A Comparative Evaluation on Tree Visualization Methods for Hierarchical Structures with Large Fan-outs

Hyunjoo Song[1], Bohyoung Kim[1], Bongshin Lee[2], Jinwook Seo[1]

[1]School of Computer Science and Engineering
Seoul National University
599 Gwanak-ro, Gwanak-gu, Seoul 151-744, Korea
hjsong@hcil.snu.ac.kr,
{bhkim, jwseo}@cse.snu.ac.kr

[2]Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
bongshin@microsoft.com

ABSTRACT

Hierarchical structures with large fan-outs are hard to browse and understand. In the conventional node-link tree visualization, the screen quickly becomes overcrowded as users open nodes that have too many child nodes to fit in one screen. To address this problem, we propose two extensions to the conventional node-link tree visualization: a list view with a scrollbar and a multi-column interface. We compared them against the conventional tree visualization interface in a user study. Results show that users are able to browse and understand the tree structure faster with the multi-column interface than the other two interfaces. Overall, they also liked the multi-column better than others.

Author Keywords

Tree visualization, large fan-outs, multi-column layout, evaluation, browsing, revisit, and topology.

ACM Classification Keywords

H5.2. Information Interfaces and Presentation (e.g., HCI): User Interfaces-Evaluation/Methodology.

General Terms

Design, Human Factors

INTRODUCTION

Organizing large hierarchical information using a tree data structure is quite common. For example, the internet directory of the Web (e.g. the open directory project [14]) is hierarchical and can be visualized in the tree data structure. Another example could be ontologies used in many research fields such as biomedical informatics, artificial intelligence, and library science. Ontologies are graphs, but they are often presented as trees. As the hierarchical structure grows bigger, it becomes more and more difficult to understand the overall structure (i.e., the topology of the hierarchical structure) and to browse it efficiently.

CHI 2010, April 10–15, 2010, Atlanta, Georgia, USA.

One of the most troublesome problems occurring in browsing a large hierarchical structure is that the screen quickly becomes overcrowded when users open nodes with a large number of child nodes. For example, a phylogenetic tree (a tree used in biology showing the evolutionary relationships among biological species) has many internal nodes with several dozens of child nodes. Also the open directory has a category structure which consists of many nodes with more than 50 child nodes. This problem gets worse when it is required to show additional attributes of the node.

We can think of several remedies for this overcrowding problem. Focus+context techniques can show many more nodes in a screen by enlarging focused nodes and shrinking other nodes. Their distortion of information space may cause challenges in target acquisition [8], known as an overshooting problem. This could hinder users' browsing tasks [9]. Another solution is to use a list with a vertical scrollbar to show child nodes. Users may need to scroll up and down to select and open up a child node. The other alternative is to show child nodes using a multi-column layout, which requires more horizontal space than the list interface but could show more child nodes in a single view. Each solution has its pros and cons, but there has been no attempt to compare such alternatives in terms of accuracy and speed in performing tasks such as browsing and understanding large trees of large fan-outs.

In this paper, we present a comparative evaluation on three interfaces for showing child nodes of trees with medium and large fan-outs; traditional (baseline), list, and multi-column interfaces. We compared the three interfaces in supporting three different tasks (browsing, revisit, and understanding topology) using hierarchical datasets with medium (around 25) and large (around 50) numbers of child nodes at each level.

This paper is organized as follows. We first provide related work, and then we describe in detail three interfaces for viewing child nodes. Then we explain the design and procedure of our study, followed by the summary of our comparative evaluation results. We close our paper with in-depth discussion on the study results and future work.

RELATED WORK

Over the last few decades, tree visualizations have been extensively studied. There are two main categories of techniques for displaying and interacting with trees: node-link techniques (e.g., [16,22]) and space filling techniques (e.g., Treemap [10,19] and Information Slices [1]).

Cone Trees makes the context easily visible by using a focus+context technique [18]. To address scalability issue of Cone Trees, Carrièrre and Kazman suggested clustering nodes [6]. Hyperbolic Browser solves the occlusion problem of Cone Trees by using hyperbolic space instead of 3D perspective [12]. Degree of Interest Tree displays an overview of the tree by only showing detail for nodes with high degree of interest (DOI) values [5]. SpaceTree combines the node-link tree diagram with a zooming environment that dynamically lays out branches of the tree to best fit the available screen space [15].

While node link diagrams show topological structures well, they make inefficient use of screen space. On the contrary, the space filling techniques remedy the situation by making full use of screen space and efficiently visualizing trees that have attribute values at leaves. However, they do not convey topological structures well, since they are focused on containment rather than connection. Cushion Treemaps [21] tried to tackle this problem by adding shading as an extra cue. Elastic Hierarchies, a hybrid approach, combines the space efficiency of Treemap with the structural clarity of node-link diagrams [24]. Cheops compresses the tree by overlapping child nodes represented with triangles [4]. However, in spite of a very compact overview of huge trees, it is difficult to interpret the tree structures even with the various visual cues.

Given many tree visualization techniques, researchers have been trying to evaluate and compare their efficacy. Cockburn and McKenzie reported an empirical evaluation on the usability of the cone tree interface [7]. They found that many participants preferred the cone tree interface to a normal tree interface, although they were significantly slower at locating named files with the cone tree interface. Barlow and Neville compared four different visualizations with respect to their ability to communicate the topology of the tree and support comparisons of node size [3]. They found that Treemap-style was the slowest for most tasks. Plaisant et al. conducted a controlled experiment to compare SpaceTree with Microsoft Explorer and Hyperbolic browser [15]. They showed that SpaceTree works better than the others for estimating overall tree topology and revisiting previously visited nodes. SpaceTree was also found more attractive than Explorer. Kobsa conducted a controlled experiment to compare six tree visualization techniques [11]. Wiss et al. evaluated three 3D information visualizations with the hierarchical data [23], using seven high level tasks chosen from the taxonomy of tasks by Shneiderman [20]. Three visualization tools were compared with respect to their suitability for different datasets and their support for tasks. Risden et al. compared a 3D hyperbolic interface with two conventional 2D browsers by using the snap.com hierarchy contents [17]. While the study demonstrated the strengths and weaknesses of those three interfaces, there were no significant differences in overall user satisfaction across them. These efforts are summarized in Table 1 for easy comparison.

Despite the extensive previous research on tree visualizations, there has not been much effort to handle the trees with large fan-outs, especially when reading node labels is important. An interaction technique combined with dynamic node link layout was proposed to help users navigate and select a node in a large tree [2]. A multi-column layout approach was presented in TreePlus [13] while visualizing graphs as trees. However, the multi-column layout in TreePlus is not persistent in that once users open a node displayed in the multi-column TreePlus replaces the multi-column with the conventional layout. More importantly, the efficacy of these techniques has not yet been evaluated. In this paper, we compare three interfaces (traditional, list, and multi-column interfaces).

TREE VISUALIZATION INTERFACES

In this section, we describe three tree visualization interfaces that take different approaches to show child nodes: traditional interface (TRD), list interface (LST), and multi-column interface (MLC) (Figure 1).

Authors	Tasks Compared	Trees Compared
Cockburn and McKenzie [7]	Navigation	Cone Tree, Normal Tree
Barlow and Neville [3]	Navigation, Understanding Topology	Normal Tree, Tree ring, Icicle Plot, Treemap
Plaisant et al. [15]	Navigation, Understanding Topology	Collapsible tree, Hyperbolic Tree, SpaceTree
Kobsa [11]	Navigation, Understanding Topology	Treemap, Cushion Treemaps, BeamTrees, Hyperbolic Tree, TreeViewer, Collapsible tree
Wiss et al. [23]	Shneiderman's 7 tasks [20]	CamTree, Information Cube, Information Landscape
Risden et al. [17]	Navigation, Understanding Topology	Hyperbolic Tree, Collapsible tree

Table 1. Previous research on evaluations of tree visualizations

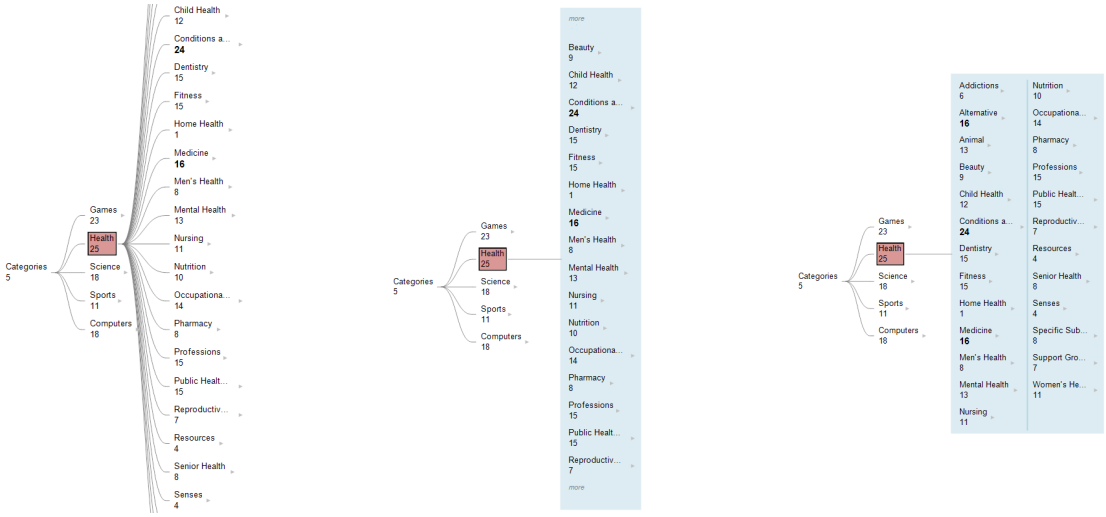
Traditional interface (TRD)

The traditional interface consists of nodes and edges (Figure 1a). Each node may have multiple rows of texts: its name and attributes of interest. When the node name is too long to display, it is abbreviated. Users can select a node with a mouse click. When selected, all child nodes of the selected node are displayed on the right side (Figure 2a). The selected node is highlighted with red background. Each child node is connected to the (selected) parent node with an edge and child nodes are arranged vertically in alphabetical order. A gray arrow is shown at the right end of a node when it has child nodes to give users a visual cue.

The traditional interface is not efficient in showing child nodes when the fan-out is large for the parent node due to its space inefficiency of wasting the parent side and cluttering the child side. We came up with two new alternative interfaces: list interface (LST), and multi-column interface (MLC).

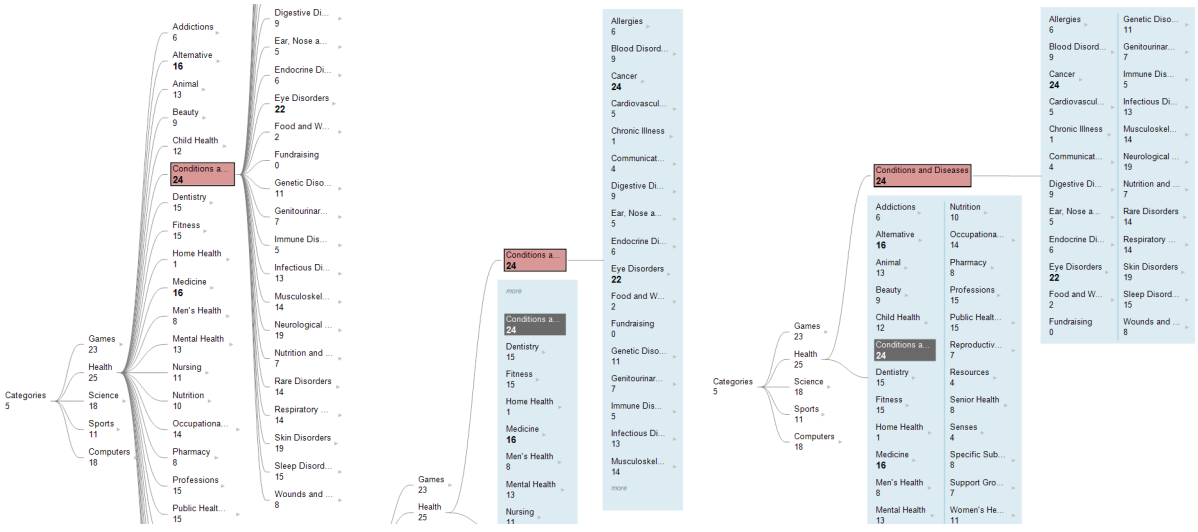
List interface (LST)

The list interface dynamically changes its appearance according to the number of child nodes to show. If there are just a small number of child nodes, LST shows the child nodes in the same way as the traditional interface. If there are too many child nodes to fit in the screen, it visualizes



(a) Traditional (b) List (c) Multi-Column

Figure 1. Three Tree Visualization Interfaces: Traditional (TRD), List (LST), and Multi-Column (MLC). TRD interface unfolds all child nodes in the traditional way. LST interface shows child nodes in a list with a cue node with the word, "more," indicating that there are more invisible ones. MLC interface shows all children in a tabular display with multiple columns.



(a) Traditional (b) List (c) Multi-Column

Figure 2. Three Tree Visualization Interfaces with a child node selected to open. TRD interface unfolds all child nodes in the traditional way. LST interface moves the selected node to the top of the list and shows its children in a list. MLC interface moves the selected node to the top of the multi-column view and shows the full name. In LST and MLC, a dummy node in gray background keeps the original position of the selected node that is moved outside.

child nodes using a custom list with background in blue (Figure 1b). Thus, users do not have to drag up and down the whole tree to explore the child nodes as with the traditional interface. In addition, since it renders only one edge from parent to the list containing child nodes, the view is less cluttered.

The height of the list is smaller than the vertical span of the screen by about twice the node height to give a space where neighboring branches can also be shown in the screen. When initially opened, the list shows the word, "*more*" at the bottom to indicate that there are more child nodes below.

When users start to explore the child nodes by moving the cursor over the list, the "more" node disappears and a standard scrollbar appears on the right side, with which users can scroll up and down to check previously invisible nodes (Figure 3). Since it behaves like the standard list, users can also scroll the list using the mouse wheel. Upon the mouse cursor exiting it, the list view hides the scrollbar and shows the "more" node appropriately at the top or bottom (or both).

When users select a child node in the list by clicking on it, the list dynamically changes its appearance again to move the selected child node to the outside, right to the top of it while preserving the alphabetical order among the moved-out nodes (Figure 2b). This transformation is rendered with a smooth animation. To avoid confusion caused by the change of the list content, a dummy node is created in the same position with the background in distinctive dark gray.

Users can select and close any opened node taken outside the list by clicking on the node or the corresponding gray dummy node. The selected node returns to the original position in the list where its dummy node is. Again the transition is done with a smooth animation to help users maintain context.

Multi-column interface (MLC)

The multi-column interface also changes its appearance dynamically according to the number of child nodes to show. If there are a small number of child nodes, MLC renders the child nodes in the same way as the traditional interface. If there are too many child nodes to fit in the screen, all of the child nodes are displayed using a multi-column layout (Figure 1c). Thus, users do not have to scroll up and down to explore the child nodes as with the list interface. Users, however, need to drag left and right to see all the child nodes if there are too many child nodes to fit in the screen even with the multi-column view.

When the child nodes are shown using the multi-column interface, all columns are balanced so that each of them holds the same number of nodes except that the last column could have a few less nodes. Balanced columns make sure that the screen space is efficiently used with no or minimum empty space in the multi-column view.

As in the LST, MLC also renders the background in blue when the child nodes are shown in a multi-column view. Columns are separated by a dark blue vertical line.

Once users find a node of interest among the child nodes shown in the multi-column view, users can click on the node to expand it. And then the multi-column view

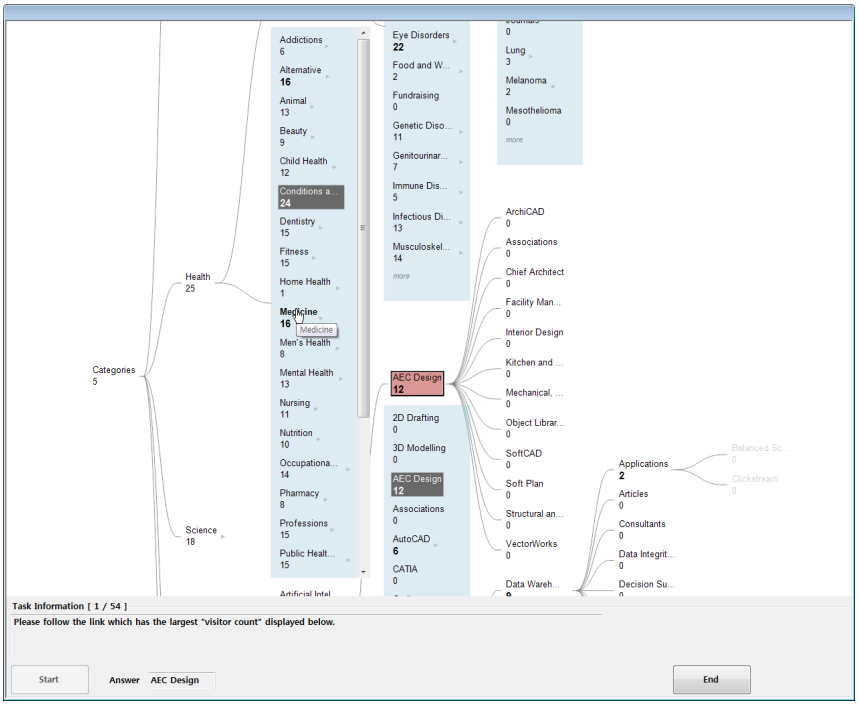


Figure 3. Study session management software. A scrollbar appears dynamically only when the cursor is on the list. Users can scroll through the list using the thumb or the mouse wheel.

dynamically changes its appearance with a smooth animation such that the selected node moves out of the tabular view to the top of the view with the alphabetical order preserved among the moved-out nodes (Figure 2c). In the same way as in the list interface, a dummy node is created at the same position in the multi-column view with background in dark gray. This transformation is also rendered with a smooth animation.

In this interface, when a node is selected and moved to the top of the multi-column view in the tabular layout, it is given a wide enough space to show its full name unlike traditional interface and the list interface. However, the corresponding dummy node in the multi-column view still remains in an abbreviated form (Figure 2c). Opening and closing of a child node are supported by smooth animations in the same way as in the list interface.

CONTROLLED USER STUDY

We conducted a controlled user study to compare the three tree visualization interfaces in terms of helping users perform with three important tasks: browsing, revisit, and topology understanding.

Participants

We recruited 18 participants (9 males and 9 females). 5 of them are majoring in Computer Science and Engineering, and others are roughly equally from 8 different majors. They are undergraduate students except for three office workers. We screened participants so that all participants know what the tree structure is but they had never used any tree visualization tools similar to the three interfaces. The participants' average age was 26.4 (26.5 for males and 26.3 for females), ranging from 22 to 32. They received about $10 for their participation. To increase motivation, a USB flash memory was given to the participant with the shortest completion time and the highest success rate.

Datasets and Tasks

Datasets used for our user study were generated from the Open Directory categories dataset available at http://www.dmoz.org/. To assure that no same tree dataset was used for different interfaces, we generated three trees with large fan-outs and three trees with medium fan-outs in addition to an example tree for training which was different from trees for real tasks. We partitioned the top level categories into roughly equal-sized three groups to generate three trees with medium fan-outs. We also generated three different trees with large fan-outs by partitioning the top level categories into another roughly equal-sized three groups. To maintain the same or similar level of task difficulties across the three interfaces, we trimmed some branches when necessary.

The number of child nodes for trees with medium and large fan-outs was determined by the resolution of screen and the size of nodes. In our experimental setup where we assumed that each node has two rows of texts (its name and a numeric attribute), TRD can show 24 child nodes in one screen. Participants have to pan the view by more than the height of the screen to check all child nodes using TRD when there are more than 50 child nodes. Thus, in each tree with large fan-outs, there were at least three nodes with about 50 child nodes on the path to the correct answer. In the trees with medium fan-outs, there were at least three nodes with about 25 child nodes on the path to the correct answer.

Each subject was asked to perform 3 types of tasks to measure the influence of adopting our interfaces. We chose the following three types of tasks because they are generally used for evaluation of hierarchical data visualization as summarized in table (Table 1).

Browsing: First-time node finding

Participants were asked to follow the path by selecting a node at each level that has the largest attribute value. To minimize the effect of cognitive burden for comparing too many numbers, we highlighted three nodes with largest attribute values at each level by showing the attribute value in bold. When participants selected a wrong node at a level, we showed a popup error message. This task tested how well people can navigate through trees.

Revisit: Visiting previously visited nodes

We asked participants to revisit the previously visited nodes after performing the browsing task twice. If participants can remember the approximate positions of the previously visited nodes, they can finish this task more quickly. If not, they have to repeat the browsing task again at each level. This task tested how well the three interfaces can help people remember the positions of the nodes on a previously visited path.

Topology: Listing all the ancestors of a node

Participants were asked to list the ancestors of a given node by clicking each node on the path from the root node to the node. The selected node was initially shown at the center of the view. This task tested how well people can understand the topology of a tree using the three interfaces.

Study Design and Procedure

We ran the study as a within-subject design that each participant performed all three types of tasks, using all three kinds of interfaces, with tree datasets of two different fan-outs. Each type of task is performed twice using the same tree but with different attribute values.

We used a 3 (Interface: TRD, LST, MLC) × 2 (Fan-out: medium, large) × 3 (Task Type: browsing, revisit, topology) design for our user study. To avoid the learning effect, we counterbalanced the order of interfaces using Latin Square Design. Participants always performed the three tasks in the order of browsing, revisit, and topology. For each task, they performed it with a tree of medium fan-outs first and then with a tree of large fan-outs.

Before beginning real tasks with an interface, we gave participants detailed instructions and showed them how to perform the three tasks with an example tree using the interface. They also had a chance to perform an example

task by themselves so that they could get used to the task and the interface. In total, we provided 6 training tasks (3 task types × 2 fan-outs) for each interface. After each session with an interface, we asked participants to fill out questionnaires for subjective evaluation. The same procedure was repeated with two other interfaces. Preferences and comments were collected during debriefing. The experiment took about 30 minutes.

Experiment Setup

Each participant worked on a quad-core PC with a 19" LCD display running at a 1280×1024 pixel resolution. The resolution of the tree view was 1264×840. All results were logged by session management software (Figure 3) running on the computer. For each trial, task descriptions were displayed at the bottom of the screen. After reading those descriptions, participants were asked to click on the "Start" button to indicate that they understood the task and they were ready to start the task. When finished, they were asked to click on the "Finish" button to submit an answer.

The session management software collected the task time, correctness of the answer, and the total length of panning. Only for the first task type (browsing), the software also logged the number of wrong selections and the total number of nodes at each level.

Hypotheses

We expected the traditional interface to perform poorly when browsing trees with large fan-out values. Since child nodes spread widely in the TRD interface, it might be inevitable for users to do a lot of panning to check the child nodes. We also speculated that it would be hard to follow the path from the root to a specific node because of the clutter made by all edges and the frequent panning activity. However, since the TRD interface is the most common way of showing trees, most people are used to it. Thus, we were not sure how this would affect the experimental results especially in terms of correctness and task time.

We also expected that the LST interface would be faster and more accurate to browse with less panning compared to the TRD interface because it shows all child nodes within a small list. The LST interface would suffer less from the cluttered edges because it renders only one edge from a parent to the list. In addition, since people are used to lists with scrollbars, they would be able to quickly scroll though child nodes in the list. Thus, we thought that it might as well be faster and more accurate than the TRD interface to follow a path.

The MLC interface unfolds all child nodes in a compact tabular layout, so users do not even have to scroll to check child nodes. There is much less clutter in the MLC interface since it draws only one edge to the tabular view of all children just like the LST does. We expected that the MLC interface could outperform the other two interfaces in terms of task completion time and error rate for all the three types of tasks. One thing that we were unsure about was the effect of some extra horizontal panning for the larger width of the tabular layout than the LST interface.

Statistical Analysis and Results

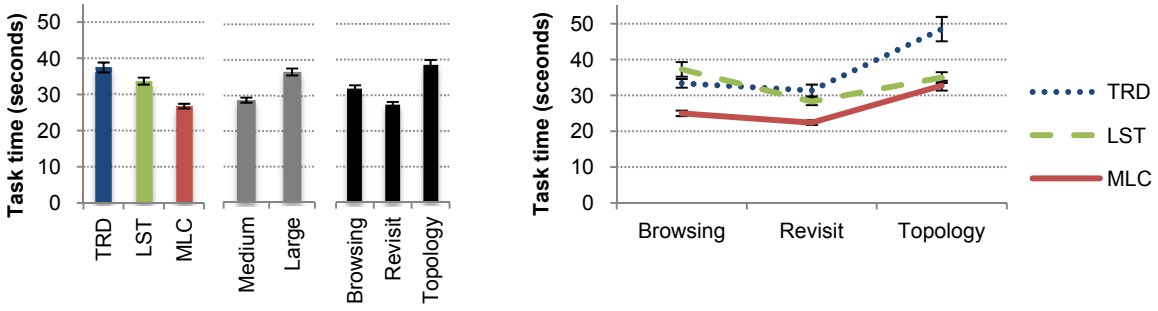
We analyzed five parameters from the experiment; total elapsed time to complete the task, number of mistakes in selecting a node while finding the answers, total length of panning per task, correctness of each task, and time to select one of the children for expansion. We also performed statistical analysis of questionnaire responses.

Time to complete the task

We analyzed the task completion time with a 3 (Interface) × 2 (Fan-out) × 3 (Task Type) repeated-measures analysis of variance (RM-ANOVA) and Tukey's HSD post-hoc test. We excluded the data for the incorrect answers.

We found a significant main effect of Interface ($F_{2,282}$=39.80, p<.001), with post-hoc tests showing that the effect was driven by the multi-column interface taking significantly less task time than either of the traditional and list interfaces (p<.001 and p<.001, respectively).

We also found significant main effects of Fan-out ($F_{1,282}$=59.46, p<.001) and Task Type ($F_{2,282}$=34.09, p<.001). It is not surprising that using a tree with large fan-outs would take more time to complete the task. It is also expected that the "topology" task would take more time as more panning operations were needed.



(a) Main effects **(b) Task Type vs. Interface**

Figure 4. Task completion time. There were significant main effects for all three factors. There was an interaction between Interface and Task Type.

Regarding all the task types, the multi-column interface took less task completion time than either of the traditional and list interfaces (Figure 4b). We found an interaction between Interface and Task type ($F_{4,282}$=4.75, p=.001) (Figure 4b). In the "browsing" task, the list interface took more time than other interfaces. Among all the possible explanations, it might be attributed to the fact that participants had to examine all the nodes to perform the "browsing" task. Thus the list interface that hides some of the nodes at first can make the first-time node finding task difficult. However, the list interface outperformed the traditional interface in the "revisit" task. More interestingly, for the more complicated task, or the "topology" task, the task time of the traditional interface drastically increased; however, the list exhibited a relatively moderate increase in the task time.

Mistakes made while finding the answers

We recorded the number of mistakes in the first task type (or, browsing) and analyzed them with 3 (Interface) × 2 (Fan-out) RM-ANOVA with Tukey's HSD post-hoc test. We found a significant main effect of Fan-out ($F_{1,102}$=12.05, p=.001); however, we found no effect of Interface ($F_{2,102}$=2.20, p=.116). A likely reason for this might be that participants only have to check three highlighted nodes at each level for the browsing task. Thus, the interface might influence the task completion time, but it was less likely that they could make mistakes choosing one out of three regardless of the interface type.

Total length of panning to complete the task

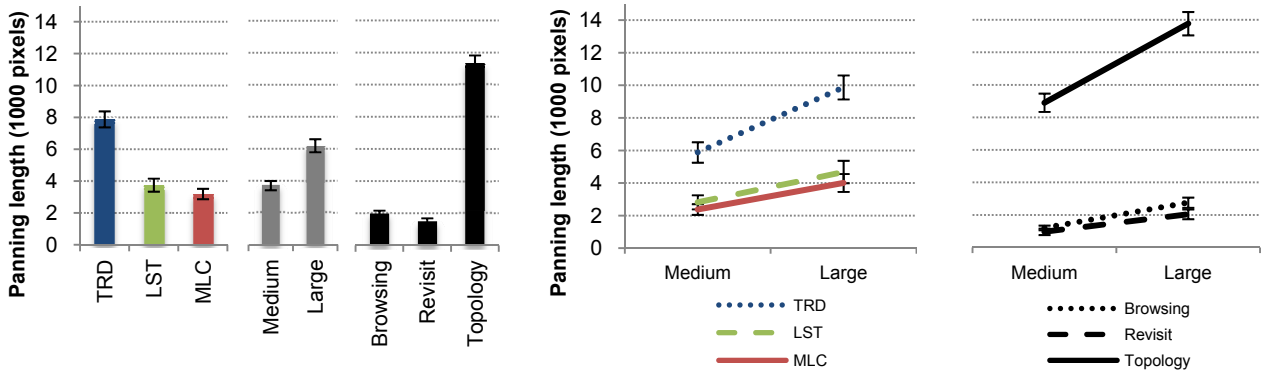
Length of panning in pixels was also recorded. The panning length was measured for each task type. We performed a 3 (Interface) × 2 (Fan-out) × 3 (Task Type) RM-ANOVA for the panning length.

We found a significant main effect of Interface ($F_{2,306}$=64.99 p<.001), with post-hoc tests showing that this was driven by the traditional interface leading to significantly greater panning length than either of the list and multi-column interfaces (p<.001 and p<.001, respectively) (Figure 5a). Interestingly, the multi-column interface took significantly less task completion time than the list interface, but the two interfaces did not show a significant difference in the panning length. The reason might be that the list interfaces does not need much more panning than the multi-column interface, but instead it requires the *scrolling* unlike the multi-column interface.

We also found a significant main effect of Fan-out ($F_{1,306}$=46.34, p<.001) (Figure 5a). This result was obvious as the tree with large fan-out values required participants to explore larger area.

We also found a significant main effect of Task Type ($F_{2,306}$=304.93, p<.001), with post-hoc tests showing that the effect was driven by the "topology" task requiring significantly frequent panning than either of the "browsing" and "revisit" tasks (p<.001 and p<.001, respectively) (Figure 5a). This can be attributed to the fact that there was no automatic centering of the focus node in the topology task, and participants had to manually pan the tree all the way up to the root node and down to the selected target node again to finish the task.

We found an interaction effect between Interface and Fan-out ($F_{2,306}$=4.22, p=.016) for the panning length (Figure 5b). Regarding the panning length, both of the list and multi-column interfaces were less affected by the fan-out values than the traditional interface. The way of displaying all the child nodes in the traditional interface might lead to drastic increase in the panning length when moving from trees with medium fan-outs to trees with large fan-outs. On the contrary, the scrolling in the list interface and the compact tabular display in the multi-column interface saved the participants from rigorously panning the trees. Participants might have to drag the mouse more with the multi-column interface because the tabular view takes a wider area than the list interface, but the difference might not be too big.



(a) Main effects **(b) Fan-out vs. Interface** **(c) Fan-out vs. Task Type**

Figure 5. Panning length. There were main effects for all three factors. There were interactions between Fan-out and Interface, and between Fan-out and Task Type. The error bar indicates the standard error of the mean.

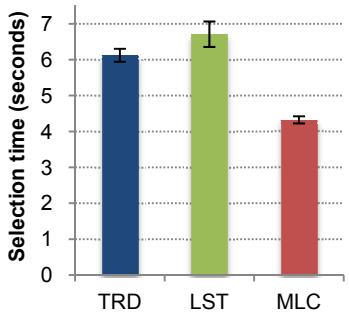


Figure 6. Time to select one child node. The error bar indicates the standard error of the mean.

Another interaction was found between Task Type and Fan-out ($F_{2,306}$=10.59, p<.001), indicating that the "topology" task was more sensitive to the fan-out value than either of the other two tasks (Figure 5c). As explained before for the main effect of Task Type, participants had to do more manual panning for the topology task than the other two types of tasks, and trees with larger fan-outs might require much more manual panning.

Correctness of answers

We analyzed the number of correct answers for the "revisit" and "topology" tasks types using a 3 (Interface) × 2 (Fan-out) × 2 (Task Type) RM-ANOVA with Tukey's HSD post-hoc test. We found no significant effects.

Time to select one of the child nodes

Participants accomplished each task by selecting the nodes which corresponded to the task description. For each node selection, we recorded the number of child nodes and the time to select one of them. We performed a 2-way ANOVA with Interface and the number of child nodes as fixed factors.

We found a significant main effect of Interface ($F_{2,1004}$=20.10, p<.001), with post-hoc tests indicating that MLC was significantly different from TRD and LST. The node selection time tended to decrease in the order of the list, traditional, and multi-column interfaces (Figure 6). We also found a significant main effect of the number of child nodes ($F_{25,1004}$=5.90, p<.001). It is not surprising considering that it takes more time to select one if there are many more choices. An interaction effect was found between Interface and the number of child nodes ($F_{48,1004}$=1.63, p=.005).

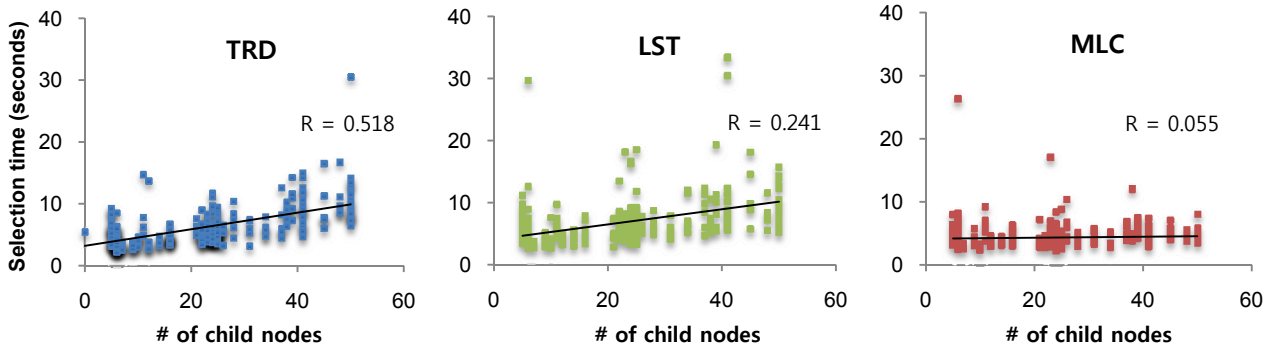
Question	TRD	LST	MLC
Q1. This interface is easy to learn.	5.8	5.5	5.9
Q2. This interface is easy to use.	4.9	5.3	5.6
Q3. This interface is fun. *	4.0	4.8	4.9
Q4. It is easy to understand the tree structure with this interface.	5.1	4.8	5.3
Q5. It is easy to follow the path with this interface. *	4.1	4.8	5.2
Q6. Overall, I like this interface.	4.3	4.7	4.9
Q7. I would like to use this interface again. *	4.0	4.4	4.7

Table 2. Average Likert scale ratings for the three interfaces using the scale of 1=Strongly disagree and 7=Strongly agree. The questions with significantly different ratings between three interfaces were marked with an asterisk (*).

We measured correlations between the number of child nodes and the selection time for each interface. The correlation coefficients of traditional, list, and multi-column interfaces were 0.518, 0.241, and 0.055, respectively (Figure 7). The coefficients were transformed with Fisher's z-transformation, and the results were compared with paired t-test. The transformed coefficient was significantly different only between the traditional and multi-column interfaces (t(359)=3.01, p=.003). This indicates that using the multi-column interface mitigated the effect of increasing number of children on the node selection time.

Subjective data

We asked each participant to answer 7 questions to collect subjective preferences over the three interfaces by using a 7 points Likert scale [Rating: 1=Strongly disagree; 7=Strongly agree] (Table 2).



(a) Traditional **(b) List** **(c) Multi-column**

Figure 7. Number of child nodes vs. time to select one child for expansion. The correlations were significantly different between TRD and MLC.

We ran Friedman Chi-Square tests for the ratings to find significant results for the questions 3, 5 and 7 (p=.002, p=.014, and p=.029, respectively) (Figure 8). Participants indicated that they enjoyed using the two new interfaces (LST and MLC) more than the traditional interface. Between the two, they preferred the multi-column interface to the list interface. This preference toward the multi-column interface was also confirmed by the fact that it was chosen to be the best interface by most people when asked to pick the best one (Figure 9). There was a significant difference among the three interfaces in terms of the number of people who voted for each interface (χ^2=14, p<.001).

Participants also indicated that it was easiest to follow the path with the multi-column interface than either of the other two interfaces and the list interface was better than the traditional interface. When asked if they wanted to use again next time, participants were more likely to use the multi-column interface again than the other two. We attribute this to the fact that people can see all child nodes at once without scrolling in the multi-column interface.

DISCUSSION AND FUTURE WORK

Our controlled user study results showed promising possibilities to support our hypothesis that the multi-column interface outperforms the other two interfaces in terms of task completion time and subjective satisfaction. We attribute this outcome to the fact that it can show all child nodes in a compact view so that users do not have to pan or scroll.

In our experimental setup, it turned out that overall, the wide width of the multi-column view did not influence participants' performance. However, it was interesting that one of the participants who majors in visual arts picked the multi-column interface as the worst interface, complaining that the visual design of the multi-column view was too wide to see them all.

While we expected that the familiarity with the list and scrolling interface might help participants performing some tasks, it was not the case. It was surprising to see that the list interface tended to be worse than the traditional interface for the browsing task in terms of task completion time. We attribute it to the fact that people are so used to the traditional interface that even scrolling could not significantly improve the total task time compared to the

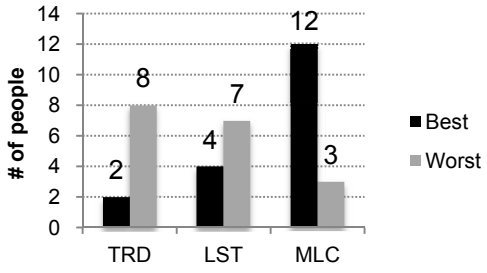


Figure 9. Best and worst interfaces.

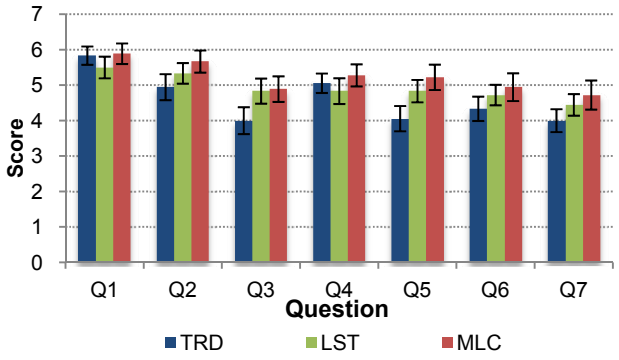


Figure 8. Subjective rating. The error bar indicates the standard error of the mean.

panning. It was also interesting to learn that the list interface did not outperform the multi-column interface in any tasks. The most likely reason for this is that the list has to hide many child nodes while the multi-column interface can show all child nodes at once.

Some participants said that it was harder to understand the tree structure with the list interface than with the traditional interface. There are two likely reasons for this. First, the traditional interface is more familiar form of tree visualization, which showed up in the freeform comments. Second, the visual cue (i.e., "more") to indicate that there are more to show above and below in the list interface was not efficient compared to the visual cue (i.e., edges) in the traditional interface. Thus, users sometimes do not notice that there are more hidden nodes in the list until they move the mouse over the list (and see the scrollbar).

Multi-column interface tries to resolve the issue of excessive occupation of vertical space for trees with a large fan-out by resorting to occupation of more horizontal space using multiple columns. In this experiment, we controlled the number of child nodes around 50 for the large fan-out case. Thus, participants could see all columns at once in the multi-column interface. When many more child nodes have to be shown, users may have to pan horizontally to see all the columns. Then the advantage of the multi-column interface that it can show all child nodes at once disappears. In that case, a focus+context technique that does not cause an overshooting problem might offer help. Or a horizontal scroll bar attached to the bottom of the multi-column interface can be an alternative. Furthermore, the necessity of horizontal scrolling increases with the trees of high depth. Additional user studies with a factor based comparison would be needed to generalize our results for other cases. For example, further evaluations using trees with larger fan-outs and with high depth would shed more light on the benefits of these alternative Multi-column interfaces.

CONCLUSION

We presented two extensions to the conventional node-link interface - a list view with a scrollbar and a multi-column - to help users better browse and understand the tree structures with large fan-outs. We conducted a controlled

experiment to investigate whether they could improve users' performance for three types of tasks; browsing, revisit, and understanding topology. We found that users browse and understand the tree faster with the multi-column interface, providing a compact view of child nodes that does not require scrolling. Overall, users also preferred the multi-column interface to the other two interfaces.

ACKNOWLEDGMENTS

We thank Prof. Johan Lim (Department of Statistics at Seoul National University) for his help with statistical analyses. We also appreciate our study participants for their time and comments. This work was supported by the Engineering Research Center of Excellence Program of Korea (MEST/KOSEF, R11-2008-007-01002-0), KOSEF grant (No. 2009-0064949), and the Brain Korea 21 Project. The ICT at Seoul National University provided research facilities for this study.

REFERENCES

1. Andrews, K. and Heidegger, H. Information slices: Visualising and exploring large hierarchies using cascading, semicircular disks, *Proc. InfoVis 1998*, (1998), 9-12.
2. Appert, C. and Fekete, J.-D. ControlTree: Navigating and selecting in a large tree. *UIST 2006 Conference Supplement*, (2006), 47-48.
3. Barlow, T. and Neville, P. A comparison of 2-D visualizations of hierarchies. *Proc. InfoVis 2001*, (2001), 131-138.
4. Beaudoin, L., Parent, M-A, and Vroomen, L. Cheops: A compact explorer for complex hierarchies. *Proc. Vis 1996*, (1996), 87-92.
5. Card, S. and Nation, D. Degree-of-Interest trees: A component of an attention-reactive user interface. *Proc. AVI 2002*, (2002), 231-245.
6. Carrièrre, J. and Kazman, R. Research report: Interacting with huge hierarchies: Beyond cone trees. *Proc. InfoVis 1995*, (1995), 74-81.
7. Cockburn, A. and McKenzie, B. An evaluation of cone trees. People and Computers XIV: Proceedings of the British Computer Society Conference on Human Computer Interaction, (2000), 425-435.
8. Cockburn, A., Karlson, A., and Bederson, B.B. A review of overview+detail, zooming, and focus+context interfaces. *ACM Comput. Surv.* 41, 1 (2008), 1-31
9. Gutwin, C. Improving focus targeting in interactive fisheye views, *Proc. CHI 2002*, (2002), 267-274.
10. Johnson, B. and Shneiderman, B. Treemaps: A space-filling approach to the visualization of hierarchical information. *Proc. Vis 1991*, (1991), 284-291.
11. Kobsa, A. User experiments with tree visualization systems. *Proc. InfoVis 2004*, (2004), 9-16.
12. Lamping, J., Rao, R., and Pirolli, P. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. *Proc. CHI 1995*, (1995), 401-408.
13. Lee, B., Parr, C.S., Plaisant, C., Bederson, B.B., Veksler, V.D., Gray, W.D., and Kotfila, C. TreePlus: Interactive exploration of networks with enhanced tree layouts. *IEEE TVCG Special Issue on Visual Analytics*, 12, 6 (2006), 1414-1426.
14. Open Directory Project, http://www.dmoz.org/, last accessed 01/07/2010
15. Plaisant, C., Grosjean, J., and Bederson, B.B. SpaceTree: Supporting exploration in large node link tree, design evolution and empirical evaluation. *Proc. InfoVis* 2002, (2002), 57-64.
16. Reingold, E.M. and Tilford, J.S. Tidier drawings of trees. *IEEE Trans. on Software Engineering*, 7, 2 (1981), 223-228.
17. Risden, K., Czerwinski, M., Munzner, T., and Cook, D.B. An initial examination of ease of use for 2D and 3D information visualizations of web content. *Internal Journal of Human Computer Studies*, 53, 5 (2000), 695-714.
18. Robertson, G.G., Mackinlay, J.D., and Card, S.K. Cone Trees: Animated 3D visualizations of hierarchical information. *Proc. CHI 1991*, (1991), 189-194.
19. Shneiderman, B. Tree visualization with tree-maps: 2-d spacefilling approach. *ACM Trans. on Graphics*, 11, 1 (1992), 92-99.
20. Shneiderman, B. The eyes have it: A task by data type taxonomy for information visualizations. *Proc. VL 1996*, (1996), 336-343.
21. van Wijk, J.J. and van de Wetering, H. Cushion Treemaps: Visualization of hierarchical information. *Proc. InfoVis 1999*, (1999), 73-78.
22. Walker, J.Q. A node-positioning algorithm for general trees. *Software-Practice and Experience*, 20, 7 (1990), 685-705.
23. Wiss, U., Carr, D.A., and Jonsson, H. Evaluating three-dimensional information visualization designs: A case study of three designs. *Proc. IV 1998*, (1998), 137-144.
24. Zhao, S., McGuffin, M.J., and Chignell, M.H. Elastic hierarchies: Combining treemaps and node-link diagrams. *Proc. InfoVis 2005*, (2005), 57-64.