81.

[AMA08] ARCHAMBAULT D., MUNZNER T., AUBER D.: Grouseflocks: Steerable exploration of graph hierarchy space. *IEEE Transactions on Visualization and Computer Graphics 14*, 4 (2008), 900–913.

[AvHK06] 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.

[BKH05] BENDIX F., KOSARA R., HAUSER H.: Parallel sets: Visual analysis of categorical data. In *INFOVIS '05: Proceedings of the 2005 IEEE Symposium on Information Visualization* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 133–140.

[BMGK08] BARSKY A., MUNZNER T., GARDY J., KINCAID R.: Cerebral: Visualizing multiple experimental conditions on a graph with biological context. *IEEE Transactions on Visualization and Computer Graphics 14* (2008), 1253–1260.

[FK46] FORSYTH E., KATZ L.: A matrix approach to the analysis of sociometric data. *Sociometry 9*, 4 (1946), 340–347.

[HF06] HENRY N., FEKETE J.-D.: Matrixexplorer: a dual-representation system to explore social networks. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 677–684.

[HF07] HENRY N., FEKETE J.-D.: Matlink: Enhanced matrix visualization for analyzing social networks. In *Proceedings of the International Conference Interact* (2007), pp. 288–302.

[HFM07] HENRY N., FEKETE J.-D., MCGUFFIN M.: Nodetrix: Hybrid representation for analyzing social networks, 2007.

[JHGH08] JIA Y., HOBEROCK J., GARLAND M., HART J.: On the visualization of social and other scale-free networks. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1285–1292.

[PS06] PERER A., SHNEIDERMAN B.: Balancing systematic and flexible exploration of social networks. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 693–700.

[PTMB09] PIRINGER H., TOMINSKI C., MUIGG P., BERGER W.: A multi-threading architecture to support interactive visual exploration. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1113–1120.

[PvW08] PRETORIUS A. J., VAN WIJK J. J.: Visual inspection of multivariate graphs. *Comput. Graph. Forum 27*, 3 (2008), 967–974.

[PW06] PRETORIUS A. J., WIJK J. J. V.: Visual analysis of multivariate state transition graphs. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 685–692.

[SA06] SHNEIDERMAN B., ARIS A.: Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 733–740.

[Shn94] SHNEIDERMAN B.: Dynamic queries for visual information seeking. *IEEE Softw. 11*, 6 (1994), 70–77.

[Shn96] SHNEIDERMAN B.: The eyes have it: A task by data type taxonomy for information visualizations. In *VL '96: Proceedings of the 1996 IEEE Symposium on Visual Languages* (1996), IEEE Computer Society, p. 336.

[Shn97] SHNEIDERMAN B.: *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[Wat04] WATTS D. J.: *Six Degrees: The Science of a Connected Age*. W. W. Norton & Company, February 2004.

[Wat06] WATTENBERG M.: Visual exploration of multivariate graphs. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems* (New York, NY, USA, 2006), ACM, pp. 811–819.