B. Gretarsson, J.O'Donovan, T.Hollerer

Evaluating Interactive Manipulation of Graph Layouts

ABSTRACT

Node-link graph visualization has been used over the years in many applications and much research has focused on graph layouts in particular. Many such graph visualization systems are interactive, allowing user interaction with the visualization in many different ways. One common way of interacting with a graph visualization is to manipulate the layout by dragging nodes on the screen. However, there has been a lack of scientific evaluation of the benefits of such interactions with regard to solving common graph analytics tasks. This paper provides a comparative evaluation of three different techniques for interactive manipulation of graph layouts in a user study, where participants performed common graph tasks using the three different manipulation techniques. The results indicate a significant difference between the performance of the methods for certain tasks. We additionally tested the best performing method overall against the same visualization interface without any manipulation of the graph layouts. This demonstrated the benefit of interactive manipulation over simply browsing a static graph layout using zooming, panning and node highlighting on selection.

Author Keywords

Graph Visualization, User Study, Graph Manipulation, Interaction

ACM Classification Keywords

H.5.2 User Interfaces: Evaluation/methodology

INTRODUCTION

In the field of graph visualization, much work has focused on how to present a graph to users. In this paper, we consider the most common way of visualizing graphs, using a node-link representation. For these types of visualizations the main focus has been on layout algorithms. Many graph visualization tools also allow user interaction with the graphs in many ways. Typically, users can select nodes to highlight them, and sometimes connected nodes and edges will be highlighted as well. Selecting a node or an edge will usually bring up other detailed information about the selected entity. Additionally, many graph visualization tools will allow the users to move nodes around on the screen, thus manipulating the layout of the graph. We hypothesize that this specific type of interaction has the potential to help users perform common graph tasks otherwise difficult to complete. We identified two common methods of allowing interactive layout manipulation and one recently proposed method, and compared all three against each other in a within-subjects user study. We evaluated the accuracy and time taken for users to complete common graph tasks. The tasks were carefully selected so that they cover most of the task types identified in the Graph Task Taxonomy by Lee et.al. [13]. We had 72 participants in our user study and the results indicate that there are significant differences among the three methods for certain tasks.

We were also interested in finding if there was a significant difference between users using a static layout versus using one of the interaction methods. In order to test that, we set up a second user study in which the best performing interaction method was compared against meaningful static layouts, where the user could use zooming, panning, and highlighting, but no manipulation of the layouts. Additionally, we were interested in studying the effect that training had on the users' performance. Thus, this second study compared three conditions: no manipulation, manipulation with no training, and manipulation with specific training. For this study we had 24 participants (different from the 72 in Study 1) and the results show that interactive manipulation has significant advantages over no manipulation and that with more training users perform significantly better.

Our studies reveal that interactive manipulation of graph layouts has benefits in terms of solving common graph tasks. A significant difference in performance for the different interaction methods was revealed. We believe that the insights derived from our studies can serve to improve current graph visualization software and will stimulate and inform future research in the field.

RELATED STUDIES

Many studies on node-link graph visualizations have been reported. For example Blythe et al. [3] and Huang et al. [12] studied the effects of layout on inference from social network data. Purchase [10] studied the aesthetics of graph layouts via a user study which measured the response time and the number of errors for users completing tasks for various static graph layouts. Ware et. al [20] also evaluated the aesthetic properties of graph layouts via a user study, revealing the importance of path continuity to perceiving shortest paths. Ware and Bobrow [18] evaluated the effects of using motion for highlighting nodes and edges connected to the selected node and later Ware and Mitchell [19] evaluated combining motion and stereo displays for highlighting. Dwyer et al. [6] compared user generated layouts against automatic layout algorithms, with force directed layouts coming out as the overall winner. All these studies analyze effects of graph layouts on user performance, but none compare different layout manipulation techniques as we do in this paper.

CHI 2012, May 5–10, 2012, Austin, TX, USA.

STUDY 1: COMPARING MANIPULATION TECHNIQUES

In this section we describe our user study design and setup for Study 1, which compares three different methods of graph layout manipulation. We discuss the dependent and independent variables, the participants, the methods that we tested, the data that we used, the visualization software used for running the study, the training that we gave the participants for each method, and the tasks that they had to perform.

Study Design

The independent variables in our study are manipulation technique and graph size. Dependent variables are time, correctness and error. Time represents the time in milliseconds taken to complete a specific task. Correctness is a boolean indicating whether the user arrived at a correct answer or not. The error metric indicates how far the user was from a correct answer. For tasks that required the users to enter a numerical value, error is the absolute difference between the correct answer and the number entered. When users were asked to enter the names of multiple nodes then error was the difference of the number of answers asked for and the number of correct answers. When the users were asked to pick the most highly connected node, error was the difference in degree of the selected node and the actual most connected node.

Participants

The study consisted of 72 participants, mostly undergraduate students at a university, although a few were graduate students (less than 10%). Average age was 20.78 years, ranging from 18 to 31. 32 were male and 40 were female. The study lasted 60 minutes on average. Participants were compensated for their time by a nominal amount. In a pre-study questionnaire, participants were asked to rate their familiarity with topics related to the study on a 1 - 5 scale. On average they rated their familiarity with social networks as 3.83. The participants' familiarity with visualization of social networks was 2.65 on average. Their familiarity with graph theory was 2.42, and finally their familiarity with graph visualization was 2.31.

Methods Tested

In order to accurately test the benefits of interactive manipulation of graph layouts, a representative set of methods was selected and compared. We identified two common ways of providing interactive manipulation of graph layouts. The first method simply allows the user to drag any node to any position on the screen and the rest of the nodes remain stationary. We refer to this method as "Simple Manipulation" (SM). The second method works by continuously running a force directed (FD) layout while the user drags nodes around, giving a very dynamic feel to the interaction as the graph layout algorithm responds to changes in node position provided by the user. Finally, we selected a third, lesser known method, which was introduced as part of a graph visualization framework by Gretarsson et. al [9]. We believe that this method has good potential in terms of benefiting users when solving common graph tasks. We refer to this third method as the "Interpolation Method" (IM). We will now describe each of these methods in more detail.

Simple Manipulation (SM)

The simplest way of enabling manipulation of graph layouts is to allow dragging of individual nodes around the screen while other nodes in the graph remain stationary. Typically the user is also allowed to select multiple nodes and move them as a group. This way of interacting with graph layouts is very common in graph visualization software. Examples include Cytoscape [14], Tom Sawyer Visualization [15], Tulip [1], and others.

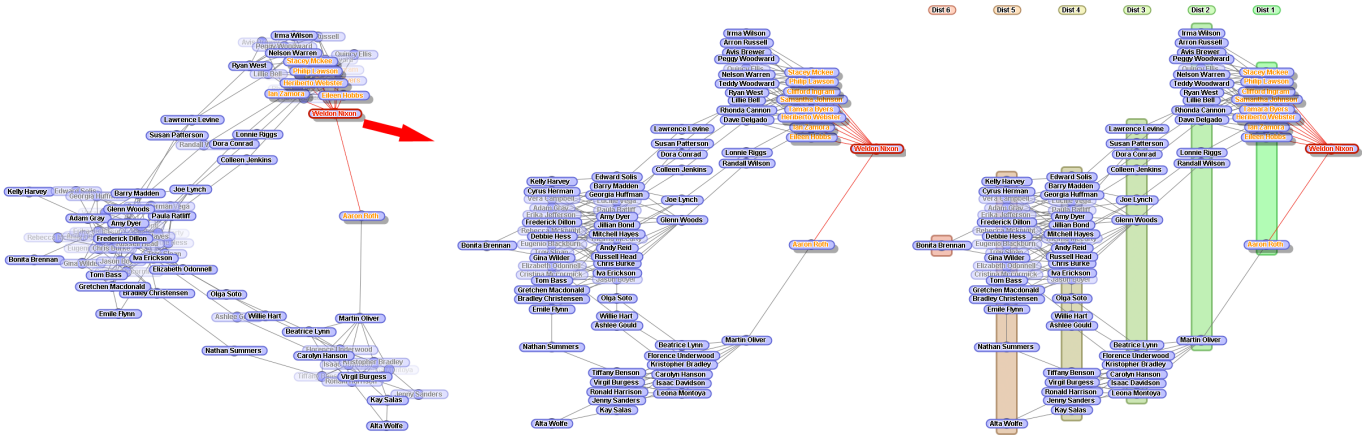
Force Directed Interaction (FD)

Force directed layout methods have been around for many years and are frequently used. Force directed layouts were presented by Eades in [7] and later improved by Fruchterman and Reingold in [8]. These layout methods are well known for producing aesthetically pleasing layouts of graphs as shown e.g. in a study by Dwyer et al. [6]. An additional benefit of force directed methods is that they can work incrementally from any arbitrary layout of a graph and thus they are often used for providing a dynamic interactive way of manipulating graph layouts. As the user drags a node around the screen, the layout algorithm adjusts the positions of other nodes, constantly trying to maintain a good layout. There are many implementations of this type of interaction methods available and currently in use, for example TouchGraph [16], the network visualization of Protovis [11], D3 [4] and more.

For our study we used TouchGraph's open source implementation of a force directed interaction. However, in order to run it with our larger graphs, we made a small improvement in the efficiency of how repelling forces are computed. We added a quad-tree to determine which nodes would potentially have effect on each node's position, thus reducing the per frame computation of repelling forces from $O(N^2)$ to approximately $O(Nlog(N))$. This change ensured that the well known scalability issues of force directed methods are not noticeable for the graph sizes that we used in our study. Our change did not affect the look and feel of the method in any other way than making it more responsive for larger graphs as verified by our pilot studies. There are other methods that are more scalable, such as Dwyers work in [5], but since the expirence is similar and our modification made FD sufficiently scalable, we opted for it as the more commonly used method.

Interpolation Method (IM)

The Interpolation Method (IM) was originally introduced in [17] and later included as part of the WiGis framework [9]. Node movement is interpolated along the edges of the graph to other nodes, such that nodes close to the dragged node (in terms of graph distance) move more compared to nodes farther away. In fact, the node farthest away from the dragged node will remain stationary. The version of this interaction method used in WiGis allows the user to change an effect parameter, which controls the locality of the effects of the interaction. For example, an effect parameter of 4 meant that only nodes that are 4 hops or less from the dragged node will be moved. And in the extreme case, an effect parameter of 0 would make this method equivalent to SM. However, to reduce the complexity during our experiment, and to ensure



(a) Before interaction. (b) After interaction. (c) After interaction. Distances highlighted

Figure 1. Interpolation method. (a.) A node is selected and dragged all the way over to the right. (b.) The nodes separate out into groups based on their distance from the selected node. (c.) Distances highlighted (this extra visualization was not available to the users during the study to keep the comparison fair).

all users had the same experience, we set it up so that the Interpolation would always affect the entire connected component. Additionally, using this interaction method, the entire graph is always kept on the screen (unless the user has zoomed in) so if the user drags a node towards the edge of the screen the positions of all the nodes are scaled to fit the entire graph on the screen. As a result, dragging one node off the edge of the screen results in a visualization where all the other nodes group into lines based on their graph distance from the dragged node, see Figure 1.

We had multiple reasons for choosing this method to compare against the other two, more common methods. First, we believe that this method can be informative for certain types of tasks, especially topology based tasks. Second, this method is very scalable when compared to FD as reported by [9]. Third, in it's simplest form (effect parameter = 0) this method is equivalent to SM and thus might gain common acceptance if it performs well in this study. Fourth, Ware and Bobrow [18] showed that animation of nodes close to the selected node is a beneficial way of highlighting, and using IM nodes move relative to their distance from the dragged node, which we believe could be beneficial in solving common graph tasks.

Data Used

For this user study we decided to use a social network as the data set. We had a couple of reasons for choosing social networks. First, it would be interesting to our users since they could imagine how they would visualize their own social network. Second, social network graphs are scale free graphs and thus an interaction method that works well for a social network should translate easily to other scale free graphs, which occur very frequently. Examples of other scale free networks include the internet, the world wide web, protein-protein interaction networks, airline networks, road networks and many more [2]. For reasons of privacy and experimental control we generated our own fictional social networks as opposed to using actual social network data. In order to ensure that we properly portray actual social networks we inspected the Facebook network of many individuals' immediate friends. We noticed that these social networks tended to have multiple clusters of people that the user had met in different places and then a few people connecting those clusters to each other.

In order to generate graphs that correctly portray such social networks, we used the Barabasi-Albert model [2] for preferential attachment, first to create multiple clusters and then once we had the clusters we would combine them all in one network and use the preferential attachment method to add more nodes that would act as the bridges connecting the clusters. We generated three small graphs ($\sim$ 80 nodes) and three large graphs ($\sim$ 1300 nodes) so that each user would get one small and one large graph for each interaction method. Our methods of generating the graphs ensured that they had similar properties for a fair comparison as verified by our pilot studies.

Finally, once we had generated the connectivity of the social networks, we needed to assign names to the nodes. We simply used a list of most frequent male and female first names along with the most frequent last names to randomly generate the names. We had one restriction though, since some tasks required the users to type the names of certain people in the graph (with the help of autocomplete), we needed to make sure that the names were unique after a certain number of characters had been typed. For the smaller graphs we made sure that the user only needed to type the first two characters of a name and it would be enough to identify the node and suggest the correct name to the user. For the larger graphs the user would sometimes have to type three or four characters before the correct suggestion came up.

Study Software

In order to ensure that all the users would have identical experience, we created an automated evaluation software which

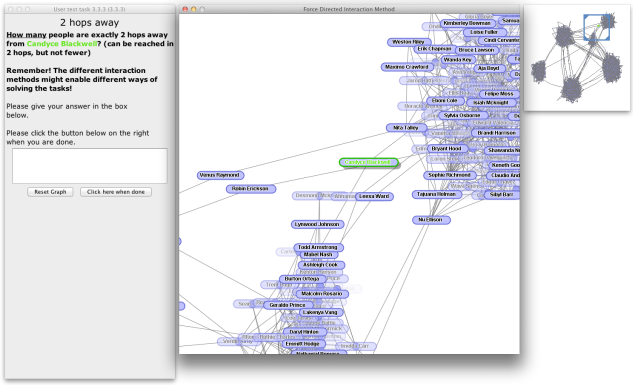


Figure 2. A screenshot from our user study software showing one of the large graphs. In this case the user has zoomed into a particular region of the graph as shown by the blue box in the overview window.

directed the users through the tasks and the training without any interference from a moderator. The software consisted of three windows side by side as shown in Figure 2. One window on the left hand side contained all the instructions for the users. This window would prompt the users to read the instructions first and then click a button when ready to see the graph and start looking for an answer. We recorded the time from the point they clicked the button until they submitted their answer. This window would also contain an input field for the users to type in their answer. A second window in the middle contained the interactive graph visualization. A third, smaller window on the right contained an overview, which always showed the entire graph and a box showing which part of the graph was currently being displayed in the main window. The user could use the scroll wheel on the mouse to zoom in to parts of interest or to zoom back out. Double clicking anywhere on the graph would reset the zoom. Our software would record every user interaction, allowing us to play back everything that the users did during the study. This enabled us to do in-depth analysis of user behavior after the study was completed.

Visualization Techniques

Since our focus was on comparing techniques for interactive manipulation of graph layouts, we kept all the visualization techniques constant across all methods. The nodes were always displayed as circles with labels on top, and the edges were displayed as straight lines. The graphs were all laid out beforehand using the same force directed layout algorithm as employed in FD, however allowing more time to reach a final layout in an offline process. Each time a graph was loaded, it would have the same layout for all users. Many studies, including [5], have shown that force directed layouts produce useful layouts, and are thus a good starting point for our interaction methods, as also verified by our series of pilot studies.

When a user moved the mouse pointer over a node, that node would always be brought to the front and highlighted. When a node was selected by clicking the mouse, it would be highlighted with a red label and a drop shadow, the edges connected to the selected node would also be highlighted in red and the nodes connected to the selected node would be highlighted with an orange label (see Figure 1). This made it easier for the users to find which nodes were connected to the selected node at any time. If we wanted to draw the users attention to a certain node, typically to have them perform a task related to that particular node, then the node would be highlighted in green (see Figure 2).

Autocomplete

For many of the tasks, the users needed to type in the names of people in the graph. To avoid typing errors, to reduce time spent on typing, and to avoid timing differences based on different names, we used an autocomplete mechanism which automatically suggested names from the graph based on what the user had typed. As mentioned in the data section above, we made sure that the names were unique after typing just a few characters. The autocomplete was based on all the nodes in the graph, not just the correct answers. The autocomplete ensured that the time spent on the task was mostly spent finding the answer and not actually typing it. Additionally, when users were asked to name nodes from the graph, every line in the user's response needed to exactly match the name of a person in the graph. This ensured that we would not incorrectly label an answer as incorrect if the user only made a typing error, although typing errors were very unlikely because of the autocomplete feature.

User Training

For our study we did not put any requirements on participants' prior knowledge or familiarity with graph visualization or graph theory. For this reason we started the study by introducing the relevant concepts for the tasks that we would give them. In addition to explaining how the social network was mapped onto the nodes and straight line edges, we explained how to count the graph distance between nodes, what it meant for a graph to be connected vs. disconnected, and what a clique is.

Once we had explained the graph concepts, we provided a tutorial showing how to make use of the the visualization components that were constant across all the methods. We made the user select a particular node, zoom into a particular region of the graph, zoom out again, select multiple nodes, delete nodes from the graph (although they would only be allowed to do that for certain tasks), and finally we asked them to type in the names of two nodes from a graph to familiarize themselves with the input and autocomplete.

After we had taught the users how to use the software, they would then be given a few training tasks for each method. We would ask them to drag a given node to a given position on the screen and point out how each method reacted to their movement. After completing the required training tasks, the users were allowed to freely play around with each method for up to 2 minutes. Once they had completed the training, they would begin solving the tasks that we were testing for both the small and large graph.

Tasks

We tested three different interaction methods for two different graph sizes. For each of those six conditions we gave the users 10 tasks to solve. Order was systematically varied using a balanced latin square approach, so that the users performed the tasks, methods, and graph sizes in different order. The only exception was for tasks 9 and 10 that were always performed last, because they were based on user recall.

We based our task selection on the Task Taxonomy for Graph Visualization by Lee et al. [13]. The task taxonomy lists tasks in the following categories:

- Low-level tasks [LL]
- Topology-based tasks [TB]
- Attribute-based tasks [AB]
- Browsing tasks [Br]
- Overview tasks [Ov]
- High-level tasks [HL]

We identified tasks that could potentially benefit from layout manipulation while trying to cover as many of the task types as possible. Here is a list of the tasks that we gave our users (X and Y replaced with appropriate names for each case):

1. Find how many friends X has [TB]
2. Find how many people are 2 hops from X [TB] [Br]
3. List 3 people that are 3 hops from X [TB]
4. List 3 people that are directly connected to both X and Y [TB]
5. List the nodes on the shortest path between X and Y [TB] [Br]
6. Find the most connected person in the graph [LL]
7. Find and delete the smallest set of nodes whose removal results in a disconnected graph with X in one component and Y in the other [Ov]
8. Find a clique of 3 nodes in the graph [Ov]
9. Try to recall how many nodes the graph had [Ov]
10. Try to recall the maximum degree of separation in the graph [Ov]

The attribute based [AB] tasks are not really applicable to interactive manipulation of layouts and thus we did not include them, while the high level [HL] tasks require a more involved user over a longer period of time and thus they were not feasible for the time frame allotted to the study.

Hypotheses

All our hypotheses are listed in Table 1. We did not expect any significant difference between the methods for tasks 1 and 6. In the case of task 1, we did not expect a significant difference since the task was simple enough to solve without any layout manipulation. While we did expect some methods to support general graph browsing better than others, we did not predict a benefit for solving task 6 using any of the methods. For tasks 2 through 5 we expected IM to do better than the other two methods. This is because these tasks are all topology based tasks and IM seems to be particularly effective when answering these types of questions, because one can easily drag one node to one side and reveal the nodes at various distances from the dragged node. For task 7 we again expected IM to perform better than the other methods, because dragging the two nodes in separate directions should reveal all the paths between them, although we were unsure if users would pick up on this benefit due to the complexity of this task. For task 8 we once again expected IM to perform better than the other methods, because if the user drags one node to one side, grouping together all the nodes directly connected to it, then the problem is reduced to finding two nodes connected to each other in that group. We could not find equivalent strategies using the other methods. Tasks 9 and 10 did not include any interaction and thus the time it took to answer them was not at issue. For Task 9 we did not expect any significant difference between the methods since none of them specifically revealed how many nodes were in the graph. For Task 10 we expected IM to perform better than the other methods, because if the user drags a node to one side then the degree of separation for that given node is revealed and the user only has to count the groups of nodes to know the maximum distance. We expected that the fact that the users had done this type of interaction several times while solving the tasks would benefit them when trying to recall the maximum separation. We did not feel that the other methods showed any benefit to this effect and thus expected them to perform worse than IM but did not expect any difference between them.

Task	Time	Correctness	Error
1	No significance	No significance	No significance
2	IM faster than SM IM faster than FD SM faster than FD	IM more correct than SM IM more correct than FD SM more correct than FD	IM less error than SM IM less error than FD SM less error than FD
3	IM faster than SM IM faster than FD SM equivalent of FD	IM more correct than SM IM more correct than FD SM equivalent of FD	IM less error than SM IM less error than FD SM equivalent of FD
4	IM faster than SM IM faster than FD SM equivalent of FD	IM more correct than SM IM more correct than FD SM equivalent of FD	IM less error than SM IM less error than FD SM equivalent of FD
5	IM faster than SM IM faster than FD SM faster than FD	IM more correct than SM IM more correct than FD SM more correct than FD	IM less error than SM IM less error than FD SM less error than FD
6	No significance	No significance	No significance
7	IM faster than SM IM faster than FD SM equivalent of FD	IM more correct than SM IM more correct than FD SM equivalent of FD	IM less error than SM IM less error than FD SM equivalent of FD
8	IM faster than SM IM faster than FD SM equivalent of FD	IM more correct than SM IM more correct than FD SM equivalent of FD	IM less error than SM IM less error than FD SM equivalent of FD
9	N/A	No significance	No significance
10	N/A	IM more correct than SM IM more correct than FD SM equivalent of FD	IM less error than SM IM less error than FD SM equivalent of FD

Table 1. List of all hypotheses for all tasks and dependent variables. Key: SM = Simple Manipulation, FD = Force Directed, IM = Interpolation Method

Results

For some of the tasks we had a few extreme outliers. This seemed to particularly apply for tasks where users had to type in a number (Tasks 1, 2, 9 and 10). We believe that this

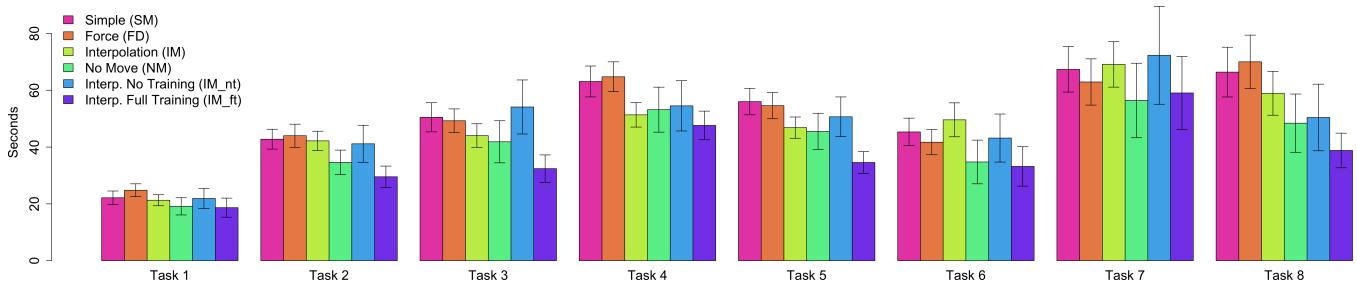


Figure 3. Average time for all timed tasks and methods in both studies. Note that smaller is better in this chart since it is showing time taken to complete the tasks. The first three bars in each task are from Study 1, while the second three are from Study 2. The error bars show standard error for a 95% confidence interval of the mean.

could have occurred because of innumeracy, but also because a small number of users may not have cared about correctness or may have mistakenly entered an extra digit. Also, since participants were being paid for their participation and they were not being monitored by a human moderator the whole time, we had a small number of users who decided to go through some of the tasks faster than they could find the correct answer. Additionally, we had some instances where users took extremely long on some tasks, perhaps because of a distraction of some sort or possibly because of fatigue. In order to filter such noise from our data set, we decided to remove a small number of outliers that were more than two standard deviations from the mean error or time for each of the tasks.

As mentioned before, our dependent variables were time, correctness, and error. For each task and each dependent variable we ran a two-way ANOVA with method and graph size as the two factors. When the two-way ANOVA indicated a significant difference across interaction methods we then ran Tukey's HSD test to determine between which methods there was a significant difference. All the significant results are listed in Table 2, including F and p values for our ANOVA and Tukey tests. In all cases where we found a significant effect of graph size on time, error, or correctness, it meant that the larger graphs took longer, produced larger error, or had fewer correct answers respectively. Whenever we found an interaction of method and graph size on one of the dependent variables it meant that there was a significant difference between the methods only for one of the two graph sizes. Since we are most interested in difference between the interaction methods we will focus the discussion on those cases where we found a significant difference between methods. Figure 3 shows the mean time for each of the tasks, Figure 4 shows the percent correct and the mean error is shown in Figure 5 and Figure 6. Please refer to Table 2 for all F and p values.

Task 1 - Count Friends

Using two-way ANOVA, we found significant effects of method and graph size on time, error, and correctness. Post-hoc Tukey's HSD tests showed that in all cases IM performed significantly better than FD, with the most significant result being in terms of correctness. There was not a significant difference between the performance of IM and SM or SM and FD.

The fact that we found a significant difference between the methods for this task was surprising to us, however based on user feedback we believe that while using FD the continuous movement of nodes while dragging a node around (although they did stop moving after a short time of no interaction) had a negative effect on the users' performance.

Task 2 - Count People 2 Hops Away

Two-way ANOVA revealed a significant effect of graph size on time, but no significant effect of method on time. Two-way ANOVA also revealed significant effects of method and graph size on error and correctness. Tukey's HSD tests showed that SM performed significantly better than FD in terms of both error and correctness while IM performed better than FD in terms of correctness. There was not a significant difference between the performance of IM and SM.

The results for this task were in line with what we expected, although we did not find a significant effect of method on time. Also, SM performed better than we expected when compared to IM. It seems that FD is in fact the worst choice of these three methods for this particular task.

Task 3 - List 3 People 3 Hops Away

Two-way ANOVA showed no significant effects in terms of time. Two-way ANOVA revealed a significant effect of method on error and correctness. Tukey's HSD tests showed that IM performed significantly better than FD in terms of error and correctness and also better than SM in terms of correctness.

The results for this task where in line with what we expected since IM outperformed both the other methods in terms of correctness and performed better than FD in terms of error. Although we did not get any significant difference in terms of time this still shows that IM is in fact the best of these three methods for this task.

Task 4 - List 3 People Connected to Both

Two-way ANOVA revealed a significant effect of method and graph size on time, error, and correctness. Tukey's HSD tests showed that IM performed significantly better than both SM and FD in terms of time. Tukey's HSD also showed that both IM and SM performed better than FD in terms of both error and correctness.

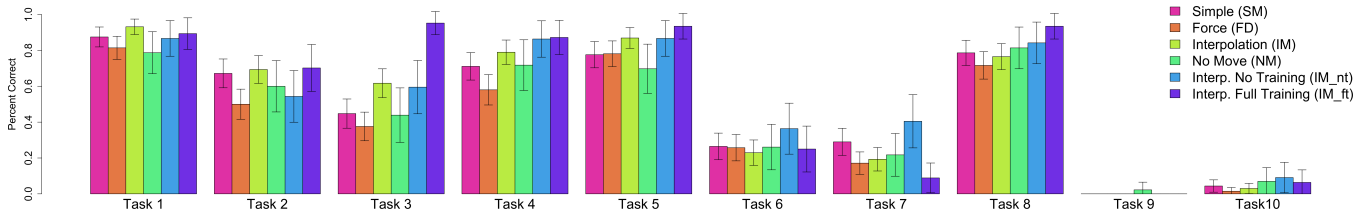


Figure 4. Percent correct for all tasks and methods in both studies. Note that bigger is better in this chart since it is showing percent correct. The first three bars in each task are from Study 1, while the second three are from Study 2. The error bars show standard error for a 95% confidence interval of the mean.

Study 1 Significant Effects						
Task	Variables		ANOVA		Tukey HSD	
	Dep. Var.	Ind. Var.	F-Value	$p <$	methods	$p <$
1	Time	Method	F(2,402)= 3.66	0.05	$IM < FD$	0.05
		Graph Size	F(1,402)=162.84	0.001		
	Error	Method	F(2,401)=3.71	0.05	$IM < FD$	0.05
		Graph Size	F(1,401)=57.14	0.001		
		Interaction	F(2,401)=3.68	0.05		
	Correct	Method	F(2,402)=5.16	0.01	$IM > FD$	0.01
		Graph Size	F(1,402)=76.23	0.001		
		Interaction	F(2,402)=5.01	0.01		
2	Time	Graph Size	F(1,398)=24.36	0.001		
	Error	Method	F(2,398)=3.44	0.05	$SM < FD$	0.05
		Graph Size	F(1,398)=19.22	0.001		
	Correct	Method	F(2,398)=6.72	0.01	$SM > FD$	0.01
					$IM > FD$	0.01
		Graph Size	F(1,398)=6.14	0.05		
3	Error	Method	F(2,419)=3.11	0.05	$IM < FD$	0.05
	Correct	Method	F(2,419)=9.01	0.001	$IM > FD$	0.001
					$IM > SM$	0.05
4	Time	Method	F(2,398)=11.35	0.001	$IM < SM$	0.001
					$IM < FD$	0.001
		Graph Size	F(1,398)=157.10	0.001		
	Error	Method	F(2,398)=11.24	0.001	$IM < FD$	0.001
					$SM < FD$	0.05
		Graph Size	F(1,398)=94.01	0.001		
		Interaction	F(2,398)=7.24	0.001		
	Correct	Method	F(2,398)=8.53	0.001	$IM > FD$	0.001
					$SM > FD$	0.05
		Graph Size	F(1,398)=62.68	0.001		
		Interaction	F(2,398)=4.12	0.05		
5	Time	Method	F(2,377)=4.94	0.01	$IM < SM$	0.05
					$IM < FD$	0.05
	Error	Graph Size	F(1,377)=19.10	0.001		
	Correct	Graph Size	F(1,377)=17.33	0.001		
6	Error	Graph Size	F(1,395)=105.61	0.001		
7	Time	Graph Size	F(1,412)=123.95	0.001		
	Error	Graph Size	F(1,412)=26.54	0.001		
	Correct	Method	F(2,412)=3.39	0.05	$SM > FD$	0.05
		Graph Size	F(1,412)=16.21	0.001		
8	Time	Graph Size	F(1,391)=2.84	0.001		
	Error	Graph Size	F(1,391)=72.16	0.001		
	Correct	Graph Size	F(1,391)=58.93	0.001		
9	Error	Graph Size	F(1,404)=8.65	0.01		
10	Error	Graph Size	F(1,400)=6.68	0.05		

Table 2. This table lists all the significant results from Study 1. Note: For convenience we always list the better performing method first in in the methods column. For dependent variables time and error smaller values are better, thus $X < Y$ means that X performed better than Y. For dependent variable correct larger values are better, thus $X > Y$ in the methods column means that X performed better than Y.

The results for this task were close to what we expected. IM outperformed FD in all the metrics and outperformed SM in terms of time. Surprisingly, SM also outperformed FD in terms of error and correctness. This clearly shows that for this particular task IM is the overall best method and SM is a better option than FD.

Task 5 - List Nodes on Shortest Path

Two-way ANOVA revealed a significant effect of method on time. Tukey's HSD tests showed that IM performed significantly better than both SM and FD. There was not a significant difference between SM and FD. There was no significant effect of method on either error or correctness.

The results were as we expected in terms of time, although we were surprised that we did not get a significant difference in performance in terms of error or correctness. In order to verify that the benefit in time was not only due to people just answering quicker and perhaps not answering correctly when they answer quicker, we decided to look at the time only for those who answered correctly. A two-way ANOVA revealed that there was also a significant effect of method on time ($F(2, 304) = 5.46, p < 0.01$) when we only considered those that answered correctly. Tukey's HSD showed that IM performed significantly better than both FD ($p < 0.01$) and SM ($p < 0.05$) in terms of time for those that answered correctly. This shows that for this task IM is in fact the best of the three methods for this task.

Task 6 - Find Most Connected Node

Two-way ANOVA revealed no significant effects of method or graph size in terms of time or correctness. Two-way ANOVA revealed a significant effect of graph size on error. This simply means that the users had more error when dealing with the larger graphs. This was just as we expected since we were not able to find a specific advantageous strategy using any of the methods to solve this task.

Task 7 - Delete Nodes Until Graph Disconnected

Two-way ANOVA revealed a significant effect of method on correctness only. Tukey's HSD showed that SM performed significantly better than FD in terms of correctness. These results were not as we expected, probably because the task was rather complex and if the users did not have a full understanding of how they could use the interaction methods to their advantage then the task was probably most easily solved without manipulating the layouts.

Task 8 - Find a Clique of 3 Nodes

Two-way ANOVA revealed no significant effect of method on time, error, or correctness. After analyzing how the users reached their answers using the playback facility of our software we realized that the users mostly used the fact that selecting a node would highlight it's neighbors to find the answers to this task and since this was the same for all the interaction methods the results did not differ much. Participants did not pick up on advantageous strategies that IM could have enabled.

Task 9 - Recall Number of Nodes

We did not compare time for this task since it was only based on recall. Two-way ANOVA revealed no significant effect of method on error or correctness as we expected. We included this task to see if using one interaction method would give the users a better sense of the size of the graph over the others, but as we expected this did not come out.

Task 10 - Recall Maximum Degree of Separation

We did not compare time for this task since it was only based on recall. Two-way ANOVA revealed no significant effect of method on error or correctness. We had hypothesized that by using IM people would get a better sense of the degree of separation in the graph since it is easily visible after dragging a node to one side. However, it turns out that users only focused on the tasks they were solving each time and solving those did not result in a better recall of the degree of separation after using one interaction method over the others.

User Opinion - Post Study Questionnaire

After the users had completed all the tasks we asked them to rate their experience using each of the methods and we also asked them to pick which one they would like to see implemented by their favorite social network service, or indicate if they didn't want any of the methods tested. The users were also asked to give a reason for their choice. The average rating of SM was 3.46, while FD had an average rating of 2.36, and finally IM had an average rating of 3.81. One-way ANOVA revealed a significant effect of method on user rating ($F(2, 216) = 34.91, p < 0.05$) and Tukey's HSD showed a statistically significant difference between each of the method pairs ($p < 0.05$). When asked to pick their favorite method 44 users (61%) selected IM, 20 users (28%) selected SM, 7 users (10%) selected FD, and 1 user (1%) selected none. This shows a high user preference for IM over the other two methods.

Learning Effect and Other Findings

We also ran a few other tests to check for learning effects and other trends in the data. For these tests we ran ANOVA to check for significant differences in terms of correctness across all the tasks. We chose to only look at correctness, since the range for error was very different between the tasks and time was not recorded for two of the tasks since they did not involve interaction.

As stated earlier we balanced for the order in which the users used the methods, the graph sizes, and the tasks. This enables

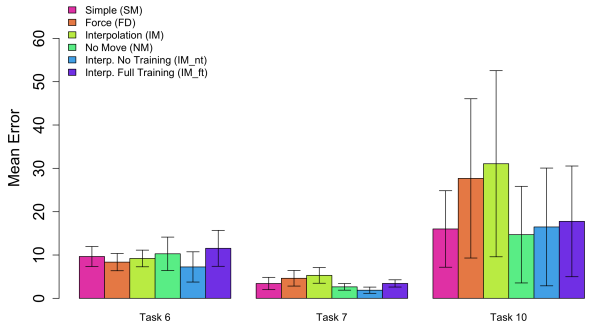


Figure 5. Average error for tasks 6, 8, and 10 and all methods in both studies. Note that smaller is better in this chart since it is showing mean error in the users' answers. The first three bars in each task are from Study 1, while the second three are from Study 2. The error bars show standard error for a 95% confidence interval of the mean.

us to check if there was any learning effect, which would show in better performance for the later methods. One-way ANOVA did not reveal a significant effect of method order ($F(2, 4059) = 1.12, p = 0.33$) on correctness. This simply means that there was not a learning effect between methods. One-way ANOVA did reveal a significant effect of graph size order ($F(1, 4060) = 5.21, p < 0.05$) on correctness. This indicates that there was a learning effect within each method, since the users performed better with the later graphs (48.7% correct) than with the earlier graphs (45.1% correct). To check for learning effects between tasks we discretized the task order into "early" (first 3 tasks) or "late" (last 3 tasks) and ignored the tasks in between. We then checked for a difference in performance between those two groups. One-way ANOVA did not reveal a significant difference of task order ($F(1, 2424) = 0.97, p = 0.32$) on correctness. This means that there was not a significant learning effect between the tasks.

One noteworthy difference was between genders. One-way ANOVA revealed a significant effect of gender ($F(1, 4060) = 13.05, p < 0.001$) on correctness across all tasks. Males performed better with 50.1% correct while females had 44.4% correct.

Another relevant finding was regarding the users' own rating of their familiarity with graph visualization. One-way ANOVA revealed a significant effect of familiarity with graph visualization ($F(3, 4058) = 5.97, p < 0.001$) on correctness. Tukey's HSD tests showed that those who rated their familiarity as 1 (41.6% correct) performed significantly worse than those that rated their familiarity as 2 (48.3% correct, $p < 0.05$), 3 (48.6% correct, $p < 0.01$), or 4 (51.4% correct, $p < 0.01$). No users rated their familiarity with graph visualization as 5.

Overall Results

For tasks 1, 3, 4, and 5 IM is the overall best performing method. For tasks 2, 3, and 4 FD is the overall worst performing method. For task 7 SM seems to be the best performing method although only for one dependent variable and only significantly better than FD. Since IM performed

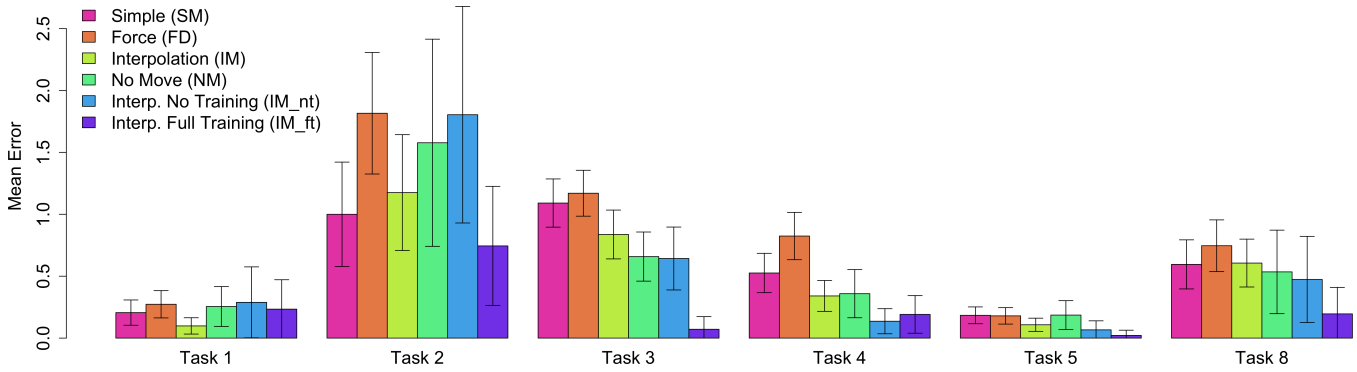


Figure 6. Average error for tasks 1-5 and task 8 and all methods in both studies. Note that smaller is better in this chart since it is showing mean error in the users' answers. The first three bars in each task are from Study 1, while the second three are from Study 2. The error bars show standard error for a 95% confidence interval of the mean.

significantly better than FD and SM in four tasks and was never significantly worse than either of the other methods, we can observe that among the tested methods IM has significant performance advantages for interactive manipulation of layouts in terms of solving common graph tasks. This is further backed by the results of the post-study questionnaire where the users gave their subjective rating and IM received the highest average score and was selected as the users' favorite method. FD certainly has it's advantages, mainly because of how dynamic it is and it can attract people's interest in visualization, but in terms of solving these common graph tasks it is certainly not as effective as the other two methods.

STUDY 2: TRAINING AND MANIPULATION BENEFITS

Study 1 provided insights into the relative strengths and weaknesses of various graph manipulation techniques, but it did not answer the question if manipulation helped users with graph-related tasks as compared with simple interaction with static layouts. Also, we wanted to clarify the effect of user training on our best performing method (IM). We compared the best performing method (IM) with no specific training (IM_{nt} - for IM no training) against the same method after training the users for the specific task types (IM_{ft} - for IM full training) that they had to solve. These two were also compared against the users solving the tasks without being able to manipulate the graph layouts at all (NM - for no move), but still being able to use all the other interactive components of the system. Such as highlighting nodes by selection, and zooming and panning. In order to test the difference between training and no training we had to let the users perform the tasks using IM_{nt} before IM_{ft}. The order of NM relative to the other two was varied and balanced across the users. As in Study 1, the starting layout for all the graphs was the result of a force directed layout method.

Participants

In this study we had 24 participants, a mix of undergraduate and graduate students at a university. The average age of the users was 21.54 years, ranging from 19 to 26. 14 users were male and 10 were female. The study lasted 60 minutes on average and users were compensated for their time as in Study 1. The same pre-study questionnaire as in Study 1 was used. On average the users rated their familiarity with social networks as 4.08. The users familiarity with visualization of social networks was lower at 2.79 on average. Their familiarity with graph theory was even lower at 2.67 and finally their familiarity with graph visualization was the lowest at 2.33. Note that all these values are slightly higher than in Study 1, although there is not a significant difference in any of the values.

Results

We list all the results where we found a significant effect of method on performance from Study 2 in Table 3. Due to space constraints we will not discuss the results for all the tasks, but rather discuss the highlights of these results. Figure 3 shows the mean time for each of the tasks, Figure 4 shows the percent correct and the mean error is shown in Figure 5 and Figure 6.

For task 3 IM_{ft} performed significantly better than IM_{nt} in terms of time, error, and correctness and it also performed significantly better than NM in terms of both error and correctness. This indicates that training helps a lot for this task, but also that Interactive manipulation of layouts is better than NM for this particular task.

For task 5 IM_{ft} performed significantly better than NM in terms of time, error, and correctness, and it also performed significantly better than IM_{nt} in terms of time. This shows that interactive manipulation of layouts is beneficial for solving this task, but also that training helps the users get the answer more quickly.

For task 7 we got the most surprising result, IM_{ft} performed significantly worse than IM_{nt} in terms of error and correctness, meaning that training had a negative effect. We believe that we have a good explanation for this. The strategy that we decided to teach the users in the training phase was not very easy to understand and with limited training time may have misled/confused the users rather than helped. As it turns out this strategy lead to a worse performance than no training at all. With the insight from this study, we would have taught a different strategy.

Study 2 Significant Effects						
Task	Variables		ANOVA		Tukey HSD	
	Dep. Var.	Ind. Var.	F-Value	$p <$	methods	$p <$
2	Time	Method	F(2,132)=5.50	0.01	$IM_{ft} < IM_{nt}$	0.01
3	Time	Method	F(2,119)=8.05	0.001	$IM_{ft} < IM_{nt}$	0.001
	Error	Method	F(2,119)=11.05	0.001	$IM_{ft} < IM_{nt}$	0.001
					$IM_{ft} < NM$	0.001
	Correct	Method	F(2,119)=15.64	0.001	$IM_{ft} > IM_{nt}$	0.001
					$IM_{ft} > NM$	0.001
5	Time	Method	F(2,128)=7.73	0.001	$IM_{ft} < IM_{nt}$	0.001
					$IM_{ft} < NM$	0.05
	Error	Method	F(2,128)=3.98	0.05	$IM_{ft} < NM$	0.05
	Correct	Method	F(1,128)=5.08	0.01	$IM_{ft} > NM$	0.01
7	Error	Method	F(2,127)=4.14	0.05	$IM_{nt} < IM_{ft}$	0.05
	Correct	Method	F(2,127)=7.06	0.01	$IM_{nt} > IM_{ft}$	0.001

Table 3. This table lists results where there was a significant effect of method on performance in Study 2. Note: For convenience we always list the better performing method first in in the methods column. For dependent variables time and error smaller values are better, thus $X < Y$ means that X performed better than Y. For dependent variable correct larger values are better, thus $X > Y$ in the methods column means that X performed better than Y.

Even though a direct comparison of data between the two studies is not exactly valid, due to the different populations and the difference in population size, comparing across the two studies can give some indications of trends. Figure 3 shows that IM_{ft} tends to be the best performing method of them all across all tasks in terms of time. Figure 4 also shows that IM_{ft} tends to be the best performing method in terms of correctness, except for task 7 which we have already discussed in the paragraph above. Figures 5 and 6 also show that IM_{ft} tends to be the best performing method in terms of error, again with the exception of task 7.

Overall, this study shows that for at least two of the tasks tested, interactive manipulation of graph layouts (IM_{ft}) has advantageous benefits over simple interactions with static graph layouts (NM). It also shows that with even the limited training possible in a study like this, the users performed significantly better than with no specific training.

CONCLUSIONS

This paper presented a comparative study of three methods for interactive manipulation of graph layouts in terms of solving common graph tasks. These tasks were selected so that they would accurately represent the tasks listed in a known task taxonomy for graph visualization. Study 1 showed that overall IM performed better than two more commonly known methods. Study 2 also showed that with more training users could do even better for many of the tasks, while with no particular training at all their performance was about as good as the performance in Study 1 which included a very limited, non-task specific training. Study 2 also showed that overall IM performed significantly better than a method that did not allow the users to manipulate the layouts at all (NM). This shows that there is a benefit in allowing users to manipulate graph layouts when solving graph tasks and that manipulation helps in solving these tasks faster and more accurately. Based on these results, we recommend that for interactive manipulation FD not be used when solving common graph tasks, unless it provides other strong benefits, such as attracting the user with its dynamic properties. Also, IM should be considered when providing an interactive graph visualization. We hope that our work will motivate further studies on the usefulness of various graph manipulation techniques. We believe that there is room for improvement in this space as so far there has been limited focus on creating and evaluating methods for interactive manipulation of graph layouts.

REFERENCES

1. Auber, D. Tulip. In *9th Symp. Graph Drawing*, P. Mutzel, M. Jünger, and S. Leipert, Eds., vol. 2265 of *Lecture Notes in Computer Science*, Springer-Verlag (2001), 335–337.
2. Barabasi, A.-L., and Albert, R. Emergence of scaling in random networks. *Science 286* (1999), 509.
3. Blythe, J., McGrath, C., and Krackhardt, D. The effect of graph layout on inference from social network data. In *Graph Drawing*, F. Brandenburg, Ed., vol. 1027 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 1996, 40–51. 10.1007/BFb0021789.
4. Bostock, M., Ogievetsky, V., and Heer, J. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2011).
5. Dwyer, T. Scalable, versatile and simple constrained graph layout. *Comput. Graph. Forum 28*, 3 (2009), 991–998.
6. Dwyer, T., Lee, B., Fisher, D., Quinn, K. I., Isenberg, P., Robertson, G. G., and North, C. A comparison of user-generated and automatic graph layouts. *IEEE Trans. Vis. Comput. Graph. 15*, 6 (2009), 961–968.
7. Eades, P. A heuristic for graph drawing. *Congressus Numerantium 42* (1984), 149–160.
8. Fruchterman, T. M. J., and Reingold, E. M. Graph drawing by force-directed placement. *Softw. Pract. Exper. 21*, 11 (1991), 1129–1164.
9. Gretarsson, B., Bostandjiev, S., O'Donovan, J., and Höllerer, T. Wigis: A scalable framework for web-based interactive graph visualizations. In *GD'09: Proceedings of the International Symposium on Graph Drawing* (2009).
10. H.C, and Purchase. Effective information visualisation: a study of graph drawing aesthetics and algorithms. *Interacting with Computers 13*, 2 (2000), 147 – 162.
11. Heer, J., and Bostock, M. Declarative language design for interactive visualization. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2010).
12. Huang, W., Hong, S.-H., and Eades, P. Layout effects on sociogram perception. In *Graph Drawing*, P. Healy and N. Nikolov, Eds., vol. 3843 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006, 262–273.
13. Lee, B., Plaisant, C., and Parr, C. S. Task taxonomy for graph visualization. In *In Proceedings of BEyond time and errors: novel evaLuation methods for Information Visualization (BELIV06*, ACM Press (2006), 82–86.
14. Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res 13*, 11 (November 2003), 2498–2504.
15. Tom-Sawyer-Software. Tom sawyer visualization. Proprietary online application, Tom Sawyer Software inc. available at http://www.tomsawyer.com.
16. Touchgraph. Touchgraph navigator. Proprietary online application, Touchgraph inc. available at http://www.touchgraph.com.
17. Trethewey, P., and Höllerer, T. Interactive manipulation of large graph layouts. Technical report, Department of Computer Science, University of California, Santa Barbara., 2009.
18. Ware, C., and Bobrow, R. Motion to support rapid interactive queries on node–link diagrams. *ACM Trans. Appl. Percept. 1* (July 2004), 3–18.
19. Ware, C., and Mitchell, P. Visualizing graphs in three dimensions. *ACM Trans. Appl. Percept. 5* (January 2008), 2:1–2:15.
20. Ware, C., Purchase, H., Colpoys, L., and Mcgill, M. Cognitive measurements of graph aesthetics. *Information Visualization 1* (2002), 103–110.