ABSTRACT

Title of Document: INTERACTIVE VISUALIZATIONS FOR TREES AND GRAPHS

Bongshin Lee, Ph.D., 2006

Directed By: Professor Benjamin B. Bederson, Department of Computer Science

Graphs are a very commonly used information structure, and have been applied to a broad range of fields from computer science to biology. There are several important issues to consider when designing graph visualizations. One of the most difficult problems researchers face is how to visualize large graphs. While an algorithm may produce good layouts for graphs of several hundred nodes, it may not scale well to several thousand nodes. And, as the size of the graph increases, performance will degrade rapidly, making it difficult to build an interactive system. Label readability will also suffer, hindering users' abilities to understand the graph data and perform many tasks. Finally, even if a system can lay out and display large graphs, the cognitive demands placed on the user by the visualization may be overwhelming.

This dissertation describes and applies several design principles to various graph visualization domains to address these issues. Tightly-coupled and highly customized views were used for graph visualization in a novel way. A new tree layout approach to graph visualization was proposed with appropriate visualization

and interaction techniques. When visualizing graphs as trees, a guiding metaphor "Plant a seed and watch it grow" was used to support information gathering and detailed exploration of the graph's local structure.

Three graph visualization systems guided by these design principles were also developed and evaluated. First, PaperLens provides an abstract overview of the full dataset and shows relationships through interactive highlighting. It offers a novel alternative to the more common node-link diagram approach to graph visualization. Second, the development and evaluation of TaxonTree provided valuable insights that led to the design of TreePlus, a general interactive graph visualization component. Finally, TreePlus takes a tree layout approach to graph visualization, transforming a graph into a tree plus cross links (the links not represented by the spanning tree) using visualization, animation and interaction techniques to reveal the graph structure while preserving the label readability.

Other contributions of this work include the development of a task taxonomy for graph visualization and several specific applications of the graph visualization systems described above.

INTERACTIVE VISUALIZATIONS FOR TREES AND GRAPHS

By

Bongshin Lee

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2006

Advisory Committee:
Professor Benjamin B. Bederson, Chair
Dr. Catherine Plaisant
Professor Ben Shneiderman
Professor David Jacobs
Professor William Fagan

Dedication

To My Parents

Acknowledgements

When I first came to study abroad in 2000, I did not have a single friend or family member here in the United States. I am now deeply grateful that I have so many people to thank. My first, most, and special thanks should go to my advisor, Ben Bederson, for all of his guidance and support. One of the rumors among Korean students abroad says that the success of study abroad highly depends on the advisor. I was extremely lucky to have him as my advisor. He is not only a great researcher having creativity and enthusiasm but also a generous person having a kind heart and patience, which makes him a great advisor. Instead of simply telling me how to make my work better, he patiently waited until I could learn and follow.

I am very fortunate to have had an opportunity to work with Catherine Plaisant. Not only has she helped me improve my dissertation, but she has also helped me become a better researcher as well. I am grateful to have had Ben Shneiderman as a committee member, not only for his advice and encouragement but also for the example he provided me through his enthusiasm for research and learning. I would also like to thank the other committee members, David Jacobs and William Fagan, for their time and effort to help me improve my dissertation.

My two internships at the Visualization and Interaction Research (VIBE) group at Microsoft Research (MSR) were an invaluable experience. I would like to thank Mary Czerwinski, George Robertson, and Patrick Baudisch for their guidance and support. I am also grateful that they welcomed me as a colleague. I have learned so much from them - how to communicate and collaborate with other people, write

papers, demonstrate my projects, design user studies, and so on. Other interns, Duke Hutchings and Heidi Lam, made my life at MSR an even more enjoyable experience.

I would like to give a very special thanks to Cyndy Parr. Her encouragements as a coauthor, a colleague, and a friend enabled me to successfully go through my life as a Ph.D. student. I want to thank to the team members for each project I have worked on. Dana Campbell, Jeff Jensen, and Svetlana Yarosh helped me with the TaxonTree project. I also need to thank the project members at Rensselaer Polytechnic Institute (RPI) for the TreePlus project; Wayne Gray, Vladislav Veksler, and Christopher Kotfila. Without their help, my project and dissertation could not have been as successful as it is.

I would like to thank all of the members of the Human Computer Interaction Laboratory (HCIL). My very special gratitude is due to Aaron Clamage, who became one of my best friends. He always helped me out not only with Piccolo questions but also with reading this dissertation and many other papers. I am also grateful to Jinwook Seo for discussing the issues of information visualization, sharing a vision of the research, and becoming a colleague. His insightful comment on my research has been immensely useful. Bongwon Suh provided me with both professional and personal advice. Anne Rose deserves a special thank you. I was always impressed by her smile and willingness to help. Alex Aris, Allison Druin, François Guimbretière, Hilary Hutchinson, Hyunmo Kang, Amy Karlson, Amir Khella, Bill Kules, Jenny Preece, Kiki Schneider, Hyunyoung Song, Lingling Zhang, Haixia Zhao, and other HCIL members provided me with a wonderful research environment.

When it comes to friends, I always feel that I am the luckiest person in the world. All of my friends deserve special thanks for their support and warm wishes. Maria Adamou, an officemate and a roommate for my first internship at Telcordia Technologies, shares one of my happiest times in the United States. Jesse and Laurel Grosjean were always friendly and fun to be with. Bohyung Kim, Yunsoo Seo, Myoung-Soo Kang, Seung-Kyu Ko, and Changbin Ko treated me as if I was a family member and enriched my life in Maryland. Yoo-Ah Kim not only helped me with challenging questions on algorithms but also became a good friend. Seung-Jong Baek, Jusub Kim, Juhyun Lee, Yuri Lee, Yoonhee Shin, and Jinhee Yang also enriched my life in Maryland.

I would like to thank my friends in Korea for being such great friends and always sharing in my happiness and sadness. They welcomed me every time I visited Korea and made me feel beloved even though I was very far away. My deepest gratitude is due to Changkyu Lee for her encouragement for me to study abroad. Her sincere support and belief served as an impetus for me to overcome my shyness, improve myself, and achieve more. Jiyoun Kwon deserves a very special thank you. Since I first met her as a private tutor, she has been exceptionally friendly, supportive, and respectful. Sanghee Park also deserves a special thank you for being such a good supporter and friend. Yoonkyung Chang, Jaewon Choi, Songhwa Choi, Jangwook Ha, Seunghee Han, Eunjung Jang, Hyosun Kim, Jungmi Kwon, Nanyoung Lee Seungjin Lee, Chanjung Park, Seonae Son, Ilda Yoo, Seungkoo Yoon, and Sooyong Wang sent supports from Korea.

My very last and special thanks go to my family members. I am very fortunate to have a wonderful sister, Myungshin. Even though she is two years younger than me, she has become my best friend. And she was extremely supportive throughout my Ph.D. life. I also would like to thank my brother-in-law, who loves my sister and other family members very much. My brother, Yongjin, and sister-in-law deserve deep gratitude for their support and taking good care of my parents. I also would like to thank my lovely nephew, Taekkyu, who will be one year old in two months. He brought great happiness to my family. My parents deserve more than the deepest appreciation for their love, devotion, and belief in me over more than 30 years. While I acquired a lot of knowledge from courses and teachers, I have learned the most important things from my parents. They taught me I should be faithful, sincere, diligent, brave, humble, and sympathetic to other people. They always supported and loved me, regardless of what choices I made. Without their devotion, sacrifice, and love, I would not have been able to accomplish anything. I will always love my parents and I dedicate this thesis to them.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Graphs, composed of objects (nodes) and relations (links), are one of the most commonly used information structures. They have broad applicability in a wide range of fields, including computer science, engineering, sociology, and biology. The World Wide Web is an obvious example (web pages are nodes and hyperlinks are links). Other examples include file system hierarchies, organization charts, social networks, gene ontologies, and food webs. Over the last few decades, there has been a great deal of research on how to effectively display and interact with graphs like these [34, 66].

1.1 Key Issues in Graph Visualization

One key issue in graph visualization is the size of the graph. While many automatic graph layout algorithms successfully produce good layouts for small graphs with fewer than one hundred nodes [38, 50, 58], most of them are not applicable for larger graphs with several thousands of nodes. Only a few exceptional systems such as Tulip [8], Gem-3D [24], HDE [64], H3 [98, 99, 100], and NicheWorks [139, 140] can handle large graphs.

Another issue is that any interactive visualization system should provide near real-time performance. However, useful operations for drawing general graphs have been proven to be NP-complete [22]. Some researchers handle this problem by transforming the graph into a tree (e.g. spanning tree), which is more tractable. They

then visualize this tree, rather than the graph. This may not work for all general graphs because some information will inevitably be hidden. However, it could work for some tree-like graphs if special visualization and interaction techniques are added to help users find the missing information.

While tree layout based approaches may not be suitable for overview related tasks, they can provide better support for tasks involving reading labels and attributes. Example tasks might include find and review 1) the nodes adjacent to a node; 2) the nodes accessible from a node; 3) the nodes adjacent to two given nodes; 4) the shortest path between two nodes; 5) the nodes having a specific attribute value; 6) the nodes connected only by certain types of links. Another example would be to list all labels in a sub-graph and follow a path. For each of these tasks, users need to read labels to make sense of the data.

Many visualization techniques have already been developed to present as much information as possible given limited screen space. The simplest approach is to allow users to zoom and pan the visible area [15, 72]. However, users often lose track of where they are within the global structure. "Overview plus detail" techniques try to solve this problem by providing an overview window [10, 108]. However, these visualizations force users to continually switch between two views and reorient themselves. "Focus plus context" techniques, on the other hand, integrate detailed views with as much surrounding context as possible, so that users can see all relevant information in a single view [13, 48, 52, 119, 120]. However, the distortion makes it difficult for users to navigate.

Even if systems can lay out and display large graphs, the cognitive demands placed on users by the visualization can be significant. Displaying an entire graph may provide users a way to see the overall structure of the graph. However, viewing this much information at once can be overwhelming. It is also difficult to interact with the graph because of the dense layout and occlusion. One way to tackle these problems is to reduce the size of the graph that will actually be displayed. Instead of showing the entire graph at once, the system can display only a portion of it and show other parts as needed. One obvious way to do this is to show only the nodes close to the focus node. It also helps to provide users a way to filter the nodes. For example, if graphs have several types of links, users could choose to see only nodes connected by certain types of links.

It is important to note that only a few graph visualization systems have actually been tested with real users. And, there has been no study comparing a tree visualization of a graph with state of the art graph visualizations.

1.2 Three Systems

To address the above issues, three graph visualization systems were designed and developed; PaperLens, TaxonTree, and TreePlus. They enable users to interactively explore large graphs.

PaperLens is a tool developed to visualize conference proceedings. Common graph visualization systems provide visual overviews of the entire dataset that display a visual element for each entity instance. In contrast, PaperLens provides an abstract overview of the full dataset and shows relationships within a complex network through interactive highlighting. PaperLens was developed to visualize 8 years

(1995-2002) of InfoVis conference proceedings and was later extended to visualize 23 years (1982-2004) of the ACM SIGCHI conference proceedings. The PaperLens concept was also applied to food web data (EcoLens) and generalized to visualize any graph that can be represented by two entity types (NetLens).

TaxonTree is a tool developed to visualize the Linnaean classification for taxonomic names in the Kingdom Animalia. To support biodiversity data, we extended SpaceTree, a tree browser developed at the Human-Computer Interaction Lab (HCIL), which combines the node-link tree diagram with zooming and animation. Our qualitative study with 18 biology students provides further evidence for the value of interactive tree visualization and integrated searching and browsing in information retrieval and understanding.

Finally, the lessons learned from building these tools were applied to design a visualization for general graphs. TreePlus, the main contribution of my work, takes a tree layout approach to graph visualization. It transforms a graph into a tree plus cross links (the additional links not represented by the spanning tree), and uses visualization, animation and interaction techniques to reveal the graph structure while preserving readability of the labels. To support information gathering and detailed exploration of the graph's local structure, a guiding metaphor was used: "Plant a seed and watch it grow." Users start with a node and expand the graph as needed, complementing traditional overview techniques that are effective at revealing patterns and clusters. A controlled user study, which compared TreePlus with a traditional graph visualization, showed that the advantage of using TreePlus increases as the density of the displayed data increases. Participants also reported higher levels of

confidence in their answers when using TreePlus. And most of them preferred TreePlus as well.

1.3 Contributions

The main contributions of this dissertation are the:

- Creation and application of several design principles to various graph visualization domains. Tightly-coupled and highly customized views were used for graph visualization in a novel way. A new tree layout approach was proposed with appropriate visualization and interaction techniques. When visualizing graphs as trees, a guiding metaphor "Plant a seed and watch it grow" was used to support information gathering and detailed exploration of the graph's local structure.
- Design and implementation of PaperLens, a novel visualization system that enables users to reveal trends, connections, and activity throughout a conference community. The interface offers a compelling alternative to more common node-link diagram visualizations.
- Design and implementation of TaxonTree, a visualization system for animal classification. The extension of an existing tool enabled support for database access and web deployment, while accommodating domain-specific requirements.
- Qualitative evaluation of TaxonTree. The description of how users in the biodiversity domain approach information retrieval provides a rich

demonstration of the value of interactive tree visualization with integrated searching and browsing.

- Design and implementation of a new interactive graph visualization component called TreePlus based on a tree-style layout. This includes the development of special visualization and interaction techniques to efficiently visualize graphs as trees.
- Empirical evaluation of TreePlus. The controlled experiment comparing TreePlus to a traditional graph visualization shows that, in general, the advantage of TreePlus over the traditional interface increases as the density of the displayed data increases.
- Development of a taxonomy of tasks for graph visualization.
- Examples of the use of these techniques. The integration of TreePlus with the next generation of PaperLens will offer great potential to the field of information visualization.

1.4 Dissertation Organization

The remainder of this dissertation is organized as follows. Chapter 2 describes related graph visualization work and Chapter 3 presents PaperLens. Chapter 4 introduces TaxonTree, while explaining the design decisions and the lessons learned from the development and evaluation processes. Chapter 5 provides a detailed description of TreePlus. Chapter 6 presents the taxonomy of tasks for graph visualization. This chapter also defines graph specific objects and demonstrates how complex tasks could be decomposed into a series of low-level tasks performed on

those objects. Chapter 7 presents three applications which use TreePlus and describes how these applications interact with TreePlus to accomplish sample tasks. Finally, this dissertation concludes with future work in Chapter 8.

Chapter 2

Related Work

Graph visualization has been studied extensively over the last few decades and has gradually been posed as a distinct sub-field of information visualization [26, 66]. Previous work on graph drawing is scattered through the computer science literature [34, 44, 48, 96, 138], including a book devoted to graph drawing [35]. Furthermore, there are several websites showing network visualization examples from various fields [6, 134] and several systems for generating such visualizations and performing statistical analyses of social networks, such as JUNG [79] and GUESS [3].

Many graph drawing algorithms have been developed to produce aesthetically pleasing graphs. To validate the aesthetic principles used by those algorithms, Purchase conducted several user studies [110, 111, 112] on small graphs. In addition, some researchers have tried to visualize a graph as a tree, which is more tractable and easier to understand [38, 63, 98, 99, 100]. Despite the vast amount of research, only a few graph visualizations have actually been tested with real users. Furthermore, there has been no research to compare these two approaches.

2.1 Graph Layout

If the data to be visualized have inherent relations among them, they can be represented as nodes of a graph, with edges representing the relations. The basic graph drawing problem can be defined simply as: given a set of nodes with a set of edges, calculate the position of the nodes and the curve to be drawn for each edge.

However, not all graph layouts are developed with interaction in mind. Many classical graph drawing algorithms are used only to produce a static view of a graph. Since this dissertation is not about graph drawing itself, we review related works from an information visualization point of view. Hence, theoretical proofs or graph properties such as planarity are not a central issue in this proposal.

2.1.1 Understanding of Graphs

What makes a graph 'understandable?' How is it possible that one graph can be objectively more understandable than another graph? Designers of graph drawing algorithms claim that graphs having a layout that optimizes certain qualities are easier to understand. The attributes that define a good graph are called aesthetics and were determined through statistical analysis in [12], [112], and [127]. Some commonly adopted aesthetics include:

- Display the symmetries of the graph.
- Minimize the number of crossings between edges.
- Minimize the number of bends along the edges.
- Maximize the smallest angle between two edges incident on the same vertex.
- Minimize the sum of edge length and the maximum length of an edge.
- Minimize the area of the drawing by producing a compact graph.

The list above shows that some aesthetics conflict with each other. One aesthetic calls for of minimizing the number of crossings between edges and other calls for maximizing the symmetry. These two aesthetics conflict with one another because in some cases it would be best to decrease the symmetry in order to have a

smaller number of crossings. Furthermore, most of the above aesthetics are computationally hard. For example, minimizing crossings is NP-hard [57]. Even if the aesthetics do not conflict, it is often difficult to handle all of them at the same time. Hence, algorithm designers may need to compromise among more than one aesthetic.

Despite the considerable effort that has gone into constructing algorithms to optimize according to these aesthetics, surprisingly little work has gone into the empirical validation of these aesthetic principles. Purchase [110] compared task performance on five pairs of graphs that were designed to differ according to the aesthetic principles of edge bends, edge crosses, maximizing the minimum angle, orthogonality, and symmetry. This study demonstrated that reducing the number of crossings is the most important aesthetic, while maximizing symmetry and minimizing the number of bends are less important. The effects of maximizing the minimum angles between edges from a node and of putting edges and nodes on an orthogonal grid were not statistically significant.

When navigation is involved, to help users preserve a stable mental model of the graph, it is important and necessary to make the results of the layout algorithm predictable. Two different runs of the algorithm for the same or similar graphs should not lead to radically different visual representations.

2.1.2 Force-Directed Layout Methods

Force-directed algorithms [35, Chapter 10] are very popular for general graph layout. They consist of two main parts: 1) A physical model for the graph, provided by a force system defined by the vertices and edges. 2) An algorithm to find an

equilibrium state of the force system. They are simple and easy to understand due to their physical analogy. Furthermore, they often produce good layouts especially for showing clusters even though they are relatively simple to code.

The simplest force-directed method uses a combination of spring and electrical forces. This algorithm, called spring embedder [40], simulates a mechanical system, where rings replace vertices and springs replace edges. The springs attract the rings if they are too far apart, and repel them if they are too close. Nodes are seeded in an initial position often randomly. The system repositions nodes in order to minimize the energy of the system. Under certain assumptions, the result of spring embedder tends to be symmetric [43].

There has been a vast amount of work to revisit and improve this class of algorithms. For example, Fruchterman and Reingold refined the algorithm for uniform edge lengths [51]. Some systems use methods such as gradient descent [80] or simulated annealing [33] to search for the desired configuration. The force simulations in prefuse uses the Barnes-Hut algorithm [11] to compute anti-gravity force between nodes in $O(|N|\log|N|)$ time rather than $O(|N|^2)$, where $|N|$ is the number of nodes [65].

In general, however, force-directed algorithms are slow since all pairs of nodes in the graph must be visited in each iteration. The quality of the result often depends on the number of iterations. This makes the systems using force-directed methods not scalable. The Gem system, one of the best variants, can handle graphs of up to 256 nodes in about 70 seconds, and is estimated to work with a time complexity of $O(|N|^3)$, where $|N|$ is the number of nodes [49]. Furthermore, when they are

applied to graphs with labeled nodes, the resulting layouts suffer from severe node occlusions [56].

Scalability is not the only problem with force-directed systems when considered from an information visualization perspective. The final visual appearance of a dataset is usually different on each invocation of the system, either because the initial node positions are random or because the minor tweaking of layout parameters results in major changes in the final layout.

It might be possible to make force directed layouts completely predictable by imposing some rules to decide the initial node positions. For example, Bertault has developed an interactive graph drawing algorithm for undirected graphs, based on a force–directed approach, preserving edge crossings [16]. However, the algorithm requires an additional step to find a planar drawing for the graph. Another naive way to make the layout predictable for labeled graphs is to evenly distribute nodes in a grid, in alphabetical order. While it would achieve predictability, there is no guarantee that it would perform better than the random case.

2.1.3 Tree Layout

Trees, strict hierarchies, are a subset of general graphs and have been investigated extensively. Tree layout is a more tractable problem than general graph layout. Supowit and Reingold [128] have proved that it is possible to find a tree layout in linear time. Furthermore, tree layouts are usually predictable.

There are two main categories of solutions to display and manipulate trees: node link techniques and space filling techniques. The algorithm by Reingold and Tilford [114], revisited later by Walker [136] is one of the well-known node link tree

layout techniques. It produces a classical tree drawing (shown in Figure 2.1) in the sense that the drawing clearly represents the inherent hierarchy of the data. It is simple, fast, and completely predictable. It can be adapted to produce top–down as well as left–to–right tree layout.

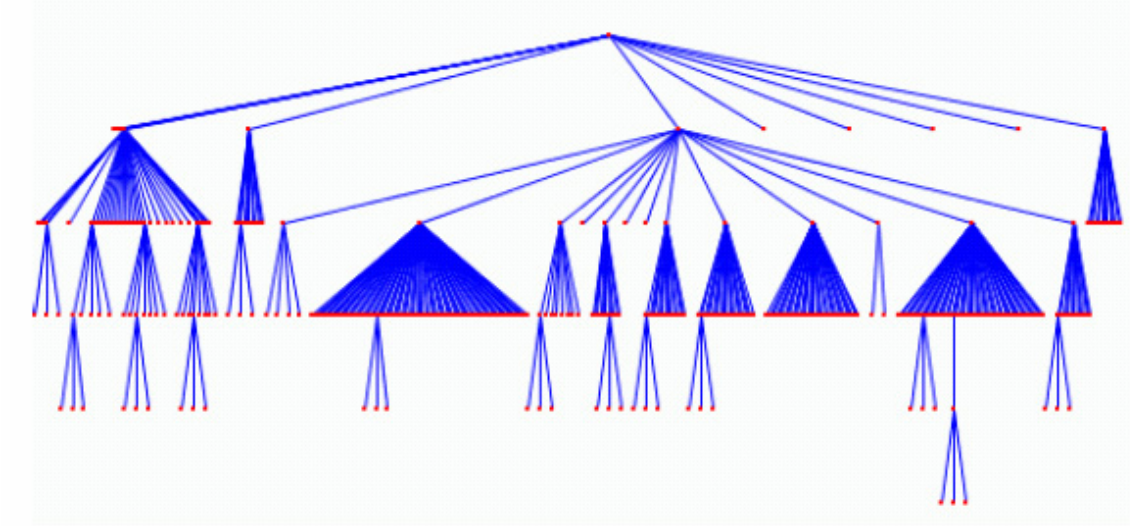
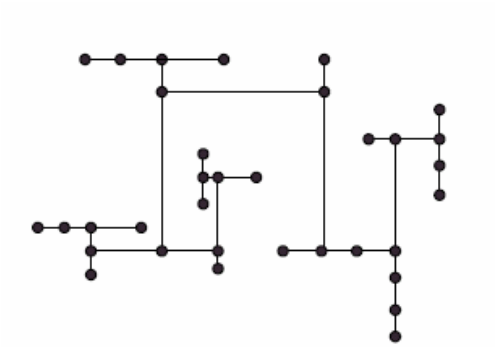


Figure 2.1 Classical tree drawing

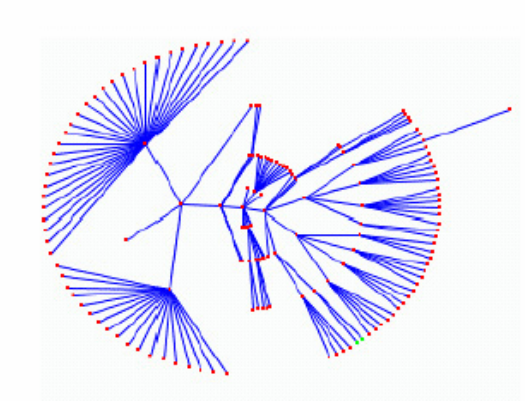
H–tree layouts (Figure 2.2a) are classical representations for binary trees [122] which only perform well on balanced trees. Eades proposed a radial view (Figure 2.2b), a variation of the algorithm that behaves well in general [41]. Nodes are placed on concentric circles according to their depth in the tree. The root of the tree lies on the center. The children of the root lie on the smallest inner ring, and their children lie on the second smallest ring, and so on. The angular position of a node on its ring is determined by the sector of the ring allocated to it. Each node is allocated a sector within the sector assigned to its parent, with size proportional to the angular width of that node's subtree. The main difference between the H-tree and radial positioning and the Reingold and Tilford's tree is that it is less clear where the root of

the tree is. This may cause users to explore the graph in a different way (e.g. non-hierarchical fashion). A balloon view (Figure 2.2c) can be obtained either by projecting the cone tree structure onto the plane or by computing the nodes' position directly.

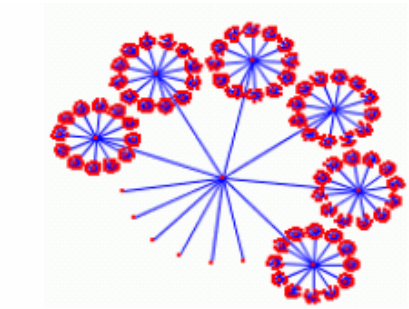
Node link diagrams, however, typically make inefficient use of screen space, wasting the root side of the tree and cluttering the opposite side. Space filling techniques such as treemaps [78] (Figure 2.2d) and information slices [7], on the other hand, make full use of screen space. Treemaps are a visualization method for hierarchies based on enclosure rather than connection [78]. Since they have been successful at visualizing trees that have attributes values at leaves, they are useful when users are mostly interested in leaf nodes and their attributes, but do not care about the topology of the tree. Because of the unfamiliar layout, it takes time for users to learn how to perceive the structure in treemaps. Cushion treemaps [133] has been developed to provide insight in the hierarchical structure by adding shading as an extra cue.



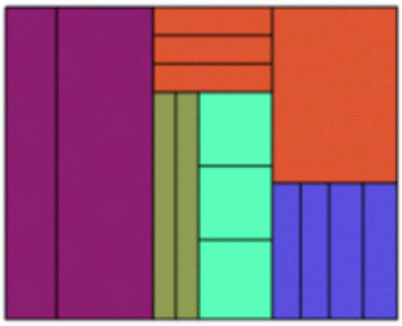
(a) H-tree layout



(b) Radial View



(c) Balloon view



(d) Treemap

Figure 2.2 Tree layouts

Some research has been done on multiple hierarchies. Time Tube [30] shows changes of a single tree over time. Multitrees [53], introduced by Furnas and Zacks, are a class of structures for information access and reuse. They are large directed acyclic graphs and unions of trees that share subtrees. In other words, they consist of several different hierarchies with the same set of nodes. They have an interesting property that the sets of ancestors and the sets of descendents of any node are both trees. Polyarchies [117], on the other hand, are multiple intersecting hierarchies. They share nodes rather than subtrees.

2.1.4 3D Layout

Another technique is to visualize graphs in 3D instead of 2D. Researchers advocating 3D believe that the extra dimension would give more space and it would alleviate the problem of displaying large structures.

One simple approach is to generalize 2D layout algorithms to 3D. For example, Rekimoto implemented Information Cube [115], a 3D version of nested boxes. The Information Cube nested children cubes inside their parent cubes to represent parent-child relationships. It displays textual labels on semi-transparent cube surfaces. Furthermore, most force-directed methods can be generalized to 3D. While they are simple, it is difficult to find the best view in 3D space [42].

There are other layout algorithms developed directly for 3D. SemNet [46], the first 3D graph visualization, was developed to support exploring and manipulating large knowledge bases represented as directed graphs. To manage complexity, SemNet makes use of fisheye views. Three types of information were used to determine the position of elements. First, it uses mapping functions from the properties of the element to its position. Second, it uses the connectivity between two elements to position them adjacent. Third, it allows users to position elements based on information that is not represented in the knowledge base.

Cone trees [118] (Figure 2.3) are one of the best-known 3D tree layout techniques in information visualization. Nodes are placed at the apex of a cone and its children are placed evenly along its base. They allow users to rotate a 3D representation of the tree to reveal its hidden parts. Cone trees are evaluated and revisited to refine the algorithm [28, 31].

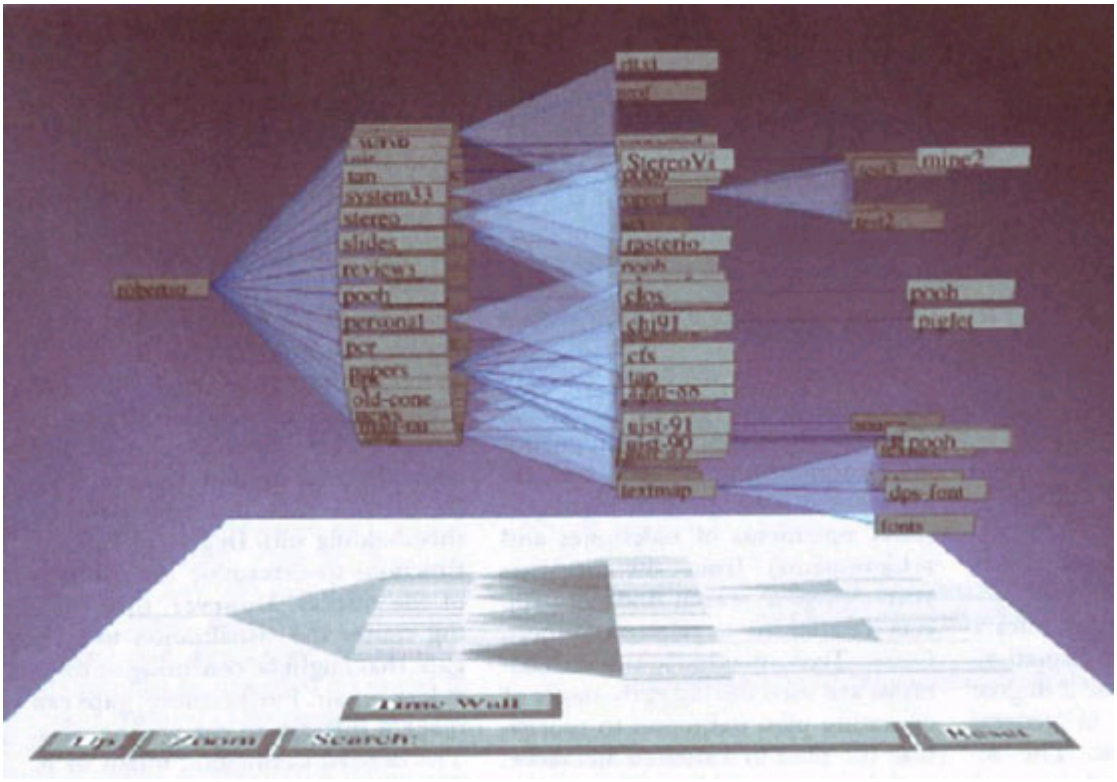


Figure 2.3 Cone tree

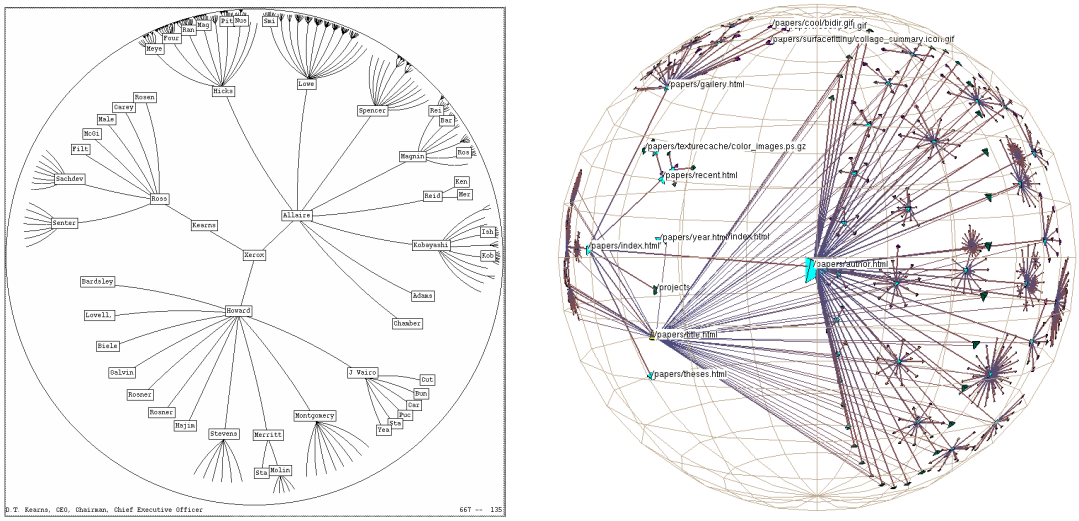
Another benefit of 3D visualization is human's familiarity with 3D in real world. The File System Navigator of SGI Workstations is one that uses a landscape metaphor. While it used a simple planar layout for laying out the file structure, it added 3D blocks on the plane to represent files. Block's size is proportional to file size. Users can fly over the virtual landscape created by those blocks.

While 3D representations are attractive, they only marginally improve the screen space problem. 3D graph visualization techniques have inherent difficulties such as occlusion of the objects and the complexity of the interaction. Discrepancy of using 2D screens and input devices to interact a 3D world is another problem. Furthermore, it is often combined with missing stereo and motion cues, which was proved to be very important by Ware and Franck [137].

2.1.5 Hyperbolic Layout

There have been a number of research to exploit hyperbolic geometry, one of the most useful non-Euclidean geometries. In the hyperbolic geometry, parallel lines diverge away from each other. This means that the circumference of a circle grows exponentially with its radius. Hence, exponentially more space is available with increasing distance.

Lamping and Rao first introduced the hyperbolic tree browser [86] (Figure 2.4a), which solves the occlusion problems of Cone Trees. It lays out the entire hierarchy on a hyperbolic plane and maps the plane onto a circular space. This results in the effect of fisheye technique. The hyperbolic browser effectively handles arbitrary large hierarchies. It provides smooth and continuous animation of the tree as users click or drag nodes to readjust the focus point of the layout. While the animation is striking, users may be confused because the shape of the tree changes. Labels are hard to read because they are not aligned and sometimes overlap.



(a) Hyperbolic tree browser (b) H3

Figure 2.4 Hyperbolic layouts

The H3 layout algorithm [98, 99, 100] by Munzner is an attempt to merge ideas from hyperbolic browsers and Cone Trees. H3 (Figure 2.4b) projects the graph onto the surface of a sphere with the focus node at the center of the sphere. It is probably known to be the best scalable and interactive graph visualization tool, displaying 20,000 nodes.

2.1.6 Circular Layout

A circular layout produces a presentation that resembles interconnected ring and star network topologies. This layout is included in several graph visualization toolkits, such as the Graph Layout Toolkit [37] and JUNG [79]. Figure 2.5 shows the connections between pairs of people in an online newsgroup in a circular layout.

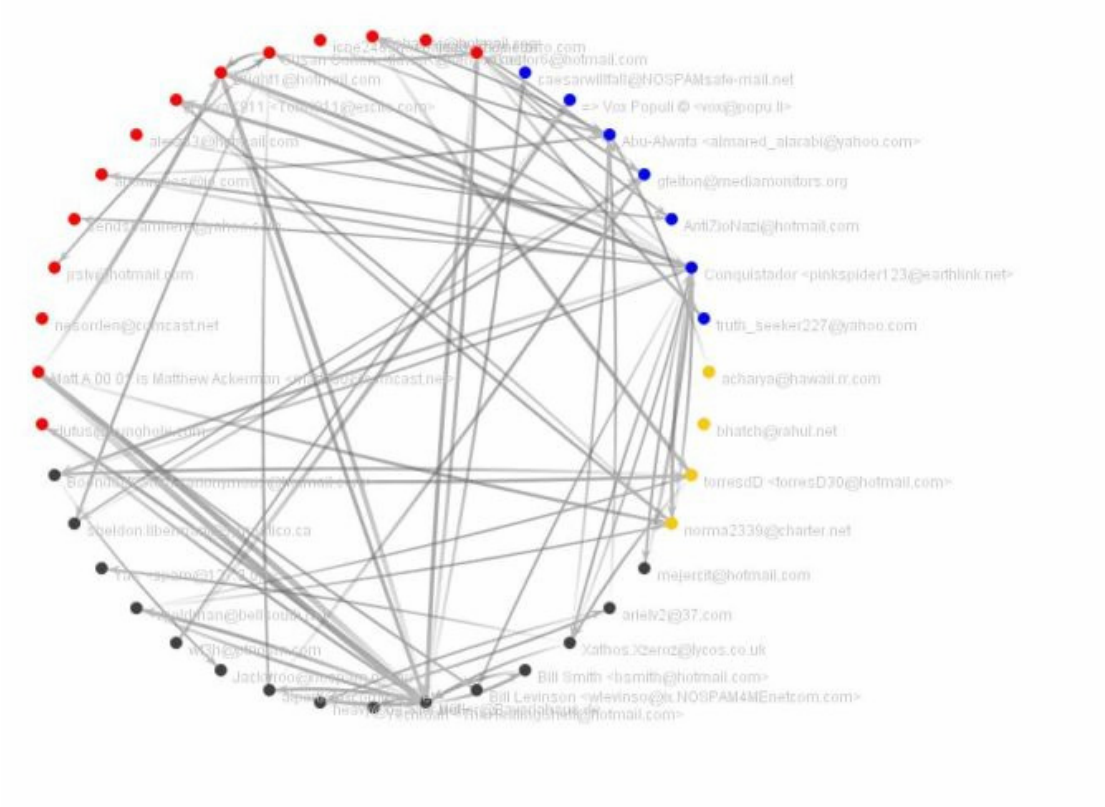


Figure 2.5 JUNG visualizes the connections between pairs of people in an online newsgroup using a circular layout.

Otter [70], a general-purpose network visualization tool, provides two options for root node placement: circular and coordinate-based. For the circular layout, Otter places nodes along the circumference of a circle. Osprey [23], a visualization for complex interaction networks, represents genes as nodes and interactions as edges between nodes. It provides several network layouts, including circular, concentric circles, spoke and dual ring.

2.2 Navigation and Interaction

Since none of the graph layout algorithms is free from the problems raised by the large size of the graphs, navigation and interaction facilities play a very important role in graph visualization.

When the graph structure is too large to see in detail all at once, the most straightforward solution is to allow users to zoom and pan the visible area. There are two types of zooming: geometric and semantic zooming [15]. Geometric zooming simply scales up or down the graph content. In contrast, semantic zooming scales data in non-geometric ways. It provides different information content depending on zoom factor. The difficulty of semantic zooming lies in assigning an appropriate level of detail.

While zoom and pan are conceptually simple, they introduce problems when used in an interactive environment. The disadvantage of simply providing navigation controls is that users often lose track of where they are within the global structure. Adding an overview window showing where the current view port is can provide some guidance. However, these visualizations require extra space for the overview and force users to continually switch between the two views and reorient themselves. Hornbæk and Frøkjær [69] showed that the performance of an overview plus detail interface was approximately 20% slower than that of a linear interface for question-answering tasks. Furthermore, the adjacent parts of the current view are not visible at all in the current view.

A large class of visualization techniques called focus plus context has been developed to address this problem by attempting to smoothly integrate detail views

with as much surrounding context as possible, so that users can see all relevant information in a single view. They complement zoom and pan instead of replacing them. One of the well-known focus plus context techniques is fisheye views. The fisheye view shows an area of interest large and in detail and show remote regions successively smaller and in less detail. These are reviewed by Leung and Apperley [90].

The fisheye technique is originally defined as a separate processing step on the graph layout algorithm. While this makes it possible to do modular programming and applying fisheye is much faster, the distortion may destroy aesthetics generated by the layout algorithm. One way to tackle this problem is to merge the distortion and the layout algorithm as hyperbolic layout does. Sarkar and Brown generalized the fisheye view to 2D graph layout [119, 120]. They computed the position, size, and level of detail of objects based on client specific functions of the object's distance from the focus and the importance of nodes pre-assigned by a prior knowledge. To allow users to simultaneously concentrate on several important areas of the graph, some researchers also extended it to handle multiple foci [82, 83].

Similar effects can also be achieved by using 3D techniques. When objects were put on 3D, the perspective or parallel projections create a distortion on 2D screen. The Vitesse system [102] applied a planar or polar transformation onto X and Y axes to achieve the distortion effects. Carpendale *et al*. have used more complex surfaces [27]. Other 3D visualization techniques such as the perspective wall [92] are other examples.

2.3 Visualizing Graphs as Trees

Trees have many advantages over graphs. It is possible to lay out trees nicely in the plane in polynomial time. Trees are also easy to understand because they better support abstraction and aggregation. To the contrary, it is extremely difficult to layout large graphs so that people can understand them. Hence, tree visualization can often be used to view graphs with some preprocessing. A number of researches have tried to visualize graphs as trees.

WebMap [38] visualizes a topology representation of the user's navigation history. If a webpage is accessed for the first time, a new topology node is created and connected with its predecessor through a primary link. Otherwise, if the document has been visited before with a different access path, a cross link is created to connect the node with its predecessor.

Hao *et al*. visualized large highly connected hierarchies in a hyperbolic space using an "invisible link" technique with a placeholder [63]. To avoid cluttering, only the primary links are shown to users. All other cross links are invisible to users and hidden as a property of a node until users access the node.

Munzner introduced a class of graphs called quasi-hierarchical graphs, which can be visualized using a spanning tree [98, 99, 100]. A graph is regarded as a quasi-hierarchical one if there is a decision procedure for selecting the preferred parent link from among all the incoming links to a node. H3 Viewer visualizes a minimum spanning tree through a graph with weighted edges, where domain-specific information is used to compute the weights. Trees and directed acyclic graphs

(DAGs) are trivially quasi-hierarchical. Graphs that are considerably denser than trees such as the hyperlink structure of the web can also be quasi-hierarchical.

Latour [67] (Figure 2.6) was first primarily developed for tree visualization. Later it was extended to handle more general structures such as directed acyclic graphs. Latour added the additional links to the spanning tree, given either from the application or automatically generated from the DAG.

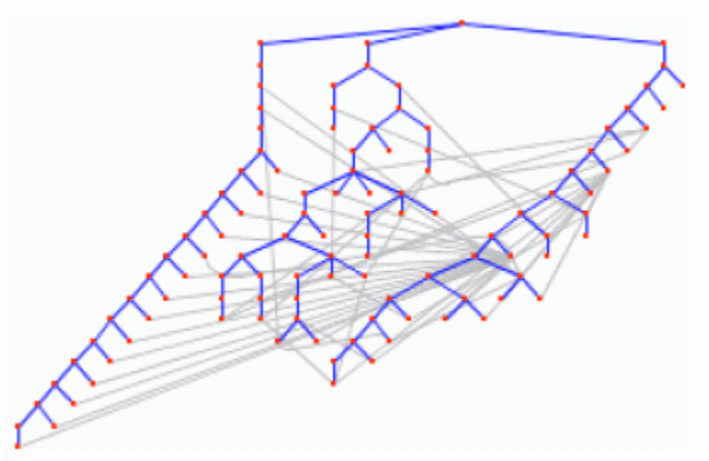


Figure 2.6 A tree with added links

Boutin *et al*. used a tree-like graph as a link filtering mechanism [20]. They first extracted a spanning tree and then added some cross links to extract dense components for clustering. Force-directed algorithms were used to lay out the tree-like graph. As is often the case with other overview approaches, node labels were ignored.

Yee *et al*. have developed an interactive exploration tool for graphs by using a radial tree layout method (Figure 2.7) [143]. They animated the transition to a new layout when users select a new focus node. The system linearly interpolates the polar

coordinates of the nodes to help user follow the transition. It was applied to the Gnutella network and social networks.

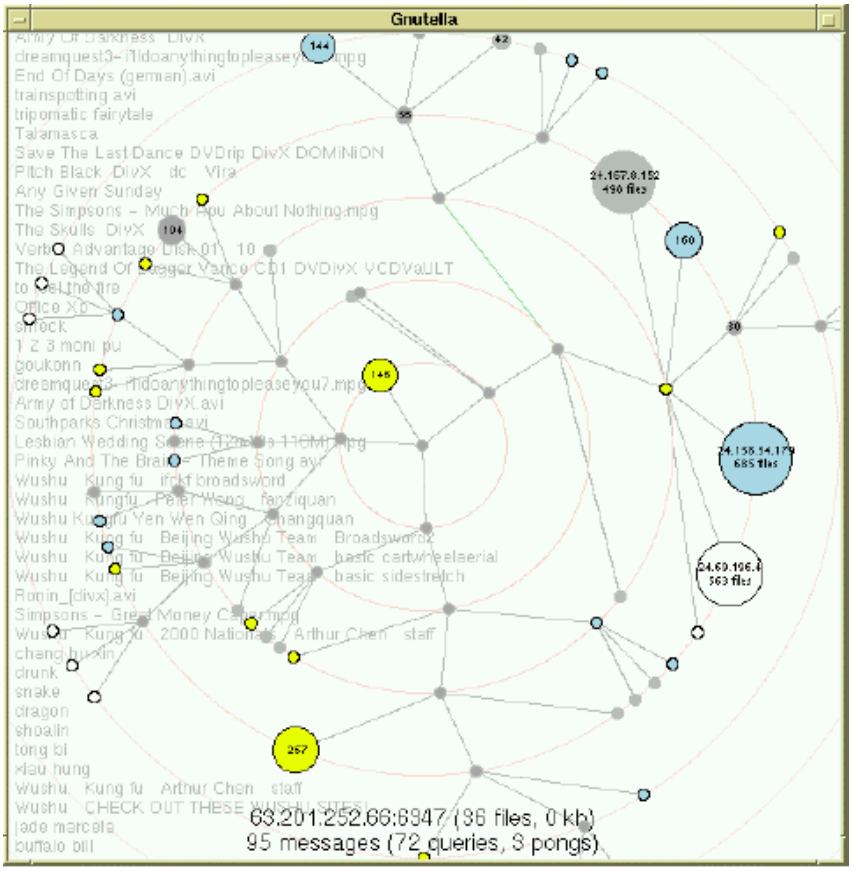


Figure 2.7 Visualization of the Gnutella network using a radial tree layout.

MoireGraphs [77] also used a radial layout to display a spanning tree of a graph. It was mainly designed for a specific set of graphs – graphs that have nodes with visual elements such as images. Focus+context technique was used to provide an overview of graphs. MoireGraphs combines a number of interaction techniques including radial rotation and secondary foci.

Boutin *et al*. also used a radial layout to visualize a hierarchical clustered graph that is the result of multilevel clustering [19]. It is a graph of clusters, where

each cluster itself is hierarchically clustered. Their transformation from a graph into a spanning tree ensures that there is no over-lap between clusters.

Many visualization systems for visualizing RDF data or ontologies have used tree layout approach. For example, OntoRama [45] enables users to browse a knowledge base (ontology) in a hyperbolic layout. Most of them duplicated nodes several times to support cross links. However, the EROS system [135] (Figure 2.8) combines the tree layout and graph layout approaches to preserve the good aspects of both approaches. The main idea behind this interface is to consider RDF properties as a mapping that relates some elements from the class hierarchy into other elements within the same hierarchy. It displays two identical trees: the left tree being domain and the right tree being range. It represents properties as arrows connecting the classes from the left tree to the classes from the right tree with a label.

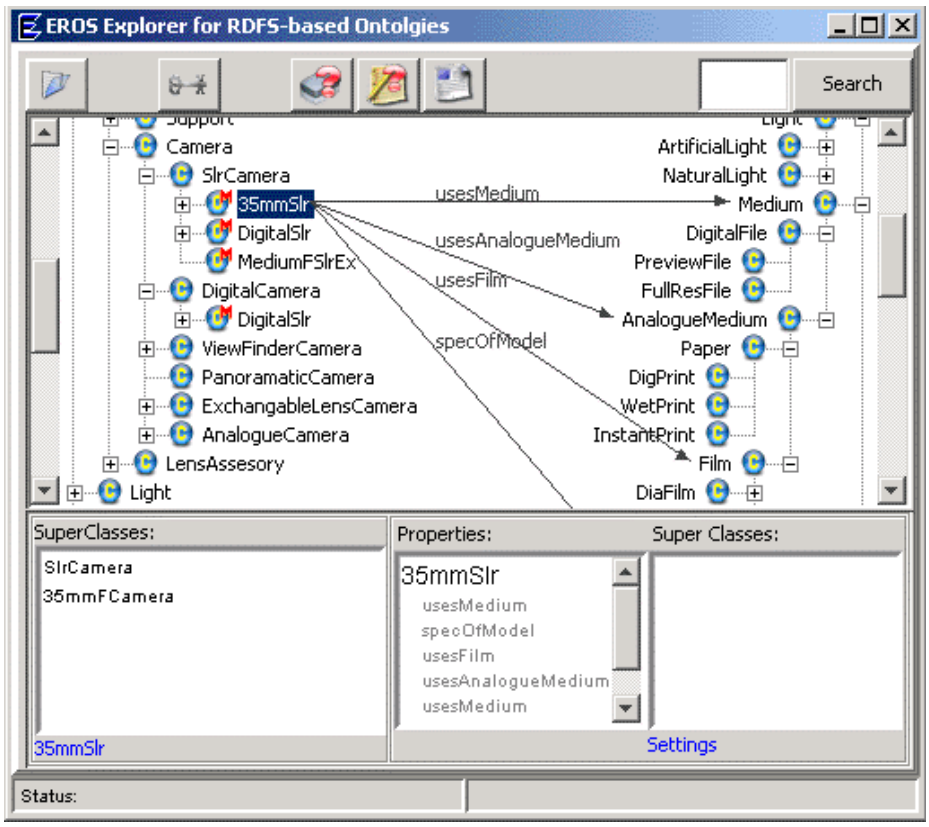


Figure 2.8 Explorer for RDFS-based Ontologies

In addition to node link layout, space filling layout was also used to visualize graphs. Treemaps are appropriate when showing the attribute value distributions is more important than showing the graph structure [78]. Fekete *et al*. displays the tree structure of a graph on Treemap and overlays the cross links as curved links on top of the Treemap [47]. The curved link is modeled using a quadrilateral Bézier curve and the offset of curvature indicates the direction of the link. Treemaps have also been extended to visualize genomic data [9]. Nodes were duplicated to support gene ontologies, which are directed-acyclic graphs.

2.4 Visualizing Graphs as Matrices

Another approach to graph visualization is to use a matrix-based representation (Figure 2.9). A graph may be represented by its connectivity matrix which is a matrix of Boolean values whose rows and columns represent the nodes of the graph. When two nodes are connected, the cell at the intersection of the corresponding row and column contains the value "true." Otherwise, it contains the value "false." Boolean values may be replaced with valued attributes associated with the links that can provide a more informative visualization. Abello and Korn presented matrix and color map based techniques to visualize phone calls made between states [1]. Van Ham used multilevel call matrices in the management of large software projects [132]. He argued that matrix-based visualizations have a number of advantages over traditional node link diagrams when users are more interested in links than in nodes. Matrix Browser [144] allows users to select and filter hierarchical substructures for the low and the column (Figure 2.10). A common tree widget is used to represent those hierarchies. VistaClara [84] is a visualization system that applies an extended permutation matrix to the task of exploratory data analysis of multi-experiment microarray studies. Ghoniem *et al*. used adjacency matrices to interactively visualize and explore relations between constraints and variables in constraint problems [61]. The benefits of adjacency matrices were shown for graphs with thousands of nodes. However, it might be difficult to follow links while reading labels.

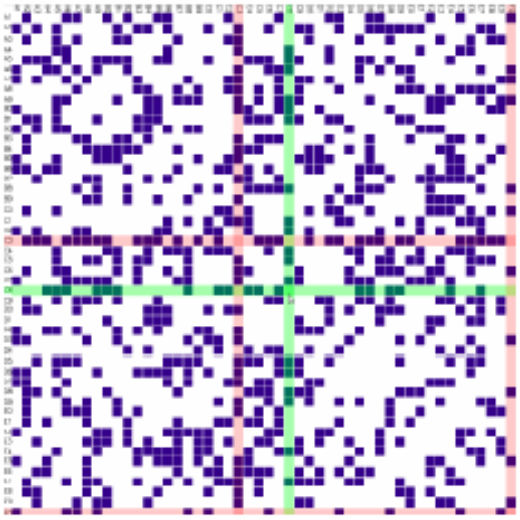


Figure 2.9 Matrix representation of an undirected graph with 50 nodes and 400 links

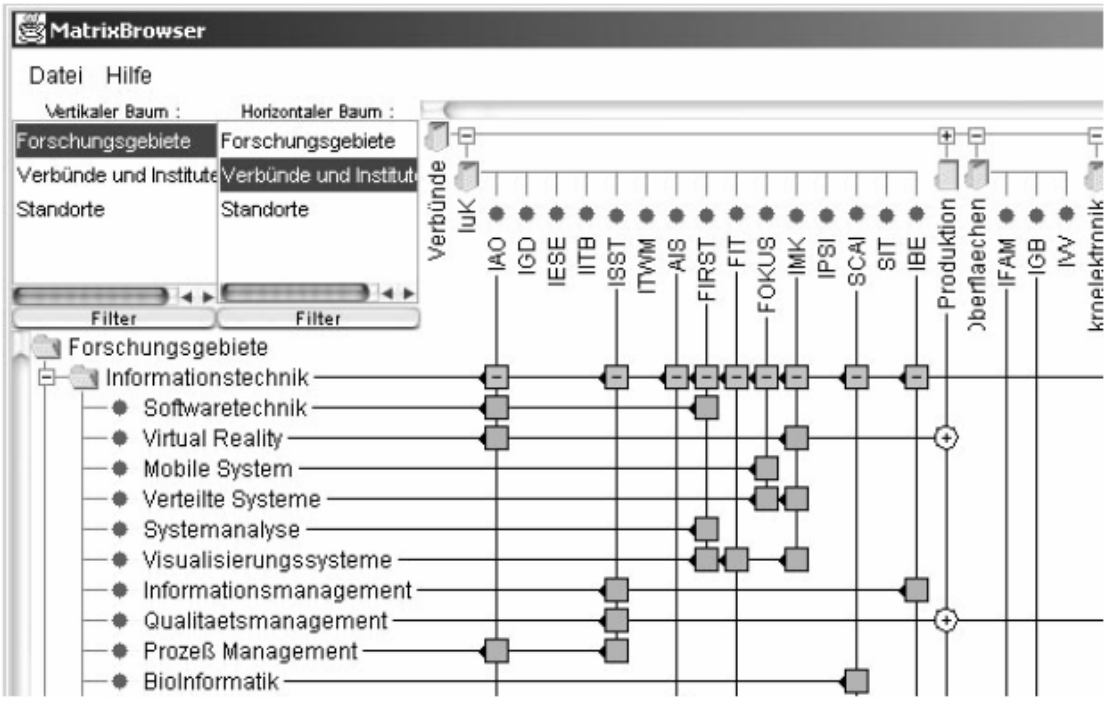


Figure 2.10 Matrix Browser allows users to select and filter partial hierarchies.

2.5 Evaluation

Many graph visualization techniques have been proposed over the last 20 years under the assumption that our perceptual system can be used to help see patterns in information. Many papers describe techniques that intuitively solve some problems. However, intuition is not always right even though it is essential for design insights. Furthermore, we often got excited just because a visualization looks cool. When a new visualization is introduced, some evaluation should also be provided to prove that the intuition is correct or the cool effect is useful and usable.

Information visualization systems can be evaluated by focusing on the set of tasks for a group of target users in a particular domain. Methods from user-centered design [97] and ethnography can help the visualization developers understand users' needs to identify their goals. However, these goals are often too high-level to be directly addressed by visualization systems. Therefore, they need to be broke down into several low-level tasks [97], which enable us to evaluate the effectiveness of visualization systems.

Evaluations range from informal usability observations in an iterative design process to formal controlled experiments designed to gather statistically significant results. It is often investigated whether performance improved for a particular task. Another evaluation method is to use qualitative field tests or case studies. One of the main limitations of these methods is that the result cannot be duplicated. In other words, the conductor may not get the same results in a different situation. However, they can be used to show the usefulness of a visualization system in a real-world environment.

Wiss *et al*. evaluated three designs for 3D information visualization [141]. They used two datasets: the table of contents of an electronic newspaper and a part of a file system. They chose seven high level tasks from the taxonomy of tasks by Shneiderman [123]. Three visualization tools have been compared with respect to their suitability for different datasets and their support for tasks.

Risden *et al*. conducted an empirical evaluation of three different interfaces for web contents by using the snap.com hierarchy contents [116]. Two of the interfaces were conventional 2D browsers and the third one was 3D hyperbolic interface. While the study demonstrated the strengths and weaknesses of those three interfaces, there were no significant differences in overall user satisfaction across them.

Cockburn and McKenzie provided an empirical evaluation on the usability of cone tree interfaces [31]. Although many subjects liked the cone tree interface, they were significantly slower at locating named files when using the cone tree interface than when using a normal tree interface.

Plaisant *et al*. conducted a controlled experiment to compare SpaceTree with Microsoft Explorer and Hyperbolic browser [109]. They showed that SpaceTree works better than others for estimating overall tree topology and revisiting previously visited nodes. SpaceTree was found more attractive than Explorer.

Ghoniem *et al*. conducted an evaluation to assess the readability of two representations of graphs: matrix-based representations and node-link diagrams [60]. They used seven generic tasks. The study suggests that when graphs are bigger than

twenty vertices, the matrix-based visualization performs better than node-link diagrams on most tasks. Node-link diagrams were favored only for path finding task.

Since there is no task taxonomy and benchmark datasets for graph visualization, it is difficult to generalize results collected from several experiments. In other words, there is no easy way to compare tools that are not directly compared in the same study. The Information Visualization Benchmark Repository [73] was created to improve the evaluation of information visualization techniques and systems by providing benchmark datasets and tasks. There needs to be a list of tasks for graph visualization that has enough detail and specificity to be useful to designers who want to improve their system and to evaluators who want to compare graph visualization systems. Furthermore, it would be useful to characterize graph visualization tools based on which objects (nodes or links) and tasks they focus on, and which graph characteristics they are strong at.

Chapter 3

PaperLens: Understanding Research Trends in Conferences

Interfaces for visualizing search results for a digital library, such as Envision [103], exist, but we do not yet have a visualization system that allows researchers in our field to understand how researchers, topics, and outside research sources interact and influence research activity in general. Hence, Smeaton *et al*. [125] performed a content analysis of papers published in SIGIR proceedings to understand research trends. Their focus was to determine what topic areas appear but not to visualize the results. They also did not include any citation analysis.

In practical terms, it is not possible to answer interesting questions with current systems such as: Which topics have come and gone over the last 23 years of CHI? What is the relationship between a given set of researchers? The IEEE InfoVis 2004 Conference chose to pose these kinds of questions about its history as the theme of the InfoVis 2004 Contest [75]. To address these questions, I developed a visualization called PaperLens. It allows researchers to see trends and topics in a field, in addition to influential papers and authors, all within a single screen visualization (Figure 3.1). PaperLens was developed while I was doing an internship at Microsoft Research in 2004. I worked with Mary Czerwinski, George Robertson, and Benjamin B. Bederson.

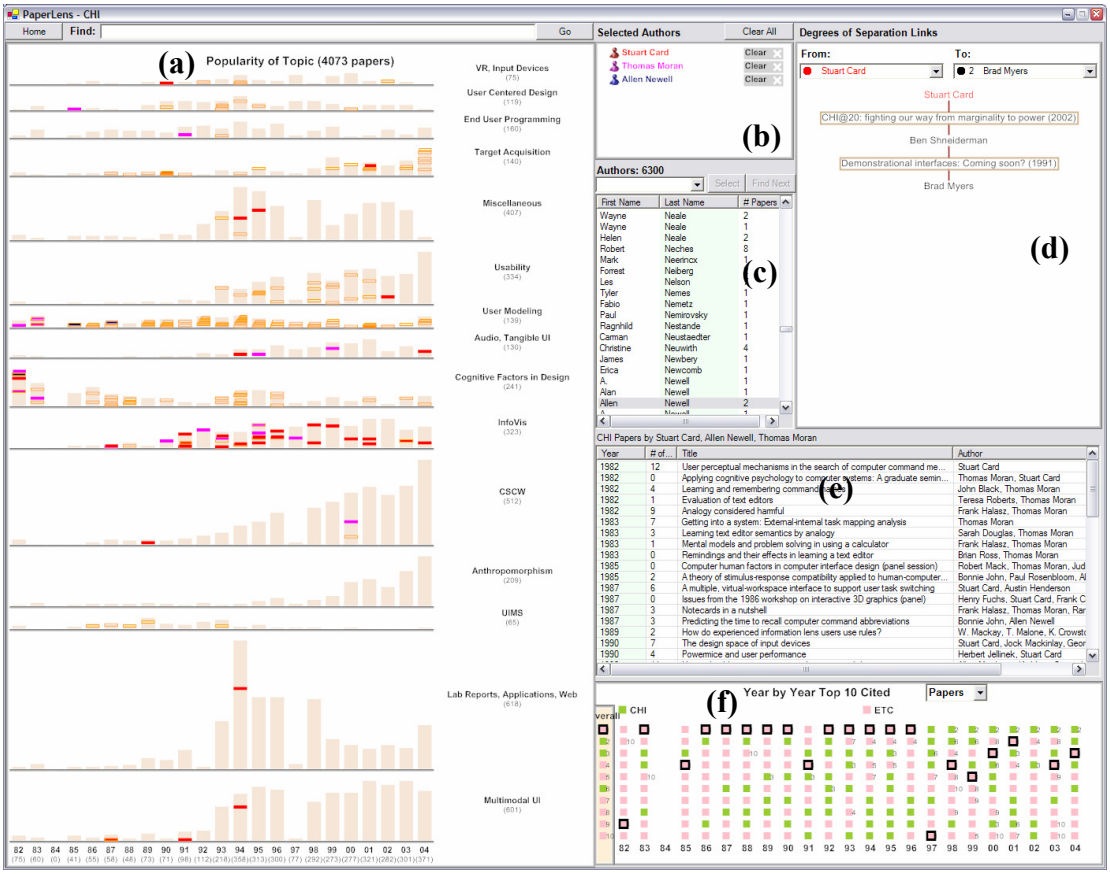


Figure 3.1 PaperLens tightly couples views across papers, authors, and references: (a) Popularity of Topic (b) Selected Authors (c) Author List (d) Degrees of Separation Links (e) Paper List (f) Year by Year Top 10 Cited Papers/Authors

When confronted with the problems described in the InfoVis contest, our first thoughts were to build some kind of node-link graph visualization tool such as Gem-3D [24], dot [55], H3 [98, 99, 100], or NicheWorks [139, 140] that shows all the data and all the relationships at once. In fact, nearly every submission to the contest used node-link diagram displays. As participants in the contest, we produced many such visual displays and still believe that they have their place, but, node-link diagrams typically do not scale well and produce cluttered overviews with few readable labels.

Thus, they have difficulties supporting even simple tasks such as reviewing the authors that cite some other author. Furthermore, there is no efficient way to show the trends of topics with these graph visualizations.

We instead opted for a simpler design with an abstract overview of the full dataset but not with all the relationships visible. The design was oriented around several small and simple tightly-coupled views which, together, provide users with powerful capabilities. While these design ideas have certainly appeared before, PaperLens brings them together in a unique fashion to address an important problem of concern to HCI researchers.

3.1 Visualization of Digital Libraries

Online digital libraries such as the ACM Digital Library (DL) [2] and IEEE Xplore [71] provide broad bibliographical and full-text access to journals and conference proceedings. The ACM DL shows which papers cite or are cited by a particular publication and lists all colleagues who have ever published with a particular author. This enables users to easily find related papers and authors. But, it is often difficult to reconstruct navigation paths and remember how a particular paper/author was found.

A few digital libraries provide some simple, statistical facts such as the most frequently cited papers/authors. However, simple analysis often requires extensive navigation and effort since the results are typically shown as a long list. Butterfly [91] combined search and browsing to visualize citation graphs. Envision [103], a digital library augmented with a flexible user interface, facilitates examining very large datasets and helps users discover patterns in the data. Shneiderman *et al*. developed a system to visualize thousands of search results at once on a two-

dimensional display that use categorical and hierarchical axes [124]. The system allows users to see the entire result set and browse it. Galaxy and ThemeView introduce visualizations of themes in document collections [142]. Börner and Chen describe motivation and usage of visual interfaces to digital libraries in [21]. They also discuss major challenges and review successful commercial systems.

3.2 Data Analysis

3.2.1 Two Datasets: InfoVis and CHI

The InfoVis 2004 Contest chairs provided the dataset containing metadata for 8 years (1995-2002) of InfoVis conference papers and their references. 315 authors published 155 papers in the InfoVis symposia. The metadata included author name(s), paper title, year of publication, and references for each paper. Some papers also had keywords, abstracts, references, and links to original papers.

The contest chairs collected all the available InfoVis publications and extracted their references by hand. They then found the referenced articles (if available) in the ACM DL [2] and collected the metadata for the referenced articles (if available) from the ACM DL. Finally, they put everything together in one XML file. After the contest chairs released the dataset, other researchers helped them clean up the dataset. For example, a duplicate authors' map was provided.

Two well-known sources of public citation information, CiteSeer [121] and ParaCite [104], automatically extract references from PDF files through complex heuristics implemented as Perl scripts. However, these scripts are unreliable and fail

to work on PDF files containing bitmaps, such as the proceedings of InfoVis from 1995 to 1997. Therefore, the references had to be extracted by hand.

Once the InfoVis data was visualized, ACM kindly provided the dataset containing metadata for 23 years (1982-2004) of CHI papers. Though the data for the papers published in 1984 is missing, the remaining dataset included not only full papers but also short papers, demos, and videos. The complete dataset includes 6,300 authors and 4,073 papers. The reference data was problematic, and only 43% of the references had a paper identifier assigned by the ACM DL. While the complete reference text was available, we chose to focus on the visualization, and therefore no further effort was made to parse or otherwise improve the reference data. However, a simple Perl script retrieved the necessary metadata such as paper source, year of publication, title, and authors from the ACM DL. The duplicate author names (e.g., Stuart Card in addition to S. K. Card, etc.) also had to be manually cleaned up.

3.2.2 Topic Clustering

To identify research topics, standard topic clustering technology, internally developed at Microsoft Research, was used. The statistical model underlying the code is called a mixture model [95]. The technology was originally developed for site administrators to help build and maintain category hierarchies. The text-clustering component suggests a set of categories when no explicit structure exists. Titles, references, and keywords in the clustering process were used. A standard list of stop words, months of the year, journal and proceeding titles, and version and page numbers were removed from influencing the cluster results.

Five InfoVis and 22 CHI clusters emerged from using the clustering tool. PaperLens was used in the process of manually naming each cluster by investigating papers and authors in the cluster. For CHI data, some topics were divided into several clusters, which were combined into one cluster. But, individual papers were not moved into other clusters. This resulted in some papers being placed in odd clusters but is typical of any clustering solution. We ended up with 15 CHI clusters summarized in Table 3.1 and sorted by the number of papers in each.

Clusters	Number of Papers
Lab Reports, Applications, Web	618
Multimodal UI	601
CSCW	512
Miscellaneous	407
Usability	334
InfoVis	323
Cognitive Factors in Design	241
Anthropomorphism	209
End User Programming	160
Target Acquisition	140
User Modeling	139
Audio, Tangible UI	130
User Centered Design	119
VR, Input Devices	75
UIMS	65

Table 3.1 15 clusters of CHI papers by topic

3.3 Description of the Interface

The goal of PaperLens is for the novice or expert to gain some insights as to how a field's topics and research activities have changed over time. PaperLens (Figure 3.1) consists of 6 main parts: a) popularity of topic; b) selected authors; c) author list; d) paper list; e) degrees of separation links; and f) year by year top 10 cited papers/authors. This section describes the PaperLens user interface along with the interesting patterns and relationships discovered.

3.3.1 Evolution of Topics

In the popularity of topic view (Figure 3.1a), PaperLens organized papers by their topic and year. The light beige bars represent a group of papers whose height is proportional to the number of papers in the group. This allows an interested user to quickly see trends of the topics over time. As can be seen from Figure 3.1a describing the CHI dataset, the InfoVis category (10th from the top) emerged in the late 1980's and then stayed steady from the early 1990's. The topics of CSCW and Anthropomorphism exhibited steady increases in popularity while the UIMS category almost died out around 1995 (11th through 13th from the top, respectively).

Furthermore, by hovering on an individual topic title, the years when that topic was most popular are signified by a brown border around the relevant column in the popularity of topic view (Figure 3.1a). To help users see when the topic was popular, PaperLens shows the year above the bar. For example, the Cognitive Factors in Design category was the most popular in early years (Figure 3.2a), Lab Reports

etc. in the middle years (Figure 3.2b), and CSCW and Multimodal UI became more popular in recent years (Figure 3.2c, Figure 3.2d).

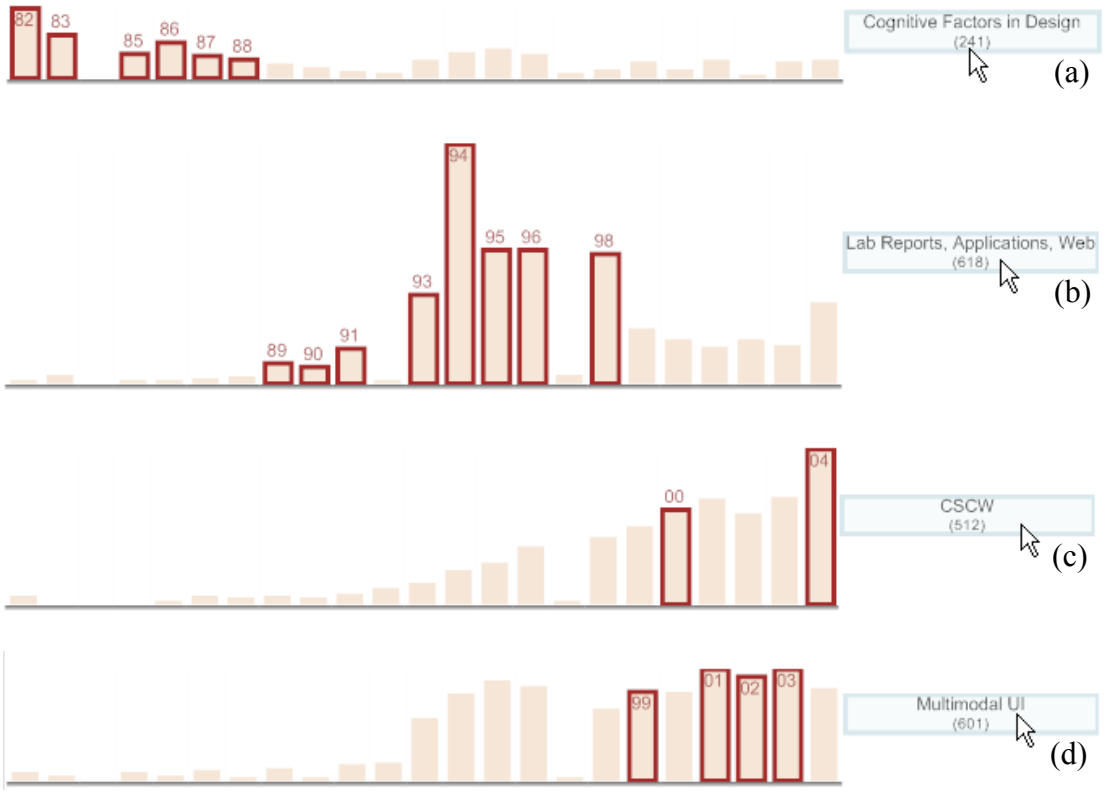


Figure 3.2 Highlighting of the most common topics: (a) Cognitive Factors in Design (b) Lab Reports, Application, Web (c) CSCW (d) Multimodal UI.

3.3.2 Easy Access to Papers/Authors

PaperLens provides a way to search for specific papers – a simple substring match by title. By typing in a keyword, such as "3d," the entire visualization is filtered to only show information related to papers that match the search string in their titles. PaperLens also enables users to get a list of papers by year, by topic, or by authors. By selecting a topic in the display, the list of all papers in that area is shown in the

paper list (Figure 3.1e). The number of citations for each paper in the paper list was also shown to show the influence of the paper.

When the user selects authors from the authors list, they are shown in the selected authors area (Figure 3.1b). The author name search is a bit different than the paper search in that its substring search only works from the initial letter of a first or last name. The current search hit is signified by a pink background and users can iterate through multiple hits by clicking the "Find Next" button.

Once authors are added to the selected authors area, all of the papers by them are shown in the paper list and are highlighted in the popularity of topic view, matched to the author by color coding. The color coding enables users to see which topic area a particular author fits in. For example, Stuart Card has mainly published in the InfoVis area, as seen by his representative red color coding seen within the popularity of topic view (Figure 3.1a). The color black was used when two or more selected authors wrote a paper together. By selecting two authors, it could be immediately seen whether they had ever published together.

For the InfoVis proceedings data, which has only 155 papers, it was possible to devote one rectangle to any individual paper in the popularity of topic view (Figure 3.3). This enabled users to select a paper by single click and was viewed positively by many participants. A fisheye technique was also used to help people reveal the individual paper titles for that year by topic when users clicked on a year. If the small rectangles representing papers without overlap for the CHI proceedings were stacked, however, the height for each paper is less than 1 pixel, which turned out to be very difficult to recognize. The fisheye technique did not work either because the number

of papers for which we could show titles in one column was less than 70, and we needed to be able to show as many as 150.

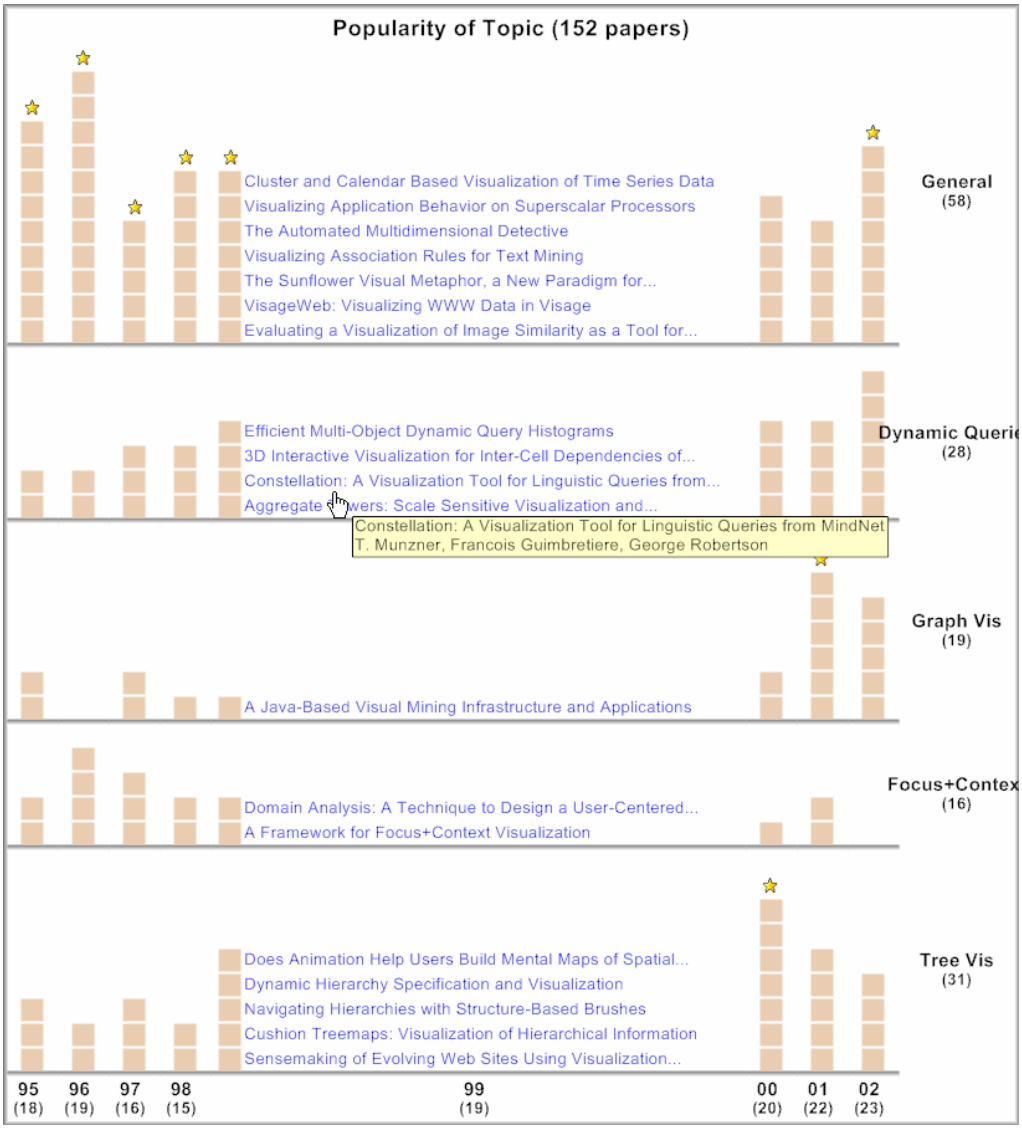


Figure 3.3 For the earlier prototype, a fisheye technique is used to help people reveal the individual paper titles for that year by topic when users clicked on a year.

So, each rectangle was rendered 4 pixels high, and to raise highlighted rectangles were raised to the front. Even so, when several papers are highlighted, the

corresponding rectangles sometimes overlap. So when overlapped, they are shifted one pixel to the right (Figure 3.4). Since overlapping made it difficult to select a paper from the popularity of topic view for the CHI data, a pop-up list menu showing the papers was provided close to the current cursor position. Paper titles are matched to the highlighted paper by color coding. The pop-up menu appears upon mouse-over like a tool tip and is pinned down when users click the popularity of topic view.

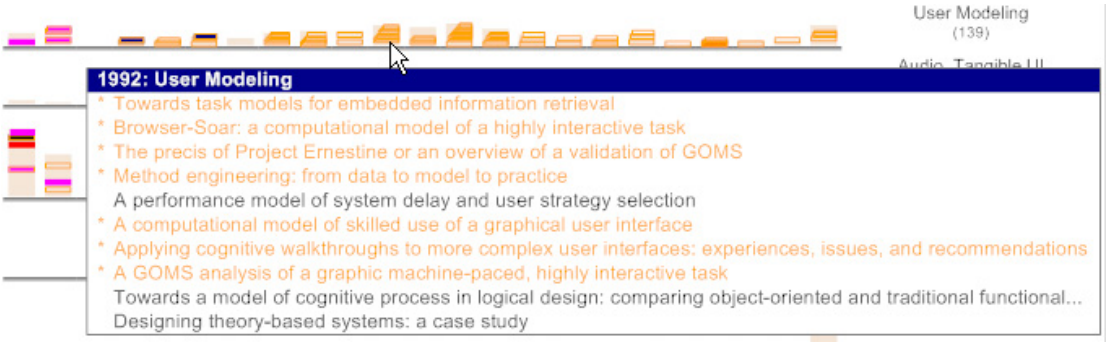


Figure 3.4 A pop-up menu showing the list of papers close to the current cursor position.

Once the menu is pinned down, it works just like a popup menu. Users can select a paper by single click from the list. When a paper is selected, its authors are automatically picked from the author list (Figure 3.1c) and added to the selected authors area. In addition, papers that have referenced the selected paper are highlighted via orange highlighting in the popularity of topic area. A double-click takes users to the ACM Portal with a link to the paper. (For the InfoVis proceedings, accessing a paper was simply a double-click on the paper's rectangle; the title and author name(s) were provided with a tool tip on mouse over as shown in Figure 3.3).

3.3.3 Most Frequently Published Authors

The number of papers published by an author was shown in the author list (Figure 3.1c). Users can sort the author list by the number of papers and immediately see who has published the most. For example, the most prolific author is Brad Myers who has contributed 41 papers to the CHI conference. When users select a topic in the popularity of topic view, a column having the number of papers in that topic is added to the author list. Now users can see who has published the most on each topic. For example, Jonathan Grudin was ranked #1 with 16 papers for the CSCW topic. 16 papers out of 26 were published by Jonathan within the CSCW topic category.

3.3.4 Relationships between Authors

A co-author collaboration graph is often used to find the relationship between individual authors and the center of the community, i.e., the author that has the shortest average path length to all other authors in the graph [29, 101, 125]. The graph among InfoVis authors, however, is too fragmented to give any useful insights. For the InfoVis data, S. F. Roth, the center of the graph, has published 5 papers with 13 co-authors and has only 19 related colleagues among 315 possible individuals. Even though the largest component of the collaboration graph for CHI authors is bigger than that for InfoVis authors, it is still fragmented with many small components. We therefore decided to display all of the related colleagues for an author when he or she is selected by users. The shortest path length between two

authors was computed on demand and called degrees of separation links (Figure 3.1d and Figure 3.5).

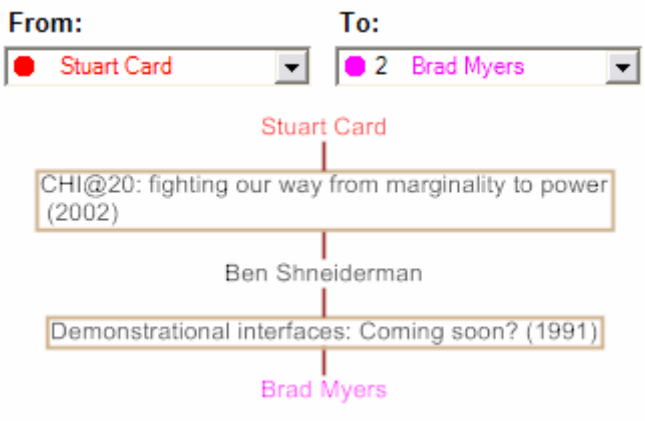


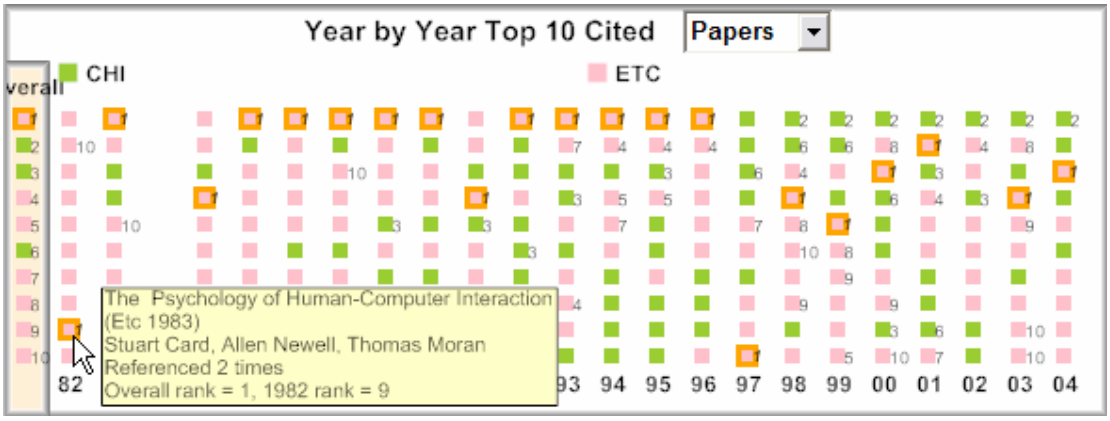
Figure 3.5 Degrees of Separation Links shows how Stuart Card and Brad Myers are connected by a co-author relationship.

The degrees of separation links view plays an important role in showing the relationships between authors in the CHI community. The From combo-box contains the same list of authors as the selected authors area. Once an author is selected from the From combo-box, all the selected author's related colleagues are displayed in the To combo-box with the corresponding degrees of separation. When an author is selected from the To combo-box, the shortest path between two authors in our dataset through their degrees of separation links is displayed. For example, Stuart Card and Brad Myers are connected indirectly to each other because they have each co-authored a paper with Ben Shneiderman.

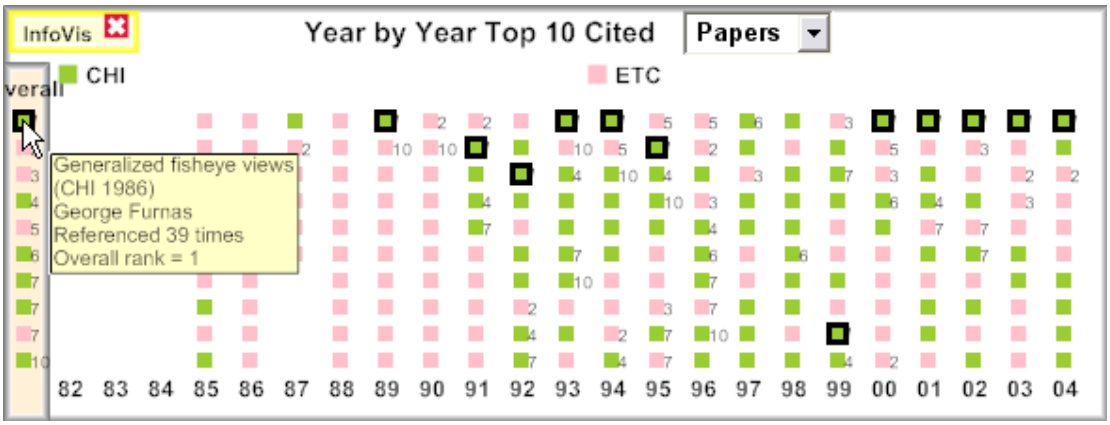
3.3.5 Most Frequently Referenced Papers

One of the interesting questions was "Which papers/authors are most often referenced?" because this is one important metric indicating influential papers/authors. The number of references was counted overall. In addition, it was computed by year and by topic to show trends.

The top 10 most frequently cited papers are viewable in the overall list within the year by year top 10 cited papers view. The papers included here are all the papers available to us in this study, including all the CHI papers and all the papers that the CHI papers reference. One can also see the detailed information for each of the top 10 papers and how it was referenced over time by hovering on an individual paper. It is easy to see that the most frequently cited publication by CHI authors to date is "The Psychology of Human Computer Interaction" by Card *et al*. (Figure 3.6a). Selecting a paper from the year by year top 10 cited papers view (Figure 3.1f) has same effect as selecting a paper for the popularity of topic view.



(a)



(b)

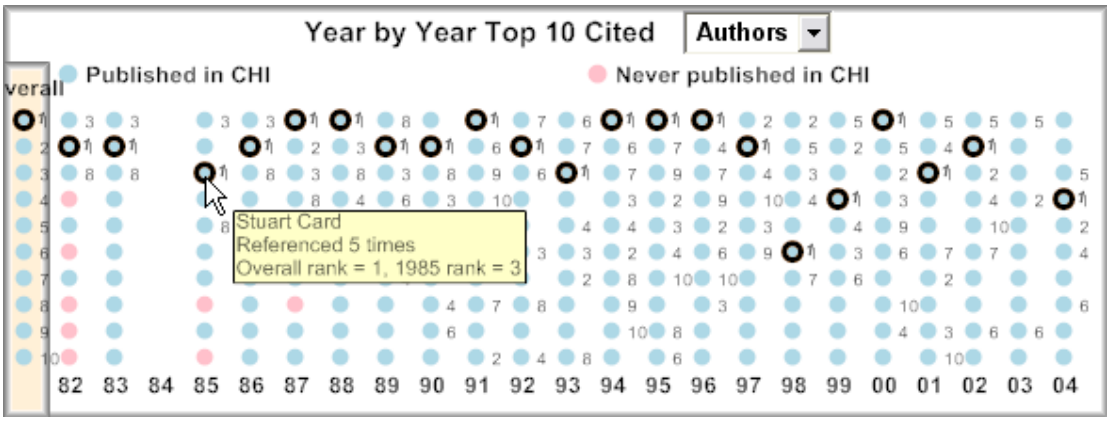
Figure 3.6 (a) The Psychology of HCI paper is the most referenced paper by all CHI papers. (b) Generalized fisheye views paper is the most referenced by InfoVis CHI papers.

The important CHI papers were distinguished with the color green. It can immediately be seen that CHI publications themselves have been a significant source of inspiration for the CHI community to date and that 3 of the top 10 papers most frequently cited are CHI publications. Most years do have at least one CHI paper in the top 10, with the notable exception of 1982, which was CHI's first year.

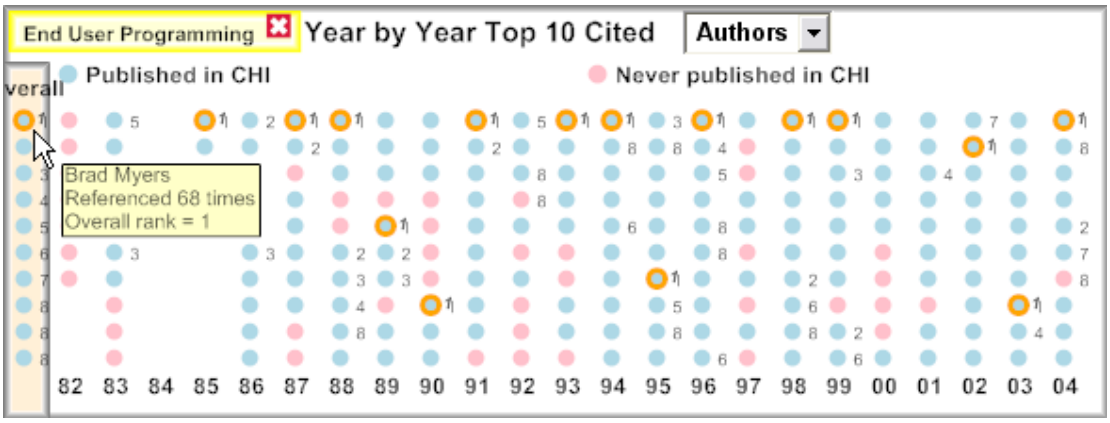
When users select a topic from the popularity of topic view, the year by year top 10 citations area is filtered to only show the frequent citations for that topic area. In this way, PaperLens allows users to quickly discover the most influential papers in a particular topic area. For instance, for the InfoVis topic, the Generalized Fisheye Views paper written by George Furnas was the #1 most frequently cited paper (Figure 3.6b).

3.3.6 Most Frequently Referenced Authors and Their Papers

Ranking the frequent citations by author shows frequently cited authors that either have or have not published in CHI. Since the authors were colored pink if they have not ever published in CHI, it can be seen that all overall top 10 cited authors have published in CHI, even for the End User Programming topic (Figure 3.7). However, for several years, many top 10 frequently cited authors for the End User Programming have not published in CHI (Figure 3.7b). Selecting an author from the year by year top 10 cited authors view (Figure 3.1f) shows not only any papers selected authors have published in CHI, but also those papers that have referenced them via orange highlighting in the popularity of topic area (Figure 3.1). Users can immediately discover which areas were most influenced by the selected author. For instance, for End User Programming, Brad Myers was the most frequently cited author (Figure 3.7b).



(a)



(b)

Figure 3.7 (a) Stuart Card is the most frequently cited author by all CHI papers and (b) Brad Myers is the most frequently cited author for End User Programming

Once we know who the most referenced authors are, it would be interesting to see how many times each of their papers is referenced. We decided to add a view similar to that available in CiteSeer. A double-click on an author in the year by year top 10 cited authors view opens a number of citations view (Figure 3.8), which shows the number of citations of the papers written by the selected author. Furthermore, when a topic is selected by users, the number of citations is counted only by the papers in that topic area. For example, Stuart Card was the most frequently cited author by all CHI authors and "The Psychology of Human Computer Interaction" was

the seminal contribution (Figure 3.8a). While he was also the most frequently cited author for the InfoVis topic, that paper was referenced only 8 times and other papers such as the Perspective Wall and Cone Trees papers were referenced more often (Figure 3.8b). It also shows the distribution of references by year at the bottom half of the window. This view has advantages over CiteSeer in that it is organized by author instead of by an author's individual paper.

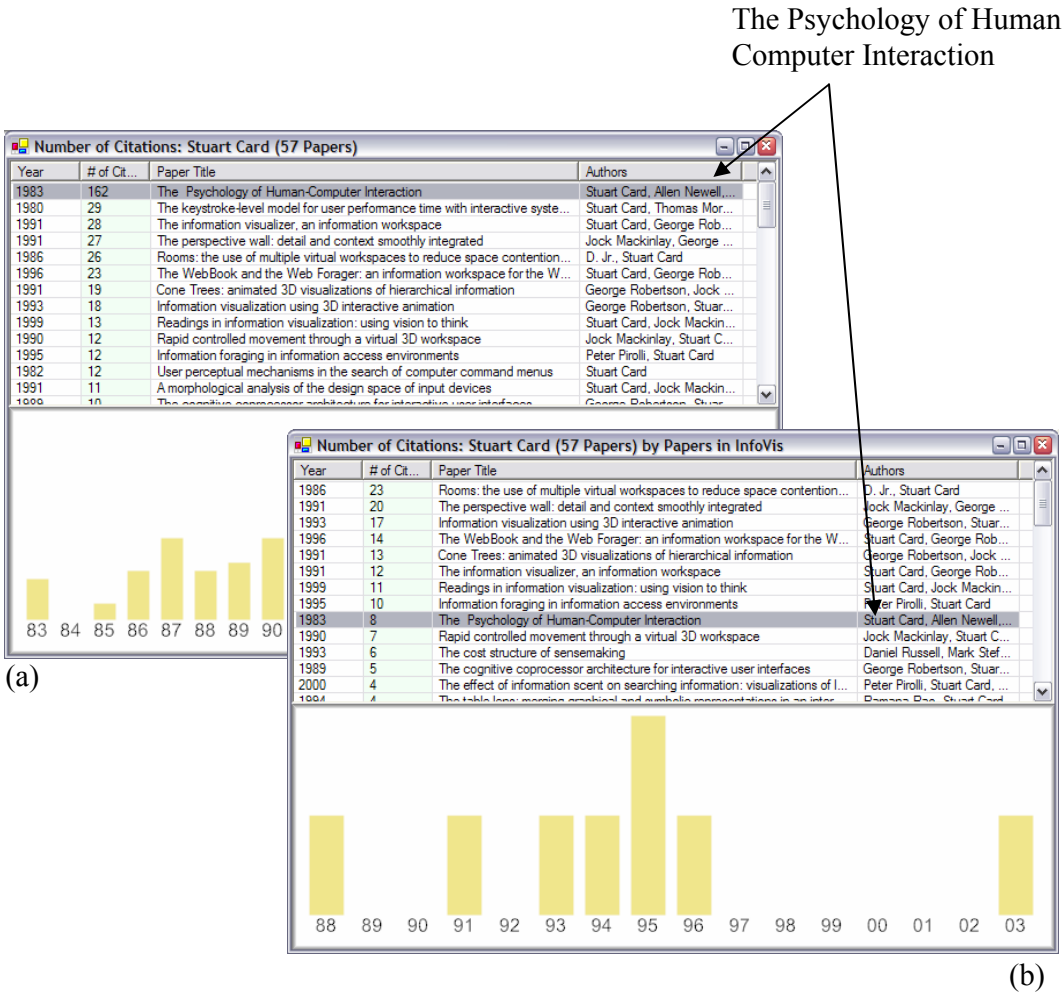


Figure 3.8 Number of citations of the papers written by Stuart Card (a) by all CHI publications and (b) by the papers in the InfoVis topic. "The Psychology of Human Computer Interaction" was referenced 162 times by all CHI publications but only 8 times by the papers in the InfoVis topic.

3.3.7 Weaknesses

As is often the case with powerful and new visualization tools, PaperLens requires some learning time. New users need to learn how to decode the various color

mappings and highlighting. They are also required to understand how views are coupled because all views and interactions are symmetrical and can be used for both input and output.

3.4 User Study

In order to understand how useful our original user interface design ideas were, and to help us iterate to a more usable design, a user study was carried out using the InfoVis dataset and the first iteration prototype.

3.4.1 Participants

Eight researchers (including 1 pilot subject) who were on the "fringe" of the information visualization community but had never published at the InfoVis conference itself were recruited internally. Four of the researchers were computer science graduate student interns, and four were full time researchers, and all were interested and actively working in the area of HCI. Ages of the participants ranged from 24 to 42. The pilot data is included in the discussion of the usability issues observed, but not in the reporting of the experimental task data, as the task set was altered slightly after the pilot.

3.4.2 Procedure

Participants were given a brief tutorial of the system, spending no longer than 20 minutes reading about and interacting with features of the system. This segment of the study was considered "think aloud," and any usability issues they experienced during this walk through of the system were noted by the experimenter.

Next, the participants were asked to carry out several experimental tasks with the system, which were timed and scored for correctness. The tasks were meant to not only evaluate the usefulness and usability of many features of the initial prototype, but also to determine areas that might need to be redesigned or even eliminated to scale up to the much larger CHI proceedings dataset. All users carried the tasks out sequentially, as quickly as they were able. Once a task was over, participants were allowed to discuss what did or did not work well with the system in terms of efficiently completing the task, and the experimenter recorded these comments. Once all of the tasks were completed, users were asked to fill out a satisfaction questionnaire about PaperLens. All sessions were completed with participants run individually and lasted no more than one hour. Participants were provided with a coupon for a free lunch or dinner from the cafeteria for their participation.

3.4.3 Tasks

The list of the tasks follows.

1) Who published the only paper on Graph Visualization in 1998?
2) How many papers did S. K. Card publish at InfoVis over the 8 years in our database?
3) Who were George Robertson's coauthors on his only paper in the database?
4) How many degrees of separation exist between S. F. Roth and S. G. Eick?
5) Which topic area has enjoyed gradual growth over the last 8 years?
6) Which topic area has all but died out in terms of papers published on that topic over the last 8 years?

7) Which topic area has had many more papers published on that topic during the last 2 years in our database?
8) Which authors are in the top 10 most frequently cited list but have not published at InfoVis?
9) How many papers of the top 10 most frequently cited papers are from InfoVis?
10) How many papers in the top 10 most frequently cited list are from CHI?
11) Which topic area references the most frequently cited paper most often?
12) Go to the most frequently cited InfoVis paper and read it's abstract.
13) In the Dynamic Queries topic area, which author is the most frequently cited?
14) What was the last year that S. K. Card published in this database?
15) Who was the most frequently cited author in 2001?
16) How many papers did J. Mackinlay and S. K. Card publish together at InfoVis over the 8 years in our database?

3.4.4 Results

3.4.4.1 Task Times and Errors

Overall, participants were able to correctly answer the tasks used in the study 97% of the time. There were only 5 incorrect answers provided out of a possible 112 questions across participants. Three participants each gave one wrong answer and one participant incorrectly answered 2 questions. Each incorrect answer was from a different question for each participant. Incorrect answer times were not included in the task time analysis.

Overall, average task times were fast, with only the last task taking much longer than the others (1 minute and 5 seconds, on average). This task required users to figure out how many papers J. Mackinlay and S. K. Card published together at InfoVis, which required users to remember that black color coding was used to signify multiple co-authors on a paper. Most other tasks were performed in less than 20 seconds, as can been seen from Figure 3.9. Usability issues leading to longer task times will be discussed next.

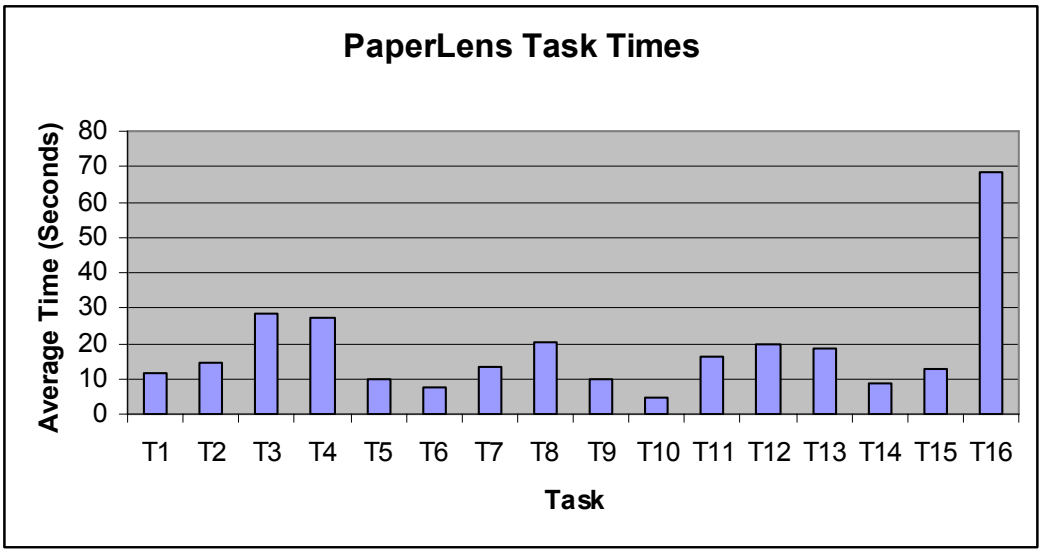


Figure 3.9 Average task times using PaperLens with the InfoVis proceedings.

3.4.4.2 Usability Issues

Several usability issues were observed that needed to be addressed through design iteration. These issues were prioritized based on how many of the participants encountered them and the severity of the issue in terms of being able to easily recover from it, or based on how long the issue delayed finding an answer to a task. The highest priority issues centered on the way searching for authors was implemented:

in this prototype it was initially a string-based search that did not allow users to search for first or last names separately, and found substring matches anywhere in the name (not just at the beginning of names). This issue was addressed by providing columns for searching on the first and last name, in addition to fixing the way our substring matches worked (from the beginning of names).

There were several high priority issues observed where our system did not behave "symmetrically." In other words, if you could launch a paper from one list view, you should be able to open it from any list view, etc. All of these symmetry issues have been addressed in the redesigned system.

Finally, users gave us feedback concerning the degrees of separation list and links views (Figure 3.10) — while some users liked the links view, others thought that it was more "recreational." There were also some issues with the default way the degrees of separation were being displayed in the links view (e.g., picking the longest link chain by default) that had to be addressed. To help alleviate usability issues in this area, in addition to freeing up screen real estate, the list and links views were combined into one view (now called degrees of separation links in the current system) and allowed users to pick the degrees of separation between any selected author and related people.

Other design changes were made to fix less severe usability problems. While there were some ways to clear selections for each view, they were not consistent. Furthermore, there was no way to clear all selections. Many participants also wanted to toggle the selections for the topic label, year, paper, and author. These issues were addressed by adding a "Home" button and making all selections "toggle-able." One

more issue about search was the desire to cycle through the multiple matches. Iterating through search hits using the "Enter" key turned out not to be intuitive. Instead "Find Next" button was added and "Enter" was used for selection.

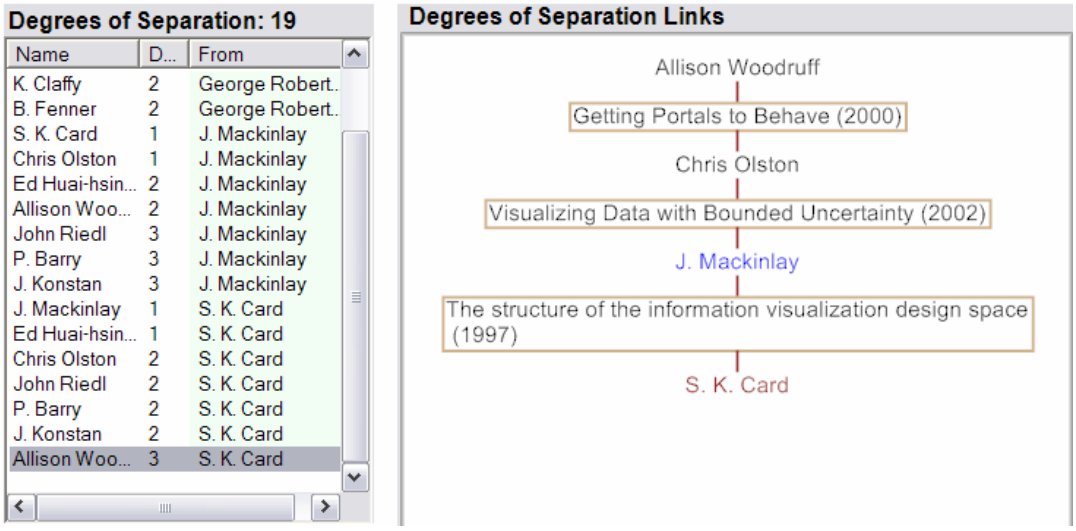


Figure 3.10 Degrees of Separation List and Links views from the earlier prototype.

Some users were concerned that there were so many different colors in the original user interface. 8 different colors were used to represent authors for the InfoVis data because 8 is the maximum number of authors of one paper. For the CHI data, the maximum number of authors is 15. We suspected that it would not be useful to have 15 different colors to distinguish authors from each other. We decided to use a single color if the number of selected authors is larger than or equal to 5.

3.4.4.3 User Satisfaction Ratings

The satisfaction questionnaire that users filled out at the end of the study session was analyzed. All ratings were on a 1-7 Likert scale, with 1=Disagree and 7=Agree. The average ratings are shown in Table 3.2 below. Overall, the system was rated fairly

highly for a first time iteration of user testing. However, there was clear user frustration around the ease of learning the system, its look and organization, the degrees of separation area, and error recovery. In other words, the usability issues that were identified were fairly well corroborated in the satisfaction ratings. It is hoped that the redesign changes will go a long way to alleviate these satisfaction problems.

Overall, I am satisfied with this system.	5.3
It was simple to use this system.	4.3
I was able to efficiently complete the tasks and scenarios using this system.	6
I felt comfortable using this system.	5.3
It was easy to learn to use this system.	4.4
The "look" of this system was pleasant.	4.9
I liked using this system.	5.9
The organization of information in this system was clear.	3.9
Whenever I made a mistake using this system, I could recover quickly and easily.	4.6
It was easy to discover trends of the topics using the "Popularity of Topic" view.	6.3
It was easy to discover relationships between authors using the "Degrees of Separation Links" view.	4.1
It was easy to discover the most referenced papers/authors using the "Year by Year Top 10 Cited Papers/Authors" view.	6.9
Overall, highlighting on the screen was helpful.	6.1
Overall, the use of color was appropriate.	5.3

Table 3.2 Average Likert scale ratings for PaperLens, using the scale of 1=Disagree, 7=Agree.

3.5 Implementation Details

PaperLens was implemented in C# and runs on any standard Windows PC. PaperLens consists of 32 classes and about 8,000 lines of code. In addition to the main data file in an XML format, an application-specific version of the datasets in several simple tab-separated text files was used. For the fast access, the entire dataset was loaded into memory.

All the graphical views are implemented with Piccolo.NET, a shared source toolkit that supports scalable structured 2D graphics [14, 106]. PaperLensControl (for the popularity of topic view), AuthorGroupControl (for the selected authors view), DOSLControl (for the degrees of separation links view), and RankControl (for the year by year top 10 cited papers/authors view) are all inherited from PScrollableControl. For the number of citations window (shown in Figure 3.8), CitationsControl is also inherited from PScrollableControl.

PaperLensControl contains three views, paperView, yearTitleView, and topicTitleView, which are an instance of PNode. The paperView is a grid, where a cell C(i, j) represents the i-th topic and the j-th year. The yearTitleView and topicTitleView show titles for columns (year) and rows (topic) respectively.

3.6 Discussion

PaperLens is a visualization tool that allows users to see trends and topics in a field, in addition to influential papers and authors. Two design iterations were described; the first designed for a smaller set of conference papers (InfoVis) and the second

designed for a larger set of papers (CHI). The design iteration was driven both by user studies and by dealing with issues of scale.

A rule of thumb for PaperLens was that the visualizations should be designed to best support the tasks. Traditional node-link graph visualizations do no provide good support for cases where users want to see evolution of topics and read the labels of directly connected nodes. The key elements of the PaperLens design that make it work well are:

- An abstract overview
- Multiple small and simple components to best show the different aspects of the data
- Relationships shown through interactivity and tightly-coupled components
- All visual elements are laid out along axes with well defined metrics

PaperLens is novel in that it visualizes a graph without drawing any nodes or links except for the degrees of separation links view. To facilitate tasks that require showing a distribution of items and direct connections, especially when reading of labels and attributes is important, PaperLens uses multiple coordinated views, tightly coupled to reveal the data in its complexity. Using PaperLens, many interesting patterns and relationships were discovered, which could not have been revealed using existing tools. For example, CHI was the most influential source of references used by InfoVis authors to date. George Robertson and George Furnas, both influential in the InfoVis proceedings, have only published once or never, respectively, at InfoVis. In the CHI proceedings, many similar trends and patterns were discovered across the field of HCI, as discussed above.

PaperLens' power comes from tightly-coupled views across papers, authors, and references. The ability to query by time and topic has enabled novel views that we found very beneficial in our explorations of this domain. PaperLens' design, which shows relationships within a complex network through interactive highlighting, is a novel alternative to a more common network visualization with a node-link diagram.

3.6.1 Limitations

As a short term intern project and a proof of concept, PaperLens is highly customized to the InfoVis and CHI datasets. While PaperLens can visualize other conference datasets prepared in the same format as the CHI dataset, the current implementation is not applicable to general graph datasets and has some issues with scaling.

In addition to the difficulties (described in the section 3.2) for preparing data, another problem is to assign a topic category to each paper. Not only is it difficult to get a reasonable number of meaningful clusters, but it is also hard to properly name each cluster. Another issue with scalability is that PaperLens loads the entire dataset into memory. Even though enough memory could be available, it might take too much time to read all the data files at start up. Lastly, the screen size is limited and currently PaperLens does not provide any aggregation mechanism. For the 1280x1024 resolution, PaperLens may not be able to visualize conferences older than 30 years or having more than 20 topics. Since there might be hundreds of topics for much larger datasets such as the ACM digital library, it may be necessary to provide ways to aggregate them and enable users to browse the topic hierarchy.

3.6.2 Co-authorship Visualization

While the degrees of separation links view show a relationship between two authors, there is no efficient way to show relationships among more than two authors. And PaperLens does not support browsing very well. Since node-link graph visualizations can help with those tasks, a new interactive graph visualization component called TreePlus has been developed based on a tree-style layout. Furthermore, it was integrated with the next generation of PaperLens. This will be explained in detail in Chapter 5.

3.6.3 Next Generation of PaperLens

After I came back from my internship, I developed a visualization tool called EcoLens (Figure 3.11), by applying the PaperLens concept to food web data. This work was a part of our biodiversity project supported by the National Science Foundation. EcoLens allows biologists to browse through a collection of food webs, find webs of interest, and then visualize an individual food web using TreePlus. It is essentially a front end to relational data tables that offers coupled interaction, searching, and simple bar chart visualization to replace complex queries. Food webs and EcoLens will be explained in detail in Section 7.2.

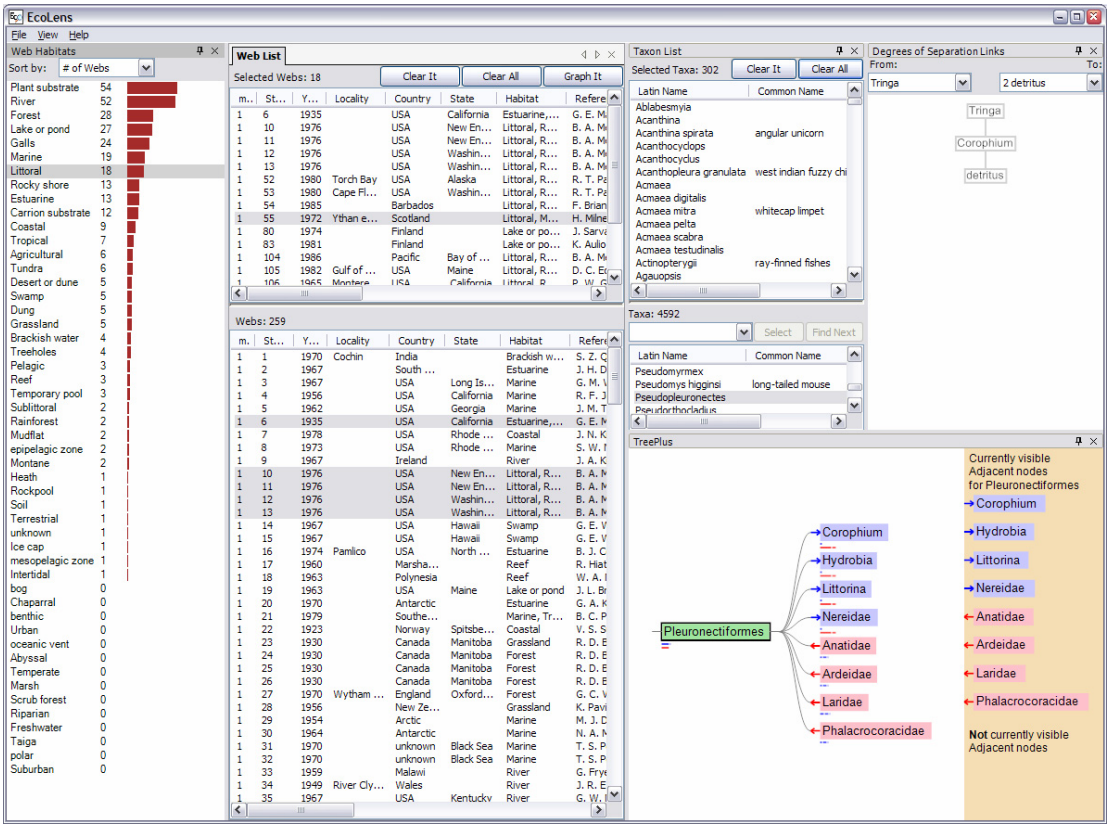


Figure 3.11 EcoLens provides easy exploration of a collection of food webs by sorting and selecting in tabular form, coupled with graphical representations in bar charts (left) or network visualizations (lower right).

Another visualization called NetLens [81] (Figure 3.12) was implemented by Hyunmo Kang at the HCIL by generalizing PaperLens. Along with Catherine Plaisant and Benjamin B. Bederson, I was a member of the project team and participated in designing NetLens. The design was driven by dealing with the limitations of PaperLens described above. It allows users to display any graphs that can be represented by two main entity types. For example, the CHI data contains papers and authors and the email data have emails and people.

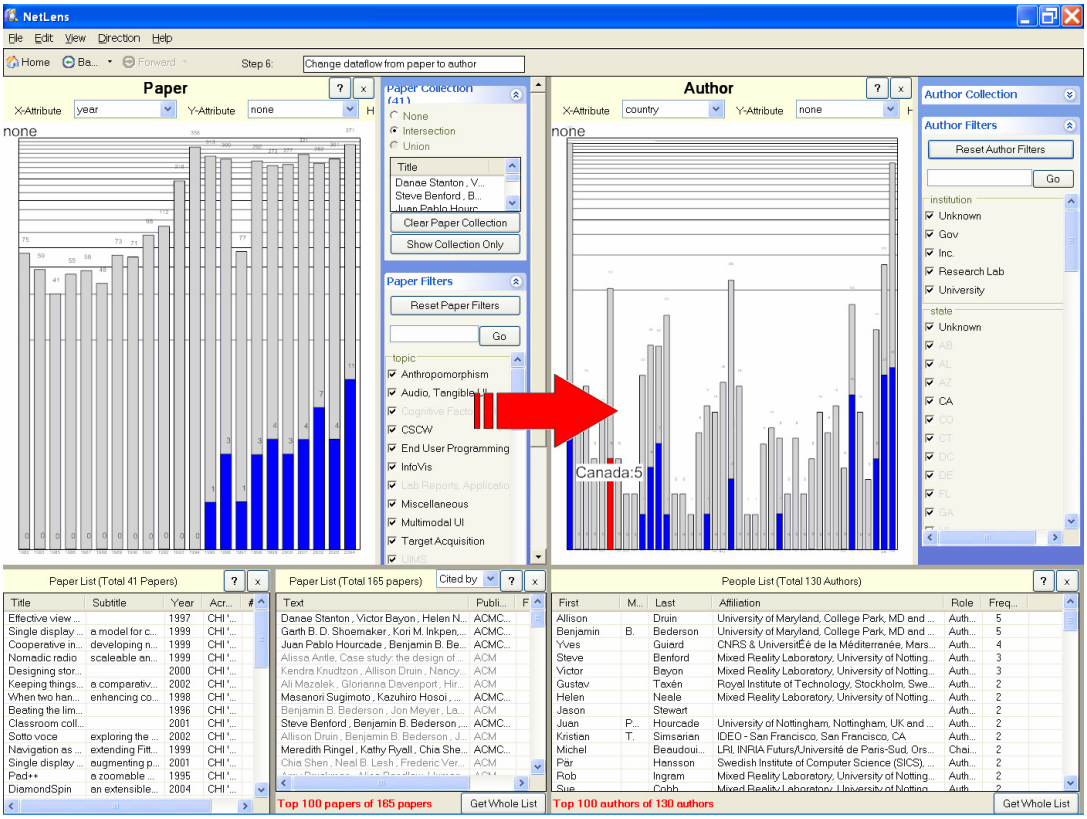


Figure 3.12 NetLens

Chapter 4

TaxonTree: Visualizing Biodiversity Information

Biodiversity databases have recently become widely available to the public and to other researchers. To retrieve information from these resources, users must understand the underlying data schemas even though they often are not content experts. Furthermore, names of organisms are essential to the biological databases, including genomic databases, so a tool that allows effective searching and browsing of these names has wide application.

I developed an interactive tree browser called TaxonTree (Figure 4.1) to visualize the Linnaean classification for taxonomic names in the Kingdom Animalia. I worked with Benjamin B. Bederson and two biologists, Cynthia Sims Parr and Dana Campbell as part of an NSF-funded project on biodiversity informatics. TaxonTree allows users to browse and search a tree of about 200,000 animal names with associated attributes constructed by integrating data from a number of public and private sources [76, 93, 130, 131]. TaxonTree uses animation, zooming and panning, and integrated searching and browsing to help users both find what they want and understand the biological context of what they have found.

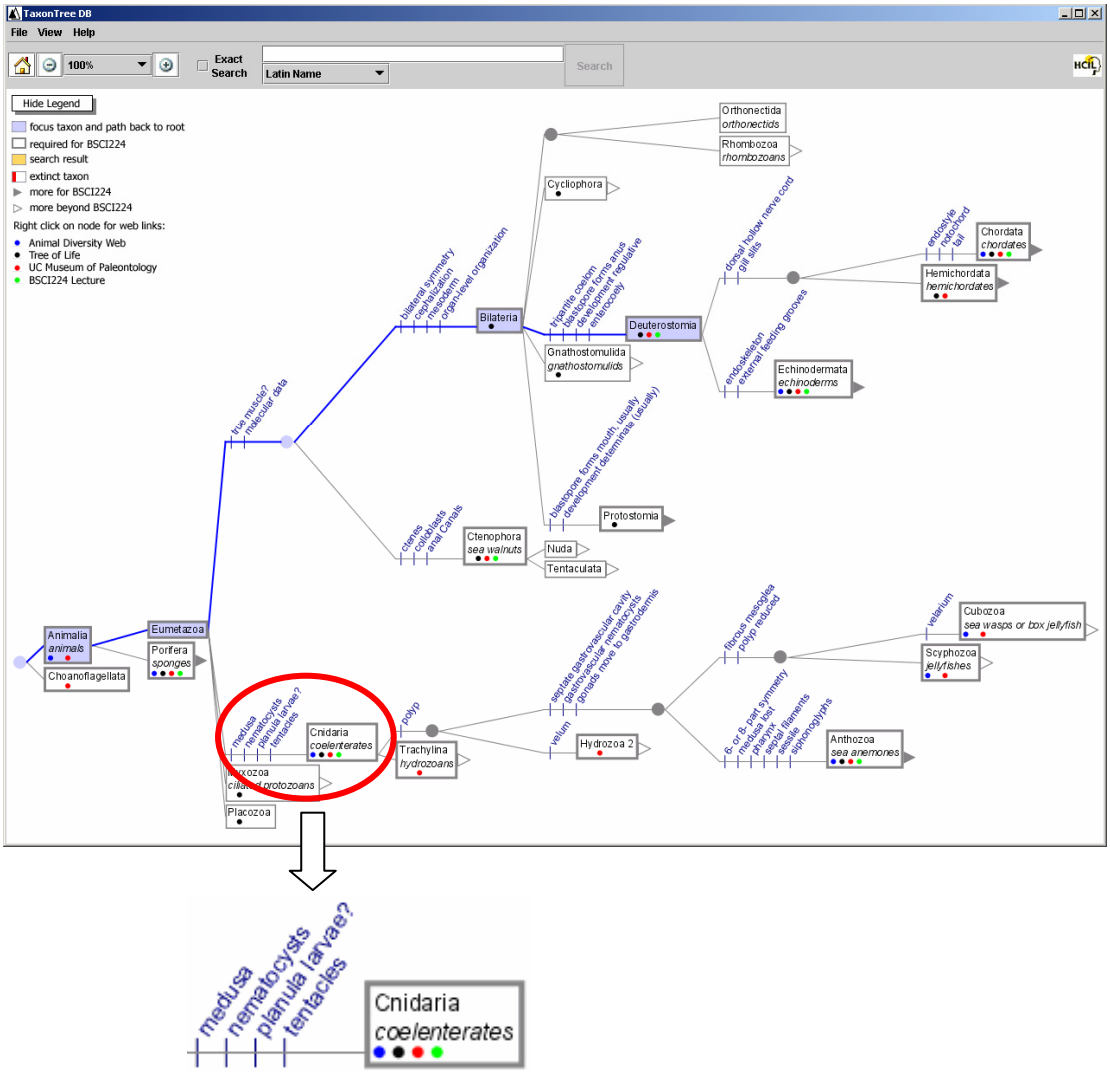


Figure 4.1. TaxonTree visualize the Linnaean classification for animal names. Magnified nodes show synapomorphies (evolutionarily significant, diagnostic characteristics) and color-coded dots to represent available external websites.

4.1 SpaceTree

TaxonTree is an extension of SpaceTree [109] (Figure 4.2), a tree browser previously developed by Jesse Grosjean, Catherine Plaisant, and Benjamin B. Bederson at the

HCIL. 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.

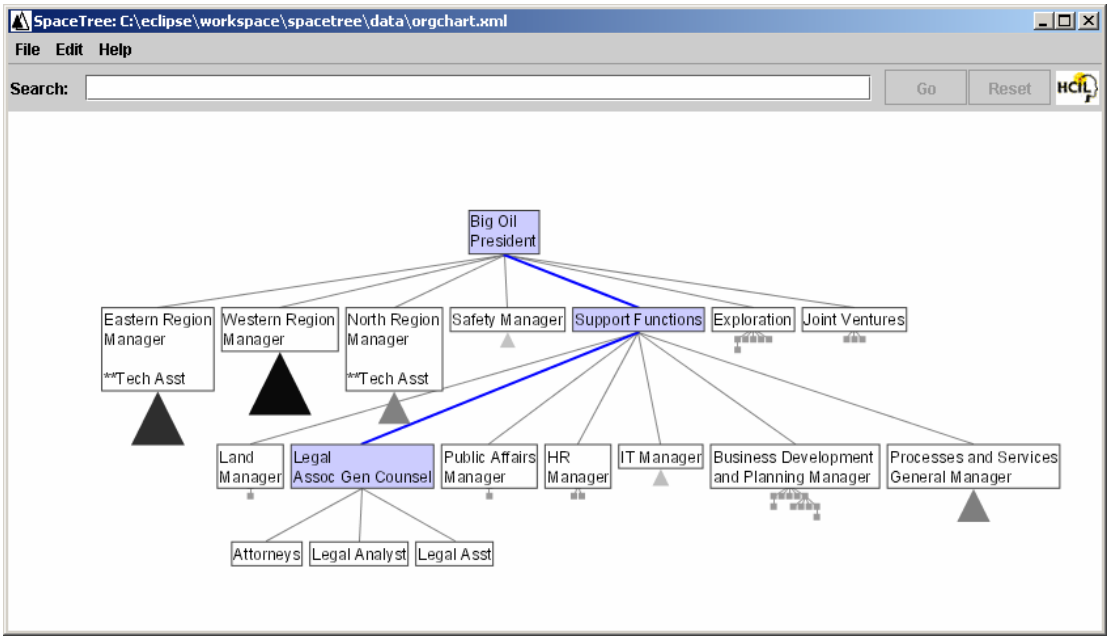


Figure 4.2 SpaceTree lays out trees to best fit the available screen space.

Users can navigate the tree by clicking on nodes or by using the arrow keys. SpaceTree maximizes the number of levels that are opened based on feedback from users that they didn't want to open the tree one level at a time. When the focus is changed, the tree is animated to its new layout, which makes full use of screen space, in three main steps: 1) trims the tree of the branches that would overlap the new branch to be opened; 2) moves the tree so that the new tree layout will center on the window, 3) expands the branch out of the new focus point. While animating, SpaceTree retains landmarks to help users maintain their orientation. It uses the current focus and the path up to the root as landmarks and highlights the ancestor path

of the current focus. SpaceTree provides icons to preview the topology of branches that cannot be fully opened because of lack of space.

SpaceTree also supports filters and searches. As users type a string, SpaceTree highlights the relevant nodes within the tree. Users can see a filtered view of the tree, displaying only the paths to the matching nodes.

4.2 Applying Tree Visualization to the Biodiversity Domain

SpaceTree was developed and tested for general-purpose applications. It was iteratively designed with feedback from users who had a particular need for hierarchy browsing at the time of the project. While the authors showed that SpaceTree perform relatively well for both navigation and topology tasks for general tree visualization through a controlled study, it had not been applied to the biodiversity domain yet.

Furthermore, despite a long history of general research in tree visualization [66] these approaches have rarely been applied to or evaluated in the biodiversity domain. To explore the biodiversity domain and develop an interface for it in concert with its users, methodologies adapted from collaborative design [39] was used.

The target audience for TaxonTree was students taking a second-year college course (BSCI 224) at the University of Maryland entitled Introduction to Animal Diversity. As biology majors, they are becoming familiar with the biological domain space but cannot be considered experts. To incorporate the extra detail that the course covered about known evolutionary relationships, we constructed a specialized tree of animal names that further resolved the Linnaean classification at higher levels

[68, 93]. A team of five "design partners" who volunteered from the Animal Diversity course assisted us.

4.2.1 Domain-Specific Visualization

TaxonTree was designed to support biodiversity data. Biologists give organisms scientific names, usually Latin or Latin-like, that must follow certain rules to be considered official by the scientific community. Common names, on the other hand, are informal ways of referring to organisms. While they are not standardized (they differ according to language and dialect of the laypeople using them) common names can be very useful for non-experts. As you can see from nodes in Figure 4.3, common names were displayed with an italic font style so that users could easily distinguish them from scientific names. We also provided links from nodes directly to external web pages on four different, publicly available websites (Animal Diversity Web, Tree of Life, University of California Museum of Paleontology, BSCI224 Lecture Note Page). In the current version, these links are advertised by color-coded dots, and are available by a right click from the node. Thus, unlike other node-link visualization programs, our users can browse or search the structure of the taxonomic data, and then immediately obtain further information on a taxon.

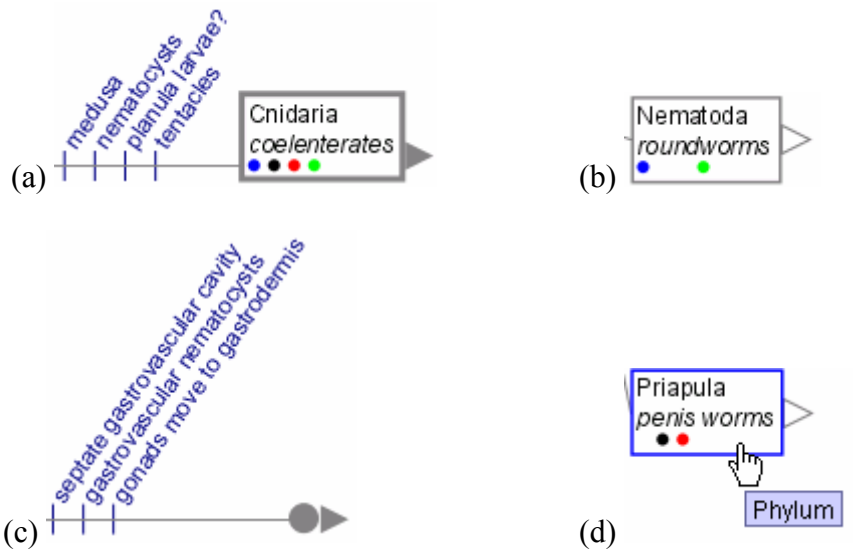


Figure 4.3 (a) Node for required course material with synapomorphies with color-coded dots. (b) Node for non-required material. (c) Unnamed node with synapomorphies. (d) Rank is shown as a tool tip.

Some features were designed explicitly for the University of Maryland Animal Diversity course, such as visual distinction of required course material from non-required material (Figure 4.3a vs. Figure 4.3b), bookmarks of names for future reference, and display of biological ranks for each node as a tooltip (Figure 4.3d). Synapomorphies, attributes of nodes that show how that node is distinguished from its siblings, are presented as text attached to slashes on the branches preceding the involved node (Figure 4.3a, Figure 4.3c). In other words, in TaxonTree, not only links but also nodes can have attributes. To accommodate the needs of our target audience, who were required to know evolutionary relationships, unnamed nodes (Figure 4.3c) were added and displayed as circles to reflect binary branching.

Only 182 nodes are required for the BSCI 224 course. The browsing can be limited only to the nodes necessary for the class, but the default was to browse all of the nodes. It is also possible to display a complete overview of all 182 nodes required for the BSCI 224 course (Figure 4.4) at once.

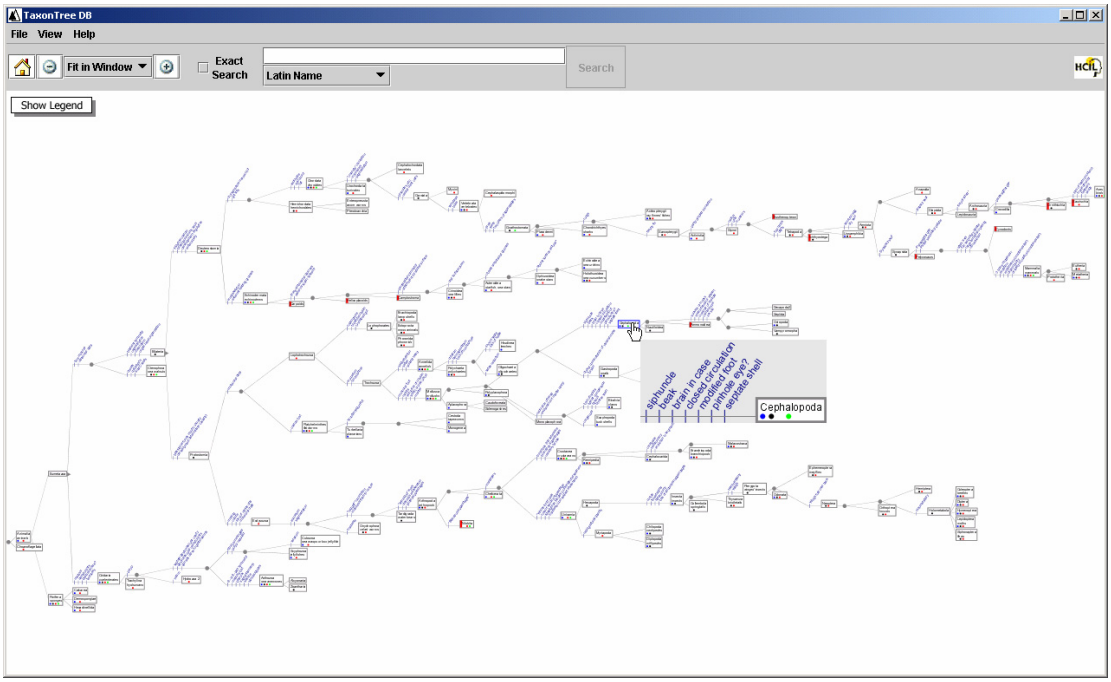


Figure 4.4 TaxonTree enables users to display an overview of all 182 nodes required for the course. Magnification of a node is shown as a tooltip.

4.2.2 Search

TaxonTree provides several ways to search. Users can search with Latin name or common name or synapomorphies. In order to serve a broad audience, often lacking content expertise, TaxonTree also enables users to search on the full text of the Animal Diversity Web. Search results are highlighted within the biological context of their classification tree (Figure 4.5). This helps users understand patterns in the

results, for example all squids are mollusks but there are several subgroups of squid. Furthermore, users can carry out additional browsing, giving them a better sense of the search results. TaxonTree also automatically zooms out so that the search result tree always fits on the screen.

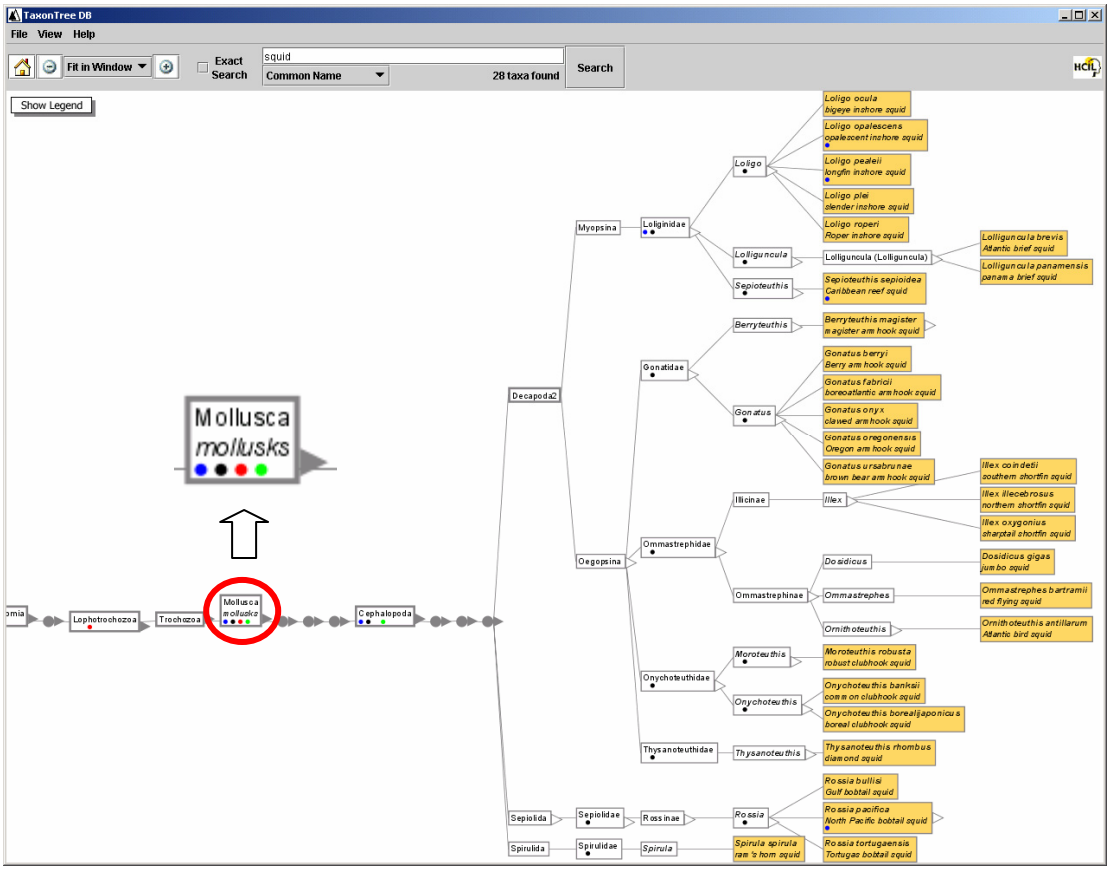


Figure 4.5 TaxonTree highlights search results within the biological context of their classification tree. All squids are mollusks but there are several subgroups of squid. Magnified node shows a small "more" triangle on the right side indicating that there are more nodes to be found by clicking on this node.

4.2.3 Modified Interaction

By modifying SpaceTree's interaction styles, TaxonTree better accommodates the target users' need. SpaceTree automatically closes the branches that would overlap a newly opened branch when users change the focus, which has two major advantages: 1) the screen is less cluttered; 2) the siblings of the focus node are always adjacent. However, it was found that some of the more sophisticated "auto-layout" features of SpaceTree were confusing to our design partners and project members, especially those with content experience and interest. The biggest extension in TaxonTree provides a way to close children nodes manually to give users greater control, thereby facilitating comprehension. Automatic subtree closing was made optional; the default interaction is that nodes toggle open and closed.

While SpaceTree provides an option to choose the tree orientation, TaxonTree fixes the tree root on the left. This allows TaxonTree to support deep trees while providing readable labels at all nodes with typical screen aspect ratios. Finally, instead of using the sophisticated preview icons, only a "more" triangle (shown in Figure 4.5) was used to indicate further nodes.

4.3 User Study

In May 2003 a qualitative study was conducted with three main goals. First goal was to characterize how users of this domain think about biodiversity information in general. Are they more likely to look for information using scientific or common names? What taxonomic rank (species or higher) are they more likely to target? What kind of information are they most interested in? Second was to investigate the

usability and interaction preferences with this particular software. Are users comfortable with integrated searching and browsing, and with animation and zooming? Third was to examine how this kind of information retrieval interface can assist information understanding in this domain. Do students use the tree visualization to successfully complete tasks that require interpretation and understanding of the underlying data structure?

A qualitative methodology was chosen because user behavior in this domain has never been studied. Also, the aim of TaxonTree is to foster content understanding so standard metrics of efficiency are unlikely to be appropriate. Insights gained from this study should guide both design and quantitative assessment of future tools.

4.3.1 Participants

18 undergraduate volunteers (8 male: 10 female, 18 to 20 years old) were recruited from the Animal Diversity course at University of Maryland. None of them were part of the above described design partner team. Each participant was given ten dollars for his/her participation. We tested five pairs of users and eight single users for a total of 13 sessions, or user "teams." The study occurred at the end of the semester so participants were largely familiar with the biological content but could not be considered experts. The software had been demonstrated in lecture and distributed to all students on CD-ROM for personal use two weeks prior to the study. Users reported they had used the program for an average time of half an hour, and eight out of 18 had not used it at all.

4.3.2 Procedure

Each session lasted 30 to 45 minutes. Each user filled out a survey to determine their computer usage background and amount of time previously spent with TaxonTree. They were seated in front of a 2GHz Windows XP laptop with an ordinary mouse, a 1280x1024 pixel display and 512MB RAM, placed on a standard office desk. The computer screen was videotaped throughout the testing to capture both the actions and verbal comments of the users.

After briefly demonstrating TaxonTree features, user teams were asked to perform nine information retrieval tasks, described below. At the end of the tasks, open-ended questions were asked about what each user liked and found difficult about the software. To help characterize user needs, we asked what kind of information they generally would like about animals.

4.3.3 Tasks

Our goals were to learn how people approach information retrieval in this domain, and to learn if they could use our visualization tool to find and understand the information. Thus, we designed a range of tasks, described below, that included general tasks appropriate for any layperson as well as specific tasks related to our users' coursework.

Two general, open-ended tasks assessed user preferences for information targets and strategies to reach them. Users were asked to use TaxonTree to find information about an animal of their choice. Depending on the strategy they took in

task 1 (searching or browsing), in task 2 they were asked to choose another target animal and use the other strategy to find it.

The other seven tasks were more specific and had a limited number of correct answers. These tasks assessed a user's ability to use most of the features of the software, to further examine their preferences for information-seeking strategies, and to examine the role of the interface in understanding the information. These tasks were as follows.

3) Find an extinct taxon.
4) Count how many extinct taxa you might need to know about for the final exam.
5) Find and name the taxon whose members are all united with the synapomorphy "Lactation."
6) What is the sister group to this group of lactating animals?
7) Now try searching on the common name, "dolphin." What do you notice about the results?
8) Find some victims or carriers of malaria.
9) What do you notice about these victims or carriers?

We recorded the information targets users chose and their initial strategy towards finding them (browsing or searching). We noted whether users completed a task and counted how many prompts we needed to give them so they could complete the task. Completing a task required a verbal indication that they had understood that the answer was onscreen. We noted what features of the tool elicited spontaneous positive reactions.

4.3.4 Results

4.3.4.1 Characterizing Users in This Domain

We noticed during testing sessions that some users were clearly interested in the content. These users verbally expressed prior content knowledge as they worked on tasks, or asked questions indicating curiosity about information beyond the task. For example, a user asked "Why isn't there anything about mosquitoes?" when looking at result for a search about malaria. In contrast, some users never departed from the tasks at hand. Figure 4.6 illustrates how often users offered extra information indicating content interest. Guided by Figure 4.6, high interest users were defined as those who spontaneously offered extra content information during at least 4 of the 9 tasks; the others were labeled low interest. Consistent with this categorization, three users labeled low interest in this way had stated they lacked interest in the Animal Diversity course.

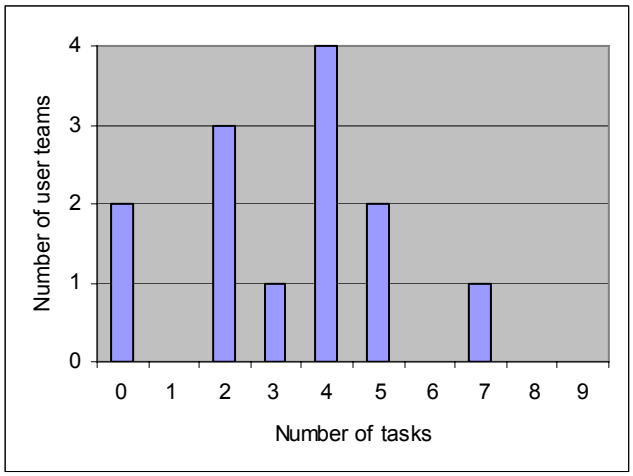


Figure 4.6 Distribution of 13 user teams based on the number of tasks in which they spontaneously offered additional biological information, indicating their interest in the domain.

Low interest and high interest users reported similar hours of experience with the application and similar levels of comfort with computing. Six of the 8 males in the study were in low-interest user teams; two of the ten females were in low interest teams.

Users tended to be interested in looking for animals using common names and above the species level e.g. “frogs” (Order Anura). Specifically, when asked to choose any animal to find, users gave 20 out of 26 initial targets as common names rather than scientific names. Many targets were clearly above the species level (14 of 26 targets) while 4 were ambiguous and 8 were species level. Fewer than half of the search targets (12 of 26 targets) were required course content. Low and high interest user teams had similar search targets. When asked the kinds of information they were interested in, 5 of 13 teams mentioned way of life (food habits, behavior, ecology). Five of 13 teams noted that they enjoyed learning unique characteristics of animals –

interesting superlatives or what sets an animal apart from others. Four user teams mentioned that they wanted only the information necessary to pass their course. Two user teams mentioned an interest in evolutionary relationships. One user wanted information to distinguish dangerous from harmless animals.

4.3.4.2 Usability

The interface seemed comfortable to users once they knew what features were available. Interaction with and interpretation of the nodes was apparently intuitive, because even users who had never used the program immediately began opening and closing nodes. Few users needed prompts explaining "more" triangles (shown in Figure 4.5), panning or zooming, node-clicking, or the ability to search. About 85% of the prompts we gave related instead to using our specific search categories and controlling the view options: how much of the tree was displayed (all nodes as they are opened, just the subset required for the course as they are opened, or all required nodes at once). Low interest users actually needed, on average, fewer prompts per session (4.3) to complete tasks compared to high interest users (7.4).

4.3.4.3 Searching and Browsing

Most users used both searching and browsing strategies together in at least one task. Only four user teams always used a single strategy within each task; three of these four teams were low interest users. Five of 13 teams browsed the tree before choosing a target or changed their target while browsing.

Most users preferred browsing the tree over searching. Only three of 13 teams used searching as an initial strategy – these were all high interest users who probably

had better ideas of what search terms to use. Even after a successful search, 10 out of 11 subject teams returned to a strategy of browsing. When asked why, they told us it was more fun than searching. For example, one user reported that "I could have done a search for birds but this is more fun." They also said that they wanted to refresh their memories, and that they didn't know exactly what to search on.

4.3.4.4 Task Completion

Users completed 92% of all tasks without prompts to interpret results shown onscreen.

Some tasks that asked for direct interpretation of the tree were very easy for the user teams. In task 5, 12 of 13 teams needed no prompts to correctly associate an attribute (the synapomorphy "lactation") with the name of the appropriate node. All but one team successfully completed task 6, identifying a sister group from a search result by opening a nearby node. Only three teams needed a prompt. Task 9, what do you notice about victims and carriers of malaria, was readily answered. Eleven of 13 teams gave an immediate answer relating to the tree structure (such as, the search results were in related branches of the tree). Task 4 asked "Count how many extinct taxa you might need to know about for the final exam." All but one team immediately moved from displaying all overview nodes, including 16 color-coded as extinct, to task completion (counting all the nodes that were color coded as extinct).

However, some tasks were clearly harder than others. Task 7 asked users to draw inferences from a search on the common name "dolphin." The task was considered complete if users gave at least one of two answers. First, there are many kinds of organisms whose common name includes the word "dolphin." Second,

organisms with a common name including the word “dolphin” appear in more than one very different branches of the animal kingdom. The first inference was immediately drawn by all but one of the 13 user teams. Such an inference would be nearly impossible to make quickly using a typical list of search engine results. The second, however, requires the more sophisticated inference requiring an understanding of biological relationships. This inference was only mentioned by five of 13 user teams.

Task 8 asked users to conduct a search for carriers or victims of malaria. This task posed particular difficulty because of its sensitivity to both the search terms chosen and the category of search that was run. Three users were unable to complete the search without more than two prompts. These plus an additional two user teams failed to look at the web pages in the results to be sure that their search terms were in the appropriate context. However, 8 of 13 teams did check for relevance.

Tree visualization helped users complete tasks. Task 9, “What do you notice about these victims or carriers?” could be completed either by interpreting a tree visualization of search results or by applying prior knowledge to those search results. For example, a user response such as “all of these victims seem to live in forests” would be an example of prior knowledge, while “all of these victims are in the vertebrate part of the tree” indicates use of the tree to interpret the results. Although three or four user teams did use prior knowledge, only one set of users used it as their first answer to the question. All the others gave tree information for their first answer.

Domain interest seemed correlated with domain expertise as high interest users but not low interest users tended to effectively use their prior knowledge to help solve tasks.

4.3.4.5 User Comments

User responses to open ended questions are summarized in Table 4.1. Users said that TaxonTree was usable and had desirable content (synapomorphies, external web links, and course information). Several mentioned explicitly that TaxonTree's visualization would be more useful to them than accessing the same information in their lecture notes or in the textbook.

Users had difficulty with unfamiliar features (search categories and view menu options). The other negative comments related to information quantity. Some users noted the difficulties inherent in displaying large amounts of information (font sizes and zooming problems). Some wanted more refined search results, while others felt that merely having so much information available to browse or search was daunting.

All but two user teams offered spontaneous positive comments while completing tasks. Visualizing search results in the tree structure elicited the most positive comments (6 user teams), along with the availability of web pages with more information (4). Four user teams also were excited about the ability to see an overview tree of the information necessary for their course.

What users liked	# of user teams
Easy to learn and use	9
Tree visualization	9
Synapomorphies	7
Ability to search different categories	6
How evolutionary history is presented	4
How tree is interactive	3
Links to external web sites.	3
Seeing which content required for their course.	3
What users found difficult	
Search categories were hard to understand	4
Font too small, especially when zoomed out	3
Too much information, too many search results	3
Had problems zooming	3
Had problems understanding view menu	3

Table 4.1 User comments to open-ended questions. Responses given by fewer than three user teams are not included.

4.4 Implementation Details

TaxonTree was implemented in Sun's Java 2, using Piccolo.Java [14]. It extended SpaceTree to accommodate domain specific requirements described in Section 4.2.1. TaxonTree was deployed from CD-ROM and via Java Web Start from a web page.

4.4.1 Database

SpaceTree loads the entire tree into a main memory (RAM) when users open a data file. This makes it possible to show the overview of the tree and to provide dynamic filters in real time. However, this approach does not scale well. Since our classification tree has about 200,000 nodes and each node has multiple attributes such as scientific name, common name, rank, and external web page addresses, TaxonTree uses database to store its data. Microsoft Access was used for the CD-ROM version and MySQL for the Java Web Start version, connecting with JDBC (Java Database Connectivity) in both cases.

As shown in Figure 4.7, the TaxonTree database consists of three tables; 1) taxon, 2) webinfo, and 3) synapomorphy. The taxon table contains basic information about a taxon such as id and names. In addition, it includes the information about the tree structure, such as parent id and index. If a taxon has the corresponding web pages, addresses are stored in the webinfo table. Similarly, the synapomorphies of a taxon are stored in the synapomorphy table. The complete description of three tables and their fields is provided in the appendix.

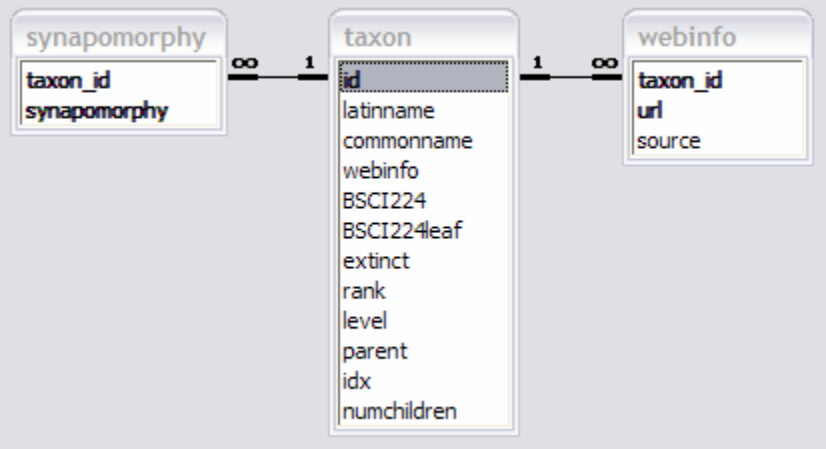


Figure 4.7 Database schema for TaxonTree

While TaxonTree is a domain-specific visualization, it can be applied to other taxonomies (or hierarchies). To show other taxonomies without modifying the current source code, only the taxon table is necessary; 1) the fields for the tree structure – level, parent, idx, and numchildren and 2) id and latinnames are required. Other fields should be filled with empty values, not null. By utilizing the webinfo table, links to the external web pages (up to 4 different web sites) can be provided. Furthermore, the link attributes of string types can be supported with the synapomorphy table.

4.4.2 Deployment

The program and the data were first distributed to students via CD-ROM. This introduced several complications. First of all, burning CDs takes time. Furthermore, it is very difficult to update either the program or the data once the CDs have been distributed. Lastly, some users had Macs and the CD-ROM version only ran on PCs. To address all of these problems, a web version was made available by using MySQL

and Java Web Start, providing a flexible and robust deployment solution for Java applications. It works with any browser and any Web server. It also works on both Macs and Windows. TaxonTree is now deployed at University of Michigan's Animal Diversity Web (http://animaldiversity.ummz.umich.edu) for the public.

It is possible to use Access with a web server by setting up a DSN or a DSN-less connection to a machine on the network where the database would be held. However, this is uncommon. Instead, MySQL was made accessible over the Web even though data has to be exported from MS Access.

4.4.3 PTaxonViewNode and PSynapViewNode

PTaxonViewNode is inherited from PTreeNode and is used to represent nodes on the screen. PTreeNode contains information for the basic tree structure, such as parent and children. PTaxonViewNode contains additional tree structure, such as level and index, and application specific information, such as id and names. PSynapViewNode is inherited from PTaxonViewNode and is used to represent nodes with synapomorphies. It contains an instance of PSynapViewHelperNode, which actually draws lines and texts.

4.5 Discussion

While TaxonTree offers only incremental extensions to SpaceTree from the implementation point of view, it shows how making a domain-specific version of a generic visualization often requires a significant number of minor changes. More substantially, the qualitative user study of TaxonTree evaluated the software in a more natural domain-specific context than the prior SpaceTree study. The study

provides further evidence for the value of interactive tree visualization and integrated searching and browsing in information retrieval and understanding.

4.5.1 Interactive Visualization

TaxonTree shows that interactive tree visualization can be applied to the biodiversity domain. The style of tree diagram (shown in Figure 4.8a) that biology students are currently familiar with is different from TaxonTree's style (shown in Figure 4.8b) in the following ways: 1) it shows animal names only at leaves; 2) internal nodes are labeled with brackets outside of the tree; 3) every branch has a fixed angle.

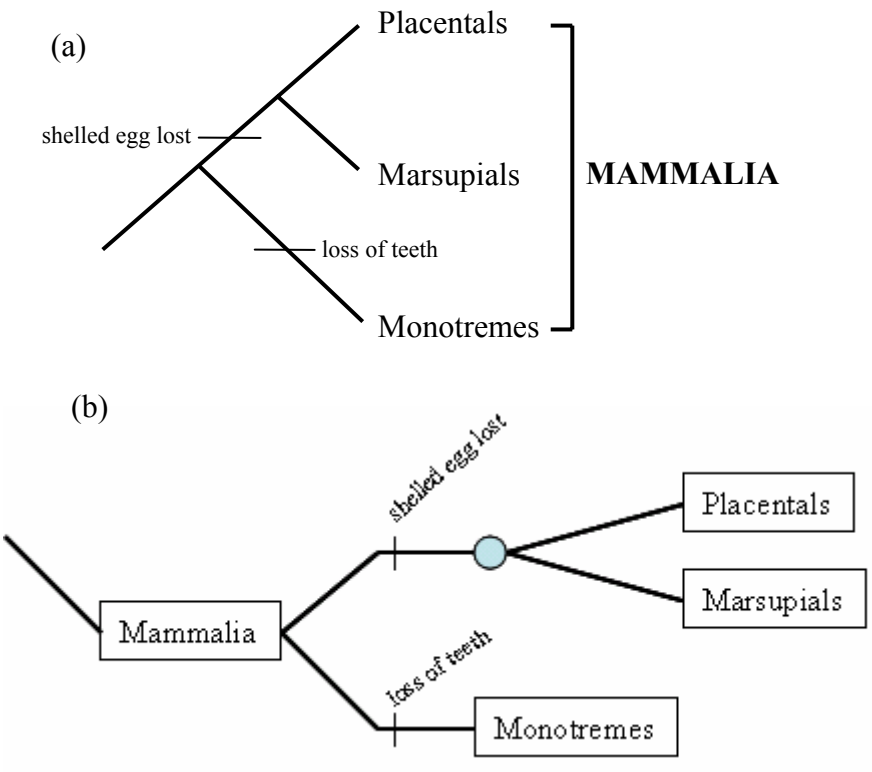


Figure 4.8 Example of tree diagrams. (a) Style currently familiar to biology students. (b) TaxonTree style.

Despite these differences, users easily understood TaxonTree's tree structure. Our combination of interaction style and tree representation could therefore be useful in other domains, but a closer look at the trade-offs of the different representations is warranted.

4.5.2 Integrated Searching and Browsing with Animation

Most users preferred browsing the tree over searching. Users gave "it is fun" as one of several reasons they preferred to browse the tree rather than search it. Perhaps they enjoyed the animated interaction. If so, making interaction fun may be another benefit of animation, particularly in often tedious domains.

TaxonTree seamlessly integrates searching and browsing. It is beneficial to present search results in an interactive classification tree that shows the biological context. Users easily interpret the search results, quickly using the tree structure to discover the quantity of biologically unique results. They often made more sophisticated inferences about relationships among the search results, which is nearly impossible by using a typical search results list of. They also carried out additional browsing, giving them a better sense of the search results. Even though users preferred to browse rather than search, they often employed both strategies for the same task, especially when they had partial knowledge of names or relationships – a situation likely to be common among biologists.

4.5.3 Incremental Exploration

Despite the fact that TaxonTree shows only a small subset of the tree at any time, it still works well to help users navigate and develop an understanding of the full tree.

This is largely because TaxonTree always provide some context for users by highlighting the path to the root and by displaying some surrounding nodes. In addition, because all transitions are smoothly animated, users can perceive the relationship between different states.

4.5.4 Adopting TaxonTree

As mentioned above, TaxonTree is now deployed at University of Michigan's Animal Diversity Web for the public. In this domain, hyperbolic tree browsers have been considered the most sophisticated tree visualization. For example, the Green Plant Phylogeny Research Coordination Group (GPPRCG) uses hyperbolic trees to show two different trees for the Deep Green project (http://ucjeps.berkeley.edu/bryolab/GPphylo/). The Glasgow Name Server (http://darwin.zoology.gla.ac.uk/~rpage/MyToL/www/) also uses hyperbolic views to show the NCBI classification.

TaxonTree has always been well received by biologists. Even though formal interviews were not conducted, biologists had reactions consistent with the study result when TaxonTree was introduced to biologists. In addition to the rich content, they like the animations and the ability to search using several types of fields. Furthermore, they consider TaxonTree as an alternate navigation method for their websites. So, biologists are now showing signs of adopting TaxonTree as a standard tree browsing technique. In addition to ADW, the Phylogeny of Lepidoptera (LepTree) project (http://www.leptree.net) will be using it. The California Academy of Sciences' AntWeb (http://www.antweb.org), Cyberinfrastructure for Phylogenetic Research (CIPRES -- http://www.phylo.org) and Science Environment for Ecological

Knowledge (SEEK -- http://seek.ecoinofrmatics.org) projects are also showing interest.

Chapter 5

TreePlus: Visualizing Graphs as Trees

I developed an interactive graph visualization called TreePlus, which enables users to iteratively explore a graph by starting at a node and then incrementally expanding and exploring the graph. TreePlus transforms a graph into a tree plus cross links (i.e. the additional links that are not represented by the spanning tree) and uses visualization, animation and interaction techniques to reveal the graph structure while preserving readability of the labels. In contrast to the more familiar overview techniques [123], which are effective at (but also limited to) revealing overall structure and the existence of clusters or bridges, our technique addresses the needs of users to explore parts of the graph in detail and rapidly read labels to analyze the meaning of relationships.

As I described in section 2.3, a number of researchers have already visualized graphs as trees. They used various types of tree layouts, such as hyperbolic, radial, and even Treemaps. Each showed the potential of the tree layout approach but had limitations. For example, H3 would not be useful if we cannot extract a meaningful spanning tree structure. The radial approach by Yee *et al*. might be cluttered for highly connected graphs since it shows all the links at once. The Treemap approach by Fekete *et al*. does not work for graphs that have cycles since nodes were duplicated to show cross links. The EROS system was specifically designed for RDFS. More importantly, for most of these systems, node readability was an issue especially for the highly connected graphs.

5.1 Our Approach

5.1.1 Plant a Seed and Watch It Grow

A useful guide to designing advanced graphical user interfaces is Shneiderman's Visual Information-Seeking Mantra [123]: "Overview first, zoom and filter, then details-on-demand." An overview of the entire data collection helps users find interesting patterns, clusters, outliers, and features. However, it is notoriously difficult to generate a good overview of large graphs. Furthermore, Blythe *et al*. demonstrated that "there is no best layout" and that the task and graph characteristics influence which layout will do better [17].

For cases where users are more interested in the local structure of the graph, rapid browsing, and easy reading of labels, we propose an alternative guiding metaphor: "Plant a seed and watch it grow." This enables users to start with a specific node and incrementally explore the graph, avoiding complexity until it is necessary. Zoom and filter and details-on-demand are still useful, but overviews remain localized and on-demand. Furthermore, this approach can be used to complement overview-first approaches.

A similar approach was very recently used in other systems. Heer and Boyd opted for "start with what you know, then grow" and applied it to a traditional graph layout [65]. McGuffin and Balakrishnan focused on visualizing only part of a graph by using "focus" as a temporary root to visualize genealogical graphs [94].

5.1.2 Design Goals

There are always trade-offs when designing an interactive visualization. This section describes the rationale of our design goals.

5.1.2.1 Take advantage of human perception of trees

Our previous work on SpaceTree and TaxonTree suggested that interaction with and interpretation of node-link tree structures poses little difficulty for novice users and therefore interactive tree visualizations can be used for a broad audience [88, 105, 109].

5.1.2.2 Make as many nodes readable as possible

Many tasks involve reading the labels of nodes. For example: find and review 1) the nodes adjacent to a node; 2) the nodes accessible from a node; 3) the nodes adjacent to two given nodes; 4) the shortest path between two nodes; 5) the nodes having a specific attribute value; 6) the nodes connected only by certain types of links. Other examples include: list all the labels in a sub-graph and follow a path. For each of those tasks, users need to read labels to make sense of the data. Users will scan names in social network data to see if they know anyone, determine if there seem to be more women than men, or look for Asian-sounding names. They will also review color codings and icons associated with the nodes to judge the distributions of attributes.

5.1.2.3 Maximize stability of layout

Stability is a very important aspect of interactive layout algorithms. In fact, one of the main problems of the commonly used force-directed layouts is that they are highly unstable (i.e., the same graph might get drawn differently depending on initial conditions that are not under users' control) [66]. To make the tree layout completely stable, two approaches are possible. First, the structure of the tree could be fixed once it was first extracted from the graph (Figure 5.2). The main drawback to this approach is that cross-linked nodes would often be very far away from each other. Second, adjacent nodes could be placed close to each other by duplicating the cross-linked nodes (Figure 5.17). Although this approach works well for graphs that have an intrinsic tree structure with a modest number of cross links, it is less suitable for highly connected graphs. Furthermore, the tree will grow forever if the graph has cycles. Instead, TreePlus follows a third approach where the tree structure is modified when users make a selection by moving adjacent nodes close to the selected node. Although this approach is not completely stable, it is predictable. Changes are limited, and if users happen to traverse the same series of nodes in two different sessions, the resulting layouts will be exactly the same.

5.1.2.4 Offer preview before committing

Incremental exploration requires users to make decisions about where to go based on the information they have at any given time. To increase the "information scent" available, clicking a node provides a preview of what nodes are connected to it. The node is not expanded until users double click on it.

5.1.2.5 Provide multi-step animations so users can follow changes

As users incrementally navigate a structure, it is necessary to change the layout. Although animated transitions help users remain oriented [85], they can be too complex or too fast to be accurately perceived. Inspired by our successful experience with SpaceTree [109] our approach was to decompose the layout change into meaningful steps.

5.2 Description of the Interface

TreePlus combines a tree-style layout, an adjacent nodes preview, and multiple custom interaction techniques to explore graphs that can be directional and cyclic. Animation, zooming and panning, and integrated searching and browsing help users understand the graph. Users navigate the tree by double clicking on nodes in the tree browser on the left (Figure 5.1), and preview adjacent nodes on the right by single clicking on a node to bring it in focus. TreePlus uses a classical tree layout by Walker [136]. The children for each node are left justified, so it is easy to scan, read, and count them. Nodes can be grouped and sorted by various ordering criteria.

TreePlus is described using a food web dataset [113]. A food web describes the feeding relationships among organisms in a community. Most animals are part of more than one food chain (or path of nodes) and eat more than one kind of food. These interconnected food chains form a complex food web. Food webs are directional and can be cyclic. Unlike datasets used in many previous tree-layout graph visualizations, food webs have no intrinsic tree structure, so they pose a greater challenge for a tree-like visualization.

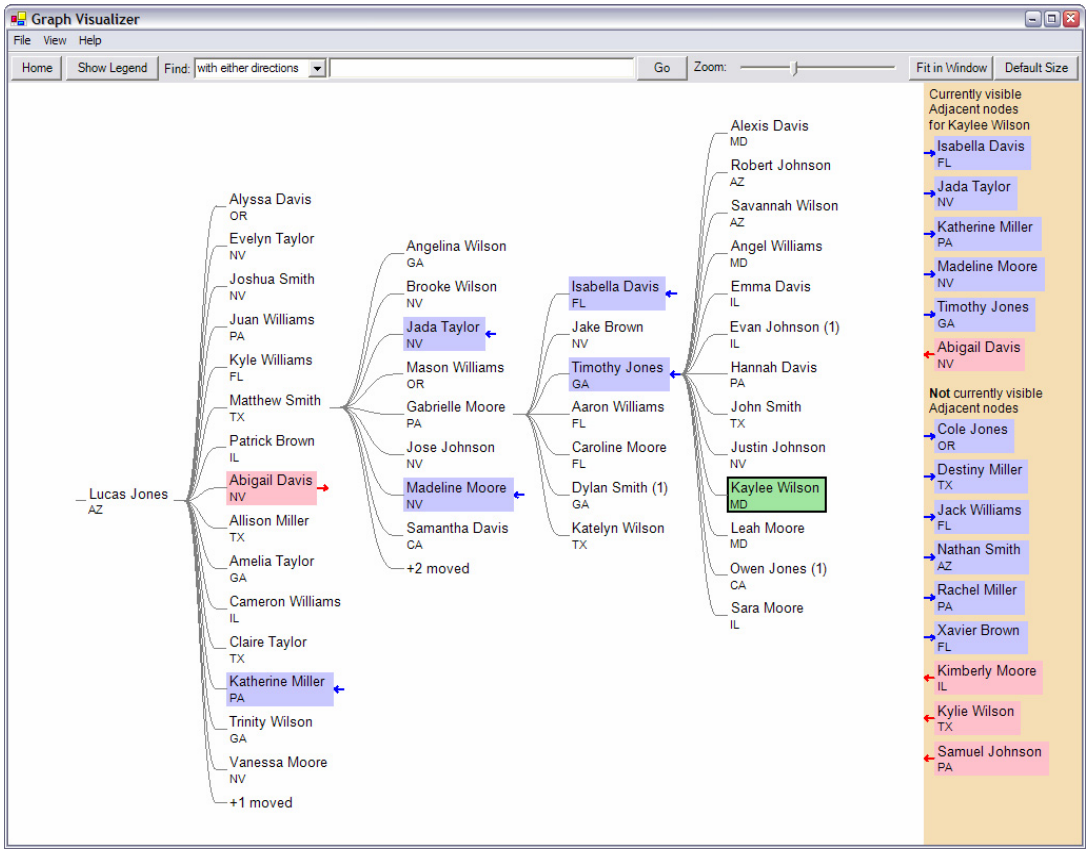


Figure 5.1 TreePlus with the low density dataset used in the user study. A single click on any node (here "Kaylee Wilson") highlights adjacent nodes already present in the tree and lists new names in the preview panel on the right. Color indicates the direction of the link. Double-clicking on a node expands the tree by adding new adjacent nodes and moving existing nodes as needed [see video demonstration at http://www.cs.umd.edu/hcil/treeplus].

5.2.1 Transforming Graphs into Trees

Graphs are transformed into trees by extracting a spanning tree. The first step is to identify a root. Domain specific default roots might exist. For example, gene ontologies have an explicit root and a canonical tree structure. Web sites have a

home page. If the graph does not have an explicit root, two possible defaults are provided as suggested in [18]: 1) the node that has the most links; and 2) the node whose cumulative distance to all other nodes is minimal. Users can change the root at any time; and it can be saved in the preferences. TreePlus builds a spanning tree from the root by a breadth-first search, ignoring the direction of links.

5.2.2 Showing Hidden Graph Structure

When visualizing graphs as trees, many cross links will inevitably become hidden, particularly in highly connected graphs. The success of a tree layout approach depends on how well the system represents those cross links.

Figure 5.2 shows a preliminary tree-layout based visualization tool, developed at the HCIL by Jesse Grosjean. By default, only primary links that form a spanning tree are shown. When users move the mouse over a node, cross links are shown in red between two nodes. One of the main problems with this approach is that two nodes connected by cross links are often very far apart. This means users will have to follow a long link to see the connected node. Furthermore, many cross links make the screen more cluttered. Therefore, TreePlus highlights connected nodes without showing the links.

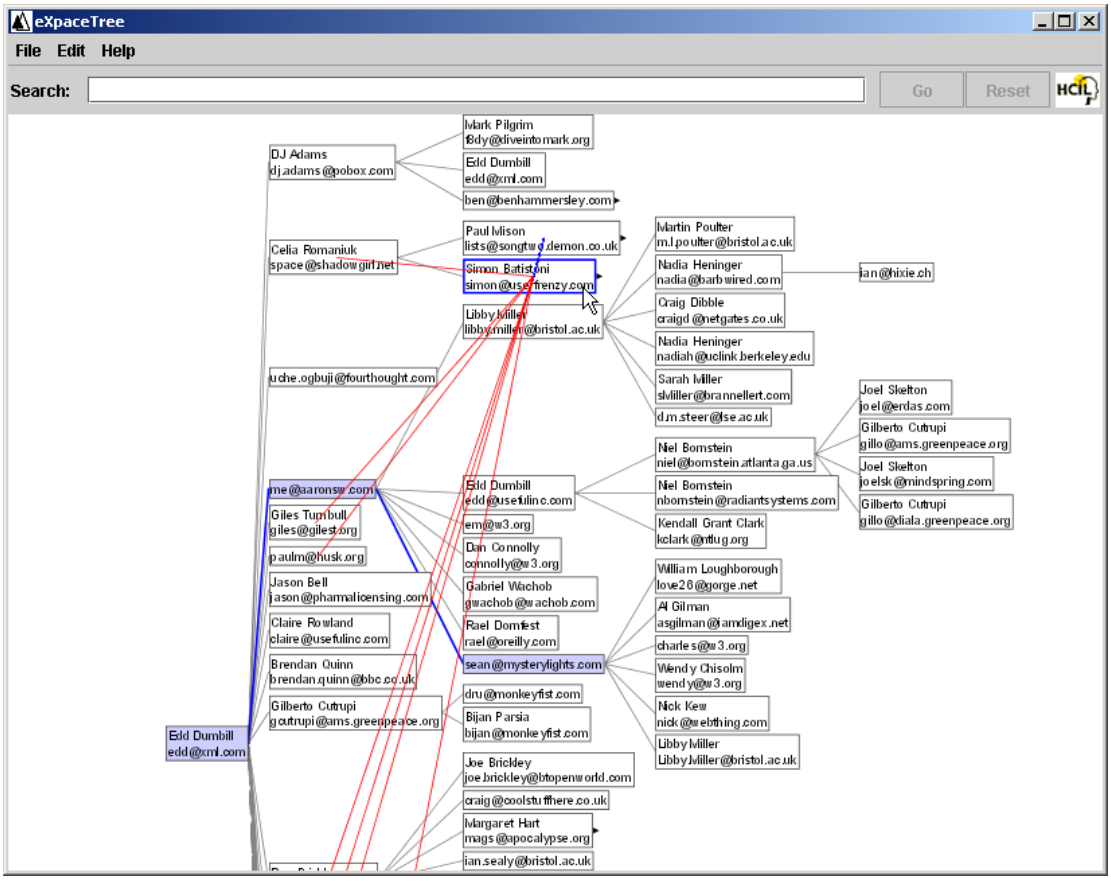


Figure 5.2 SpaceTree Extension to visualize graphs. Cross-linked nodes of the focus node are connected by red lines.

Since screen space is limited, some nodes will be located off screen. With highlighting, it is very difficult to see whether there are more off screen connected nodes. To address this issue, TreePlus previews adjacent nodes when a node is focused. This enables users to see all of the connected nodes of the focus node in one place (adjacent nodes preview panel) instead of looking around the whole screen space. TreePlus also moves adjacent nodes close to the selected node by updating the tree structure when a node is opened. To help users follow the changes, we carefully animate the transitions.

5.2.2.1 Highlighting and Preview of Adjacent Nodes

When users click on a node, the node gets the focus, indicated by a green background and thick border. In the example of Figure 5.3, "stripe-headed tanager" has the focus; a list of its five adjacent nodes is shown in the preview panel on the right. Three of these nodes already appear in the current tree display, and are therefore highlighted in color on the tree. Users can see that "fruits," "red-tailed hawk," and "broad-winged hawk" are directly connected to "stripe-headed tanager" (as indicated by the highlighting) and to "rat" (as indicated by the tree layout). Changing the focus rapidly by using the arrow keys to go up or down a list of nodes allows users to systematically review cross links that are not apparent in the tree layout.

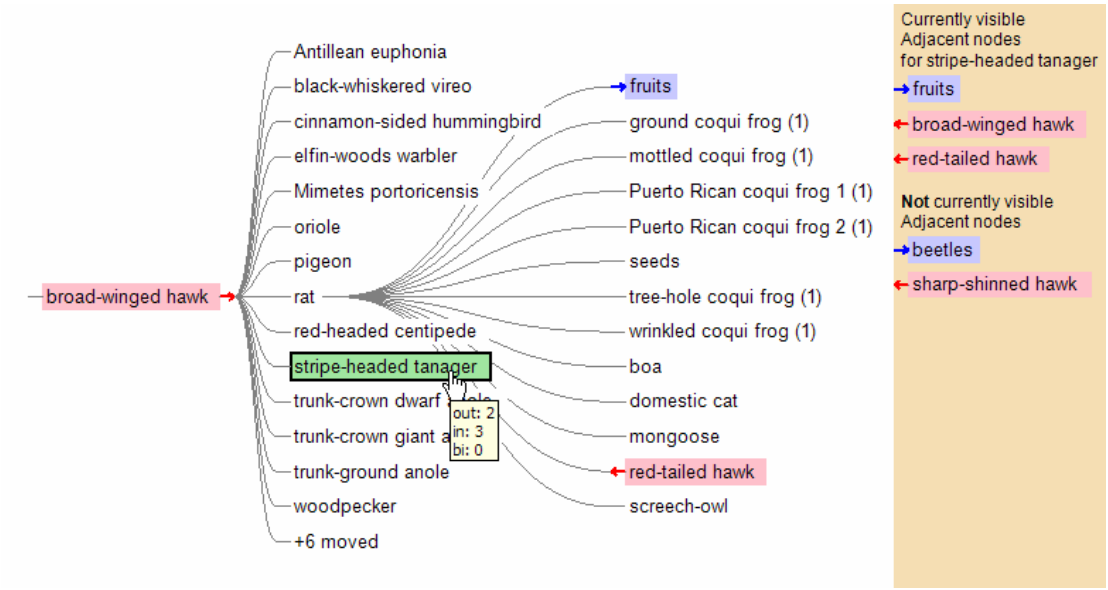


Figure 5.3 "broad-winged hawk" was set as the root, and users selected "rat" which added all its adjacent nodes to the tree. A single click on "stripe-headed tanager" gives it the focus and shows a preview of its adjacent nodes in the preview panel on the right. The adjacent nodes already present in the tree are highlighted in the tree revealing that "fruits," "red-tailed hawk," and "broad-winged hawk" are connected to both "rat" and "stripe headed tanager." Color indicates link direction.

The color of the node background and arrows indicates the direction of links relative to the focus node. TreePlus uses the color blue for outgoing links, red for incoming links, and purple for bidirectional links. For example, in Figure 5.3, red nodes (e.g. "broad-winged hawk") eat the "stripe-headed tanager" while the "stripe-headed tanager" eats blue nodes (e.g. "fruits").

5.2.2.2 Animated Update of the Tree Structure

When users double click on a node (i.e., make a new selection) the tree is expanded to include all the adjacent nodes. For example, when users select "stripe-headed tanager," two new nodes are added to the tree ("beetles" and "sharp-shinned hawk") while the nodes "fruits" and "red-tailed hawk" move from being children of "rat" (Figure 5.3) to being children of "stripe-headed tanager" (Figure 5.4). This change corresponds to the assumption that users are more interested in the node they last opened. The node "broad-winged hawk" remains where it was as parent in the path.

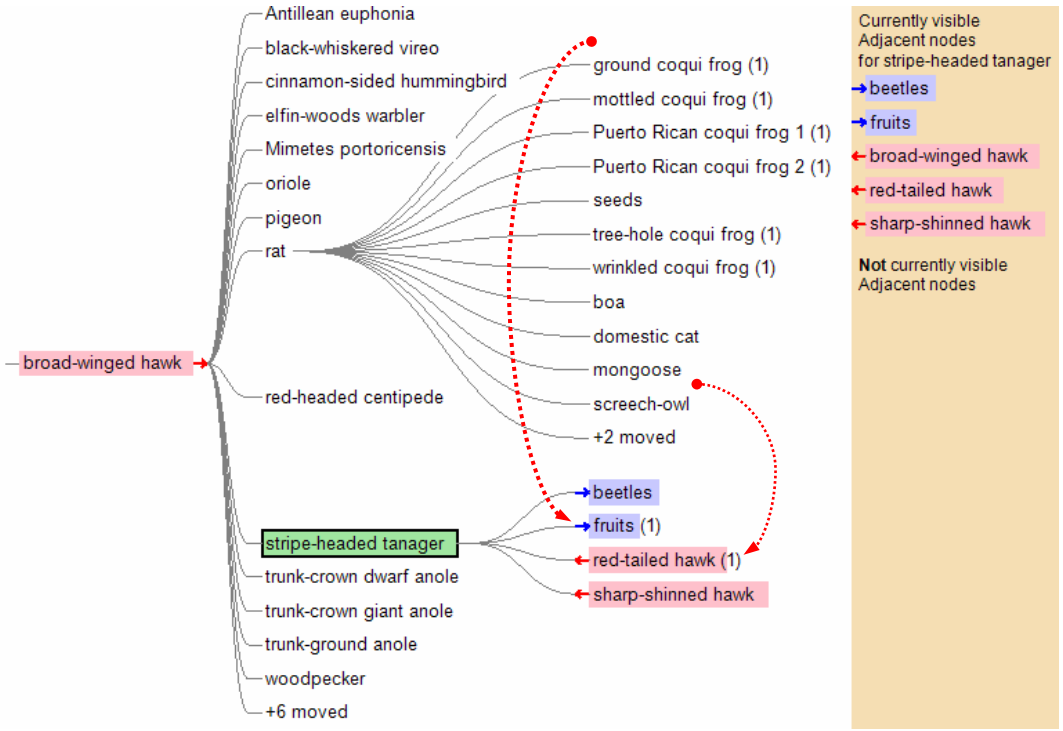


Figure 5.4 Once users open "stripe-headed tanager" by double clicking, the tree is expanded to shows all its adjacent nodes as its children and parent (the red dotted arrows were added to this figure to illustrate node movement).

To help users maintain context, the tree is animated to its new layout (e.g. from Figure 5.3 to Figure 5.4) in three steps. First, TreePlus makes room for the new nodes by translating parts of the tree and creating empty space. Next, the nodes that need to move within the tree structure (i.e., "fruits" and "red-tailed hawk" in our example) move to their final position as the children of the new selection. Finally, the nodes of the preview panel move to their position in the tree. Once the animation is over, the preview panel is refreshed to reflect that all nodes are now visible. (Only a video can adequately illustrate this interaction; please see our video demonstration at http://www.cs.umd.edu/hcil/treeplus.)

When nodes have to move within the tree structure, TreePlus leaves a trace to indicate that a move took place. The label "+2 moved" under "rat" in Figure 5.4 indicates that 2 nodes have moved. Bringing the cursor over this "+2 moved" label highlights two nodes that have moved ("fruits" and "red-tailed hawk"). Users can also see that "fruits" and "red-shinned hawk" have moved once because of the "(1)" on the right side of the labels. When this number grows large it indicates that the node is linked to many of the nodes users had selected during their exploration. To be reminded of what those nodes were, users can single-click on the node to bring it in focus.

5.2.2.3 Visual Hints of the Graph Structure

During graph exploration users may want to follow a path based on the attributes of the nodes, such as the number of outgoing links. In TreePlus users have the option to preview how fruitful it would be to go down a path. Color bar graphs placed below the nodes represent how many organisms are reachable in each direction (Figure 5.5).

Users can also see how many levels they can go in each direction by counting the number of white ticks. For example, if users follow the "wrinkled coqui frog" path, they will reach the end of the food chain after opening up to two levels. Similarly, users will reach the start of the chain after opening up to three levels.

Figure 5.5 Colored bars give a preview of how fruitful it would be to follow a path in each direction. "broad-winged hawk" is a start of a chain since it does not have a red bar (nothing eats it). "fruits" is an end of a chain since it does not have a blue bar (fruits eat nothing).

5.2.3 Sorting

Children of each node are depicted with a vertical list. By default this list can be sorted by name (nominal attribute), the direction of the links relative to the parent (categorical), and the number of links (quantitative). Other application-dependent sorting attributes can be added. Within each category, nodes are sorted by name in alphabetical order. The nodes in the preview panel are sorted by the same attribute.

5.2.4 Search

TreePlus provides support for search. Typing a word and pressing the "Go" button in the containing application invokes the Search function of TreePlus. Then, TreePlus displays the search results colored in beige and restricts the view to the nodes relevant to the search results (Figure 5.6). In order to get a valid shortest path, a desired link

direction can be specified. To find connections between two arbitrary nodes users can search for one node and set it as root, then search for the second node.

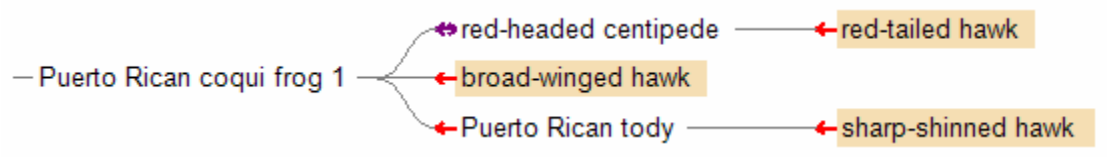
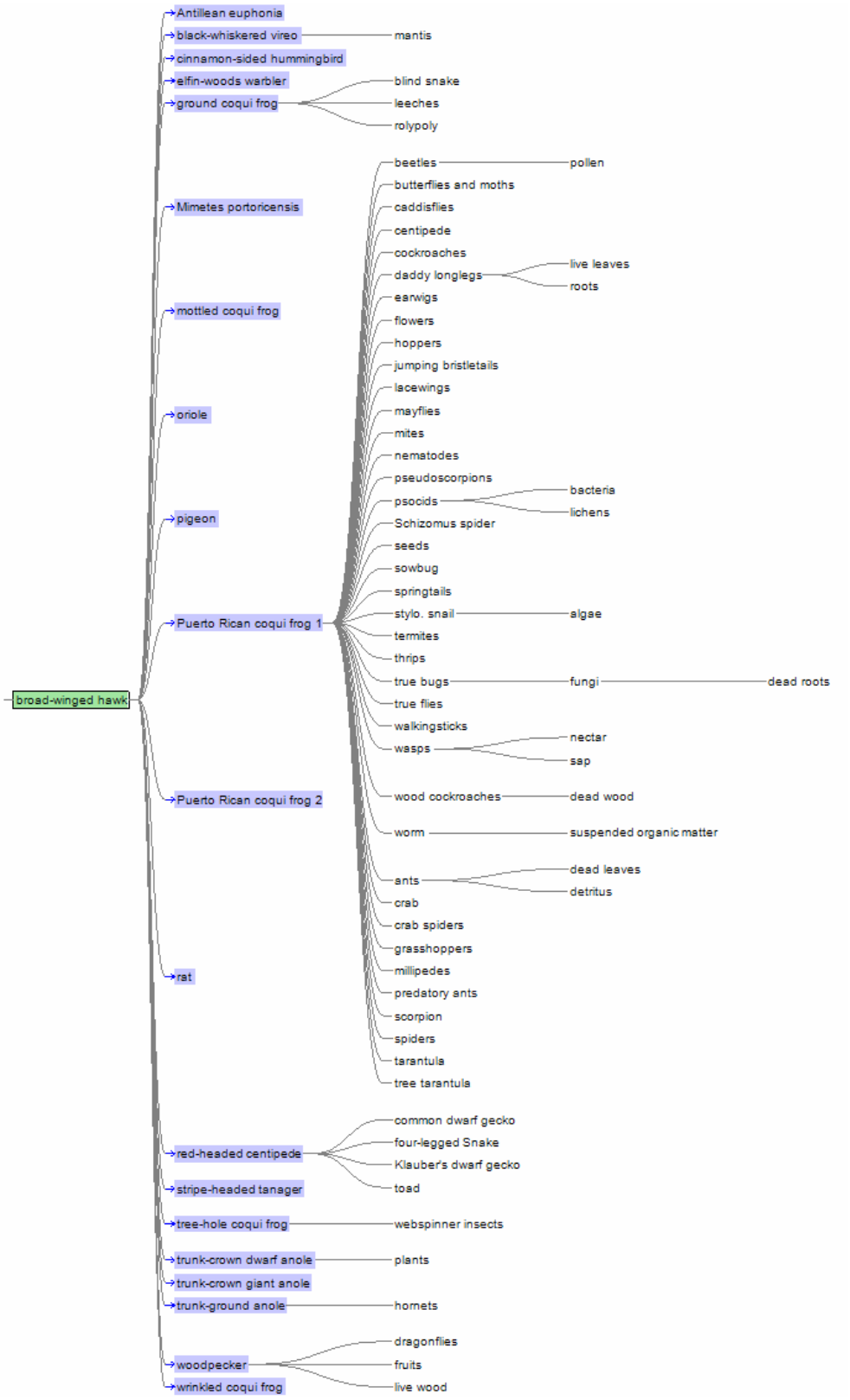


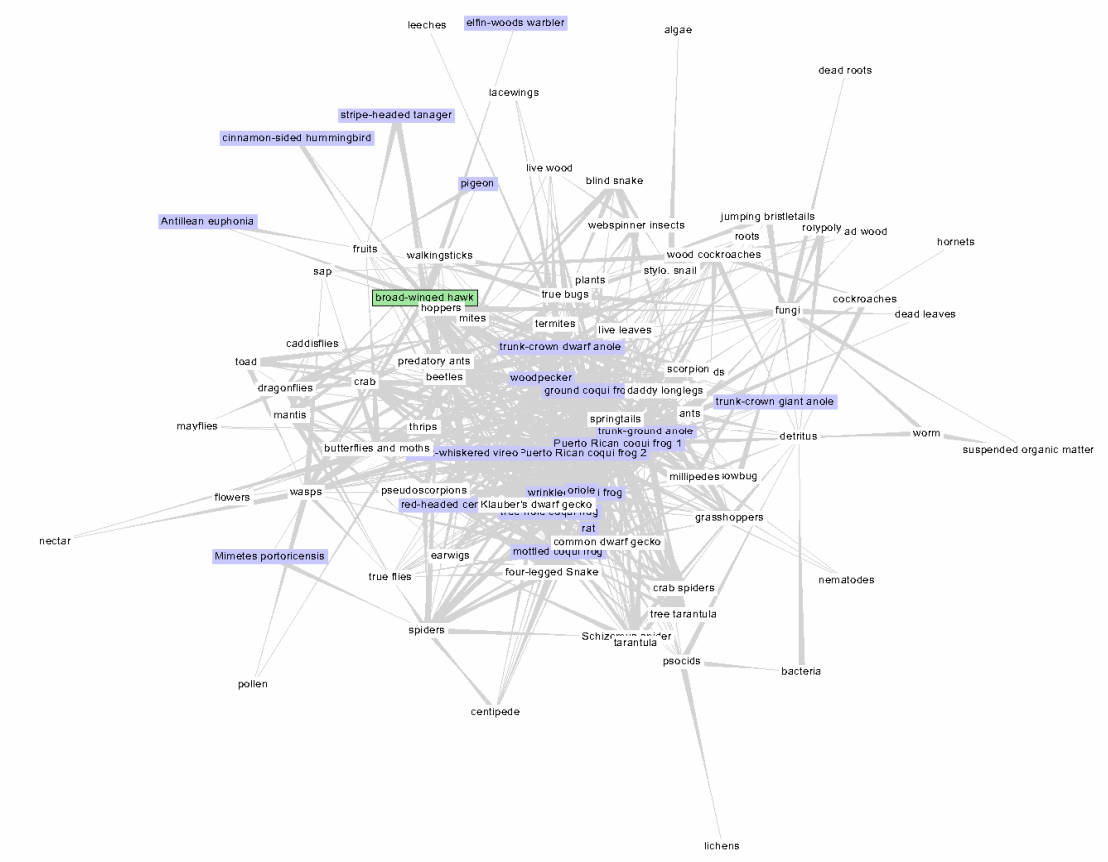
Figure 5.6 A search for "hawk" with "Puerto Rican coqui frog 1" set as root shows that the frog is eaten by the "broad-winged hawk" and indirectly by the "red-tailed hawk" and "sharp-shinned hawk."

5.2.5 Partial Overview

Even though TreePlus was not aimed at providing complete overviews of large graphs, it can generate partial overviews by automatically expanding the tree from any starting node, for each direction, at a selected level of expansion. For example, starting with "broad-winged hawk" and expanding as far as possible (here level 4) with outgoing links, 89 nodes and 537 links can be reached (Figure 5.7). The tree overview allows users to rapidly scan labels and estimate the number of nodes and the path lengths. Users can pan to read all labels and zoom out to see everything at once. Clicking on a node and then navigating with arrow keys allows users to get a quick idea of the graph structure. For comparison, we show in Figure 5.7, below the TreePlus overview, the same partial overview with a traditional graph layout (using GraphPlus, see user study section), here zoomed out to fit the narrow column width of the paper. TreePlus may not show all cross links at any given time but makes other aspects of the graph more visible.


broad-winged hawk
Antillean euphonia
black-whiskered vireo
mantis
cinnamon-sided hummingbird
elfin-woods warbler
ground coqui frog
blind snake
leeches
rolypoly
Mimetes portoricensis
mottled coqui frog
oriole
pigeon
Puerto Rican coqui frog 1
beetles
pollen
butterflies and moths
caddisflies
centipede
cockroaches
daddy longlegs
live leaves
roots
earwigs
flowers
hoppers
jumping bristletails
lacewings
mayflies
mites
nematodes
pseudoscorpions
psocids
bacteria
lichens
Schizomus spider
seeds
sowbug
springtails
stylo. snail
algae
termites
thrips
true bugs
fungi
dead roots
true flies
walkingsticks
wasps
nectar
sap
wood cockroaches
dead wood
worm
suspended organic matter
ants
dead leaves
detritus
crab
crab spiders
grasshoppers
millipedes
predatory ants
scorpion
spiders
tarantula
tree tarantula
Puerto Rican coqui frog 2
rat
red-headed centipede
common dwarf gecko
four-legged Snake
Klauber's dwarf gecko
toad
stripe-headed tanager
tree-hole coqui frog
webspinner insects
trunk-crown dwarf anole
plants
trunk-crown giant anole
trunk-ground anole
hornets
woodpecker
dragonflies
fruits
live wood
wrinkled coqui frog

(a)



(b)

Figure 5.7 Partial overviews of the graph consisting of the reachable nodes from "broad-winged hawk" with outgoing links. It contains 89 nodes and 537 links. (a) TreePlus layout: every path from the root to nodes is a valid shortest path between the root and the node (b) The more traditional graph layout of GraphPlus for the same data.

In contrast to other graph visualizations aimed at providing complete overviews that reveal clusters and connected components, TreePlus focuses on providing local overviews. Note that if a complete overview with both directions is generated, there may not be a valid path from the root to some of the nodes. For example, in a spanning tree with "rat" as a root, the path, "rat"—"mongoose" —

"plants" is not a valid food chain because "mongoose" eats "rat" and "plants." In other words, though TreePlus builds a spanning tree by a breadth-first search, some of the paths from the root to nodes are not meaningful shortest paths.

5.3 Usability Study

To identify any major usability issues with TreePlus, a preliminary usability study was conducted with two biologists and three computer science graduate students. The biologists used the sample food web whose density was about 25% and the computer scientists used a randomly generated graph of 200 names and 3,600 links (30% density). Both of them were directed graphs. For the tutorial, all participants used the same dataset, another randomly generated directed graph of 100 names and 900 links (30% density). The time to complete each task was measured by using a stopwatch. The number of wrong answers and the number of times users gave up were also counted. Each session lasted about an hour.

5.3.1 Procedure

Participants read through the tutorial and played with the program to understand basic features of TreePlus. They were allowed to ask any questions about the program. While there was no time limit for the tutorial, participants spent about 22 minutes on average. The biologists were asked to conduct 13 tasks and the computer scientists were asked to do 14 tasks, because the food web data did not have any of the attributes needed for one of the tasks. After the session was over, participants filled out a questionnaire.

5.3.2 Tasks

The following tasks were provided to three computer science graduate students. The same types of tasks for the food web dataset were given to two biologists.

- Adjacency (direct connection)
  1) List the names directly connected to "Charles."
  2) List the names starting with "B" in the set of nodes directly connected to "Sierra."
  3) List the names directly connected to "Jaden" with incoming links to "Jaden."
- Accessibility
  4) List the names accessible by outgoing links from "Sean" within distance 2.
  5) List the names by incoming links to "Hannah" within distance 2.
  6) Is "Owen" accessible by outgoing links from "Adrian?"
- Common connection
  7) List the names connected to both "Hunter" and "Jayden."
  8) List the names connected to all three of these nodes: "Jenna," "Natalie," and "Hannah."
- Browsing
  9) Follow a path: "Adrian" → "Blake" → "Julia" → "Landon" → "Kylie" → "Trinity."
  10) Read the path you just followed in a reverse order.
- Etc
  11) Find a shortest path from "Patrick" to "Blake."

12) Among the people who are connected to "Jenna" by incoming links, who has the most incoming connections?

13) Among the people who are adjacent to "Jessica," how many of them were born in "NY?" (This task was not available for the food web dataset.)

14) Identify a cycle of length 3 which includes "Adrian."

5.3.3 Results

There were 11 incorrect answers and 7 give-ups provided out of 68 tasks across participants. For task 1, one participant forgot to include the parent of the node as an adjacent node. For task 4, three participants listed the organisms of distance 2 instead of within distance 2. In other words, they did not include the adjacent nodes in the answer. We believe that participants misunderstood task descriptions. For task 6, two participants only checked the adjacent nodes instead of accessible nodes. For task 7, two participants did not answer all commonly connected people. For task 11, two participants ignored the link directions. Sometimes participants were not able to come up with a strategy to complete tasks. One participant failed task 7, two participants were not able to complete task 8, and only one participant was able to complete task 14 with help from the experimenter. Incorrect answer times were not included in the task time analysis. Task completion time corresponds to the errors and give-ups as can be seen from Figure 5.8.

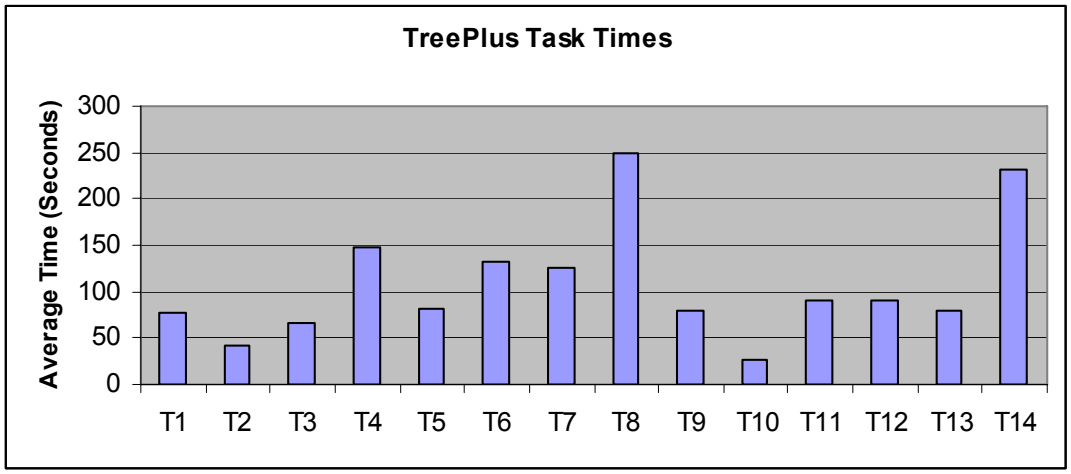


Figure 5.8 Average task times using TreePlus

For user satisfaction questionnaire, all ratings were placed on a 1-9 Likert scale, with 1=Disagree and 9=Agree. The average ratings are shown in Table 5.1 below. On average, users felt it was not easy to correct mistakes. In contrast to CS graduate students, biologists did not think the preview was useful. While participants thought the system was not quite easy to use, they felt comfortable using it. They also felt that labels were easy to read and the highlighting, colors, and arrows were helpful.

	B1	B2	C1	C2	C3	Average
Overall, the system was easy to use	5	7	4	8	4	5.6
I felt comfortable using the system	8	7	7	8	3	6.6
Animation of the system is easy	6	3	8	7	NA	6
Adjacent nodes preview is useful	1	1	7	7	9	5
Labels are easy to read	9	3	7	9	8	7.2
Highlighting and colors are helpful	9	5	7	8	9	7.6
Arrows are helpful	3	8	8	7	9	7
Correcting your mistakes are easy	5	3	3	6	7	4.8

Table 5.1 Average Likert scale ratings for TreePlus, using the scale of 1=Disagree, 9=Agree

5.4 Controlled Experiment

The goal of the controlled experiment was to determine if TreePlus could outperform a classic graph visualization for certain tasks and graph densities. A 2x2x6 (2 interfaces with 2 densities of the graphs by 6 tasks) repeated-measure, within-subject design was used. To control for the effect of order and learning, two sets of graphs with equivalent tasks of similar difficulties were prepared. The order of presentation of the interfaces and the set of graphs was counterbalanced. The order of densities (from low to high) and the order of tasks (from simple to complex) were kept

constant. Four dependent variables were collected in this study: Completion Time, Success Rate, Error, and User Confidence.

Success rate is the percentage of tasks correctly answered. Error was computed as the difference between the correct answer and participant response (which was possible only for two tasks requiring users to count). Participants indicated their confidence in their answer for four tasks, and completed satisfaction and preference questionnaires.

5.4.1 Interfaces: TreePlus and GraphPlus

GraphPlus (Figure 5.9) lays out nodes using the TouchGraph [129] layout algorithm. TouchGraph is a commercial product but one of the early versions provided an open source version of the layout algorithm. This algorithm was chosen because it was designed for incremental exploration, and does a good job at avoiding occluded labels by repositioning nodes during layout. When users select a node, a new layout is recomputed to accommodate newly introduced nodes but the TouchGraph layout algorithm tries to minimize the movements of the nodes that were already displayed. In TouchGraph, links are elongated triangles that show direction (the base of the triangle is the start and the apex is the end). However, GraphPlus was implemented to have similar features as TreePlus: it uses the same red-blue color direction coding and dynamic highlighting to reveal adjacent nodes. The amount of time for multi-step animations (1.8 seconds per complete transition) was also controlled. On the other hand, we considered the preview panel a major novel element of TreePlus and did not provide it in the GraphPlus control interface. The goal was to compare the

TreePlus interface with a state-of-the-art graph visualization interface, not to solely compare the layout algorithms.

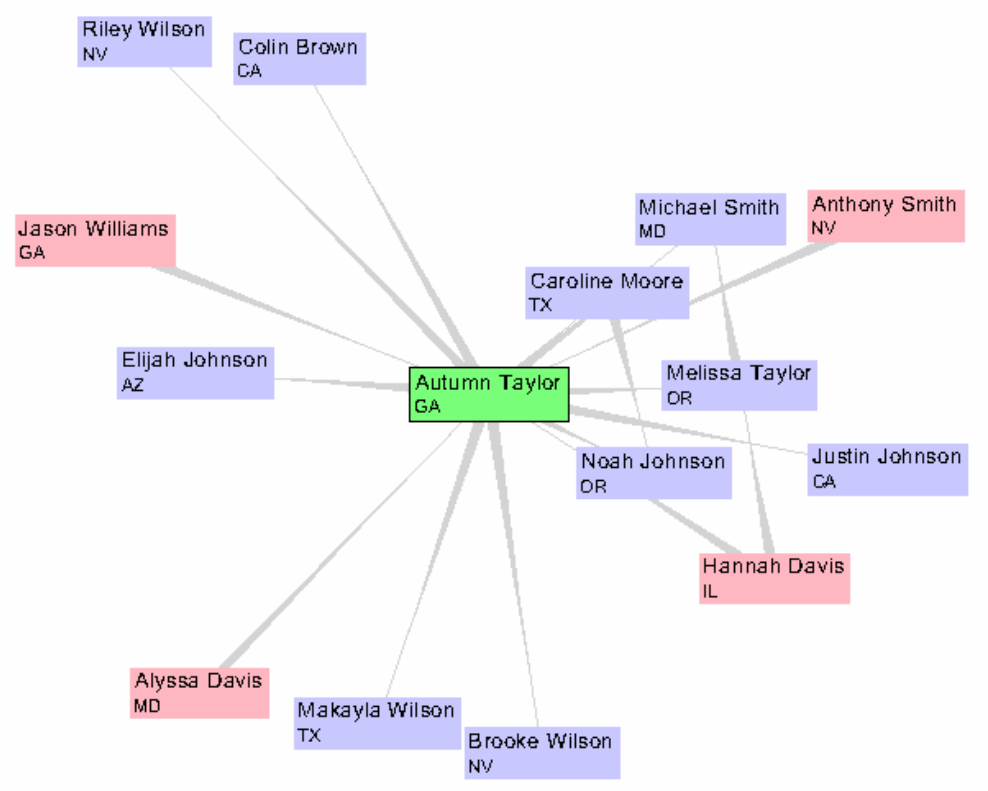


Figure 5.9 GraphPlus, showing one of the displays used in a connectivity task of the experiment: “Of all the people who emailed with “Autumn Taylor” click on the one who is email contact with the most of the others”

5.4.2 Data and Density of the Graphs

During the usability study it had been observed that users of the food web dataset would spend time reflecting whether what they saw made sense in the domain context instead of simply answering questions about connectivity. Therefore we chose to use more neutral datasets of names for the controlled experiment.

Two randomly generated graphs of 200 names of people were created. First names were unique and taken from online lists of popular baby names in Maryland in 2004. Only 10 popular last names were used, resulting in 20 occurrences for each last name. Link direction was random, and bidirectional links were allowed. Participants were told that the links represent an email communication relationship among two people. The density of graph, *d*, is defined as in Ghoniem *et al*. [60]:

$$d = \sqrt{\frac{l}{n^2}},$$

where *l* is the number of links and *n* is the number of nodes. This definition was used because its value ranges from 0 for a graph with no links to 1 for a fully connected graph. Ghoniem *et al*. used graphs with three link densities (0.2, 0.4, and 0.6) with a maximum of 100 nodes. However, graphs of 200 nodes both were used to increase the graph's complexity and because these are at the upper end of currently studied food webs, a typical graph analysis domain, and also of particular interest to us. Two link densities – low (15%) and high (30%) – were used. These numbers correspond to the range of densities found in real food webs. The number of links was 900 for 15% density and 3,600 for 30%. One drawback of this definition is that the number of links increases with the square of the number of nodes. So, 15% density might be considered very high for graphs with a large number of nodes. The 15% and 30% densities are the densities of the whole graph, and subgraphs were carefully selected to have a similar number of nodes and links.

Each node had an attribute – the US state where that person lives, randomly chosen from a set of 10 states – indicated with a two-letter abbreviation below the name.

5.4.3 Apparatus and Data Collection

We used a PC running Windows XP (3.0GHz Pentium 4 with 2GB RAM) equipped with an LC Technologies, Inc. eye tracking device and a 17" LCD monitor at 1280x1024 resolution. Results from the eye tracking study will be reported elsewhere. The size of both TreePlus and GraphPlus was 1280x863 (instructions filled the remaining lower screen). Since the width for the adjacent nodes preview panel was 150, the size of the main tree browser was 1130x863. By default, labels were displayed with a 10 pt Arial font, and attributes with an 8 pt font, and users were able to zoom in and out.

Both TreePlus and GraphPlus were instrumented using the Visualization-Interaction Architecture (VIA) software [54] developed by the CogWorks Laboratory. VIA enabled the collection of all mouse clicks, mouse movements, eye data, and systems events to a log file where they were time stamped to the nearest 16.7 ms.

5.4.4 Participants and Procedure

5.4.4.1 Participants

28 participants (20 males and 8 females) and 3 pilot testers (2 males and one female) were recruited. They were mainly CS and Engineering students who were comfortable with computers and able to quickly understand graph terminology. They already understood graph and spanning tree definitions. They received $20 for their participation. To increase motivation, a $5 bonus was given to the participant with the fastest completion time and highest success rate for each interface.

5.4.4.2 Procedure

Each participant used both interfaces; interface order was counterbalanced. Participants first received training on the first interface and the eye tracking system was calibrated for them. A custom-built testing program presented a series of tasks and allowed participants to complete the tasks using the first interface. Each task included 2 practice trials and from 3 to 5 timed trials depending on the tasks. Participants were allowed to ask questions during the practice trials but not during the timed ones. Task descriptions were always displayed in an instruction panel at the bottom of the screen. The first 2 timed trials used the low density graphs and the remaining 1-3 trials per task used the high density graph. Each trial had a 3-minute time limit and participants were allowed to give up a task at any time. Once participants completed all tasks for the first interface, they answered a subjective satisfaction questionnaire. After a short break, the same procedure was repeated with the second interface. Preferences, comments, and suggestions were collected during debriefing. Each session lasted up to two hours but was typically 1.5 hours.

The tutorial was always administered by the same person following a basic script (explanations and demonstrations) and a training dataset consisting of 100 names with 400 links (20% density). Then participants used the interface on their own and asked questions. The tutorial for the first interface – whichever it was - included some information pertinent to both interfaces (e.g. color coding, directionality, and basic interactions) so it lasted longer than that for the second interface. The training lasted about 15 minutes for the first interface, and 8 minutes for the second. In addition to demonstrating the features of the interfaces, we also

included training on basic strategies to complete the tasks for both interfaces (for example, in both interfaces the strategy to find connections between 2 people is to search for one name, use it as root, and search for the second name). If participants did not recall strategies, they were reminded during the practice trials.

5.4.5 Task Descriptions, Predictions, and Results

The six tasks and hypotheses are described below. As we predicted that each of the two interfaces would be superior for different tasks, we report independent analyses for each of the six tasks (see Figure 5.10, Figure 5.11, and Table 5.2). The complete set of tasks is provided in appendix.

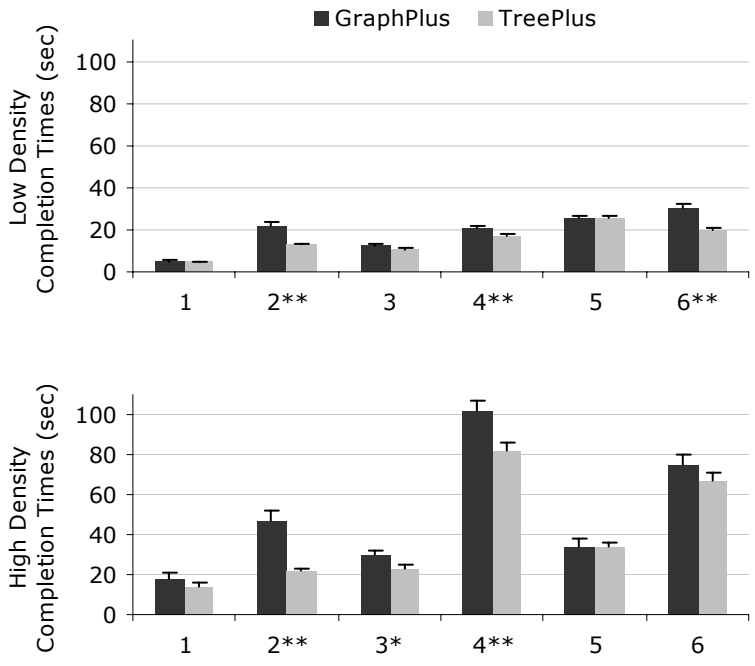


Figure 5.10 Average completion times for tasks 1 through 6 (error bars indicate Standard Error)

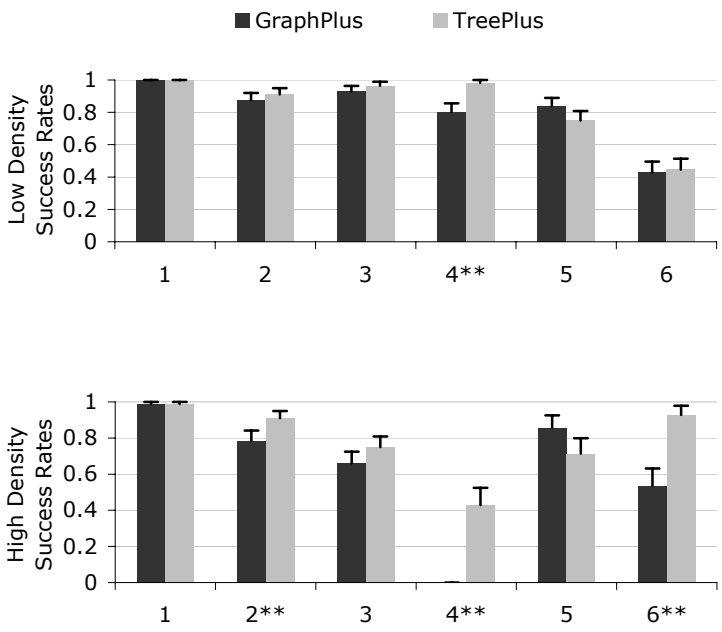


Figure 5.11 Average success rates for tasks 1 through 6 (error bars indicate Standard Error)

	Tasks					
	1	2	3	4	5	6
	Find	Browse	Adjacency	Accessibility	Common Connection	Connectivity
Completion Times						
Interface	2.69	38.95**	7.50*	14.87**	0.00	8.47**
Density	64.45**	72.94**	133.00**	411.86**	12.40**	167.89**
I*D	2.03	12.65**	2.66	9.62**	0.01	0.19
Success Rates						
Interface	0.00	2.99-	1.87	59.57**	1.83	18.04**
Density	2.08	1.99	19.02**	216.6**	0.04	6.20*
I*D	0.00	1.99	0.47	3.57-	0.19	9.08**
Error						
Interface			0.00	34.78**		
Density			11.24**	52.49**		
I*D			0.01	28.17**		
User Confidence						
Interface			7.39*	20.42**	0.04	13.93**
Density			27.45**	105.14**	0.03	3.17-
I*D			4.18-	7.65**	0.59	3.16-

** p<.01, * p<.05, - p<0.1

Table 5.2 F(1, 27): F-values for Two-Factor repeated measures ANOVA's for individual tasks (columns), with Success Rates, Completion Times, Error, and User Confidence as dependent variables (bold), and Interface and Density as the factors (rows).

Task 1. Find: Find a person that is already displayed. The person might be off screen.

Even though this task is presented as a search task our goal was to evaluate how participants scanned the entire layout. We predicted that TreePlus would perform better because the labels are aligned and sorted, and that differences would be larger when more nodes are on the screen. However, when using TreePlus users might lose time panning the display, or by forgetting that nodes are grouped in categories. For two out of five timed trials, the person to be found was off screen. Completing these trials required participants to pan or zoom out. To see whether participants benefited from the more stable layout of TreePlus, the last trial was a repeat of a previous one ("Find again").

Contradictory to our prediction, results showed no significant differences in completion times between interfaces neither at low nor at high density. They were no differences in success rates either.

For the "Find again" trials, a two-factor ANOVA was conducted to test for Trial x Interface effects on completion times in these two trials. Although the main effect of Interface was not significant, the key interaction of Trial x Interface was marginally significant, $F(1, 27)=3.22$, $p=0.084$. Planned comparisons showed that completion times on the "Find again" trial significantly improved for the TreePlus interface, going down from M=24.54 (SE=4.85) to M=8.57 (SE=1.01), $t(27)_{two\text{-}tail}=3.29$, $p<0.01$, but showed no significant improvements for GraphPlus, dropping from 22.82 sec down only to 17.46 sec, $t(27)_{two\text{-}tail}=1.06$, $p=0.30$.

Task 2. Browse: Follow a path.

To test the combination of reading and browsing we asked users to follow a path of length 3. We predicted that, regardless of the density of the graphs, TreePlus would work better because the nodes are easier to locate and read. The last of the four tasks asked users to go back to the first node on the path. Differences between interfaces should be greatest for this last trial ("Browse and revisit") as, in TreePlus, it is easy to backtrack via parents in the tree and the nodes will have moved only slightly.

The results showed a main effect of interface and an interaction effect of Interface x Density, confirming our hypothesis that TreePlus performed better than GraphPlus and that the benefits of TreePlus increase with density. The speed advantage was not compromised by more errors as there was still a marginal significant advantage for TreePlus in success rate.

Similarly, our hypothesis that TreePlus should work better on the "Browse and revisit" trials than GraphPlus was supported as well. A two-factor repeated measures ANOVA on the effects of Revisitation (Browse vs. "Browse and revisit" trials) and Interface in the Browse task revealed a significant main effect of Interface, $F(1,55)=37.61$, $p<0.01$, a significant main effect of Revisitation, $F(1,55)=26.42$, $p<0.01$, and a significant Interface x Revisitation interaction, $F(1,55)=14.11$, $p<0.01$. Planned comparisons showed that the interaction reflected the fact that completion times for the TreePlus interface were not significantly different between Browse ($M=16.39$, $SE=1.17$) and Browse+Revisit ($M=18.36$, $SE=1.10$), $t(55)_{\text{two-tail}}=1.45$, $p=.15$, whereas completion times for GraphPlus were significantly longer for the

Browse+Revisit (M=43.16, SE=4.47) trials than the Browse (M=25.98, SE=2.18) trials, $t(55)_{two\text{-}tail}=4.71$, $p<0.01$.

Task 3. Adjacency: Among all those who communicate with a specific person, count those with a given characteristic.

We asked participants to count to simulate a task where all node labels have to be read. Participants had to expand the graph, scan the node labels, and be aware of the direction of the links. We marked the specific starting person with a red circle so that participants would not spend time finding it. We predicted that there would be no difference between interfaces at low density, but that at high density, GraphPlus would suffer from severe occlusion problems.

The predicted interaction of Interface x Density was not significant; however, the data showed a slight advantage for TreePlus as density increased. We might expect the advantage to be greater for higher densities than used in this study. There were no differences for success rate and error.

Task 4. Accessibility: Count people with a given characteristic within two links (distance 2) of a given person.

Users had to use a menu item to expand the tree two levels down and then read and count nodes. We again marked the starting node with a red circle. We expected that the results would be similar to the adjacency task: no difference between interfaces for the low-density graph, with GraphPlus suffering from occlusion at high-density. However, with TreePlus, users may also spend a lot of time panning the tree, or forget to pan the tree.

The results showed a significant effect of interface and an interaction effect of Interface x Density, confirming our hypothesis that TreePlus performed better than GraphPlus (Figure 5.10 and Table 5.2). The benefits of TreePlus increase dramatically with density. We also found a strong significant difference in favor of TreePlus for success rate and error.

Task 5. Common Connection: Find all people who have been in direct email communication with two given people.

Nodes for the two given people were not on screen so participants needed to use the strategy they had learned (search one person, re-root, and search the other person) with both interfaces. Regardless of link density, we predicted that there would not be any difference in time between interfaces because the search strategies are identical.

As predicted, no significant differences were found in completion time or success rate.

Task 6. Connectivity: Find who has the most email relationships with other people in a group.

Here we expected GraphPlus to do better than TreePlus because all links are drawn on the display while TreePlus requires interaction to reveal cross links. However, for the high density graph, performance with GraphPlus may suffer due to occlusion. The "group" to explore was defined as all the people who exchanged email with a specific person and we again marked the node of that specific person with a red circle.

To our surprise, there was a significant main effect of completion times as well as success rates that favored TreePlus (Figure 5.10, Figure 5.11, and Table 5.2). The advantage of TreePlus increased with density.

For error, significant main effects of Interface and Density, as well as an Interface x Density interaction were found, with error being smaller for the TreePlus interface, and especially small on the low-density trials.

5.4.5.1 Confidence and Preference

Except for the Find and Browse tasks, user confidence was recorded. The overall ANOVA showed significantly higher self-reports of confidence for TreePlus (M=8.01, SE=0.19) than for GraphPlus (M=7.45, SE=0.24), and for low-density (M=8.15, SE=0.18) than for the high-density trials (M=7.30, SE=0.23). Individual task comparisons revealed significant advantages for the TreePlus interface for Adjacency, Accessibility, and Connectivity tasks. Significant effects of Density were found for Adjacency and Accessibility tasks, with a significant interaction of Density x Interface for Accessibility tasks.

When asked which interface they preferred, 26 out of 28 participants chose TreePlus over GraphPlus. This general question was followed by a 10-item satisfaction questionnaire with ratings on a 9 point Likert scale. Individual two-tail t-tests were performed for each of the ten questions. TreePlus received significantly better ratings for Overall use, Navigation, Layout of information, Reading many labels was easy/clear, Arrows representing direction were helpful/clear, and Use of color was helpful/clear. No significant differences were found between TreePlus and

GraphPlus for Predictable system response, Ease of learning, Use of highlighting was helpful/clear, and Use of arrows was helpful/clear.

5.4.6 Observations and Discussion

There were wide differences between participants in terms of speed and accuracy. However, despite individual variability, TreePlus performed significantly better for most of the tasks. TreePlus even outperformed GraphPlus for the Connectivity task where we had hypothesized GraphPlus would perform better. The benefits of TreePlus increased with density. We first discuss in more detail the surprising results of the Connectivity task then discuss specific features of the interfaces.

5.4.6.1 Task 6: Connectivity

We were surprised to observe that in Connectivity task many participants did not use the topology information in GraphPlus. Though all cross links are drawn and users should have been able to spot nodes with the most links – especially in the low density graphs – observations lead us to suspect that participants gave up or lost confidence using the link arrows after they used messy and poorly readable graphs in previous tasks. Instead they mostly used highlighting just as TreePlus required them to do. TreePlus even had a higher success rate, possibly because this highlighting exploration could be performed in a more orderly way. After observing this effect with many participants we considered reminding users that they could better use the links information of GraphPlus but decided that we could not change the training procedure in the middle of the experiment.

5.4.6.2 Occlusion vs. Panning

Occlusion seemed the most important problem for GraphPlus. During the most complex trial (the Accessibility trial for the high density graph), many participants gave unhappy exclamations such as "Oh boy" and "Wow." Furthermore, while 12 participants answered correctly with TreePlus, with GraphPlus no one answered correctly, one participant could not finish within the 3 minute time limit, and one participant gave up. For TreePlus, panning was an issue. Some people seemed to take a few moments to remember to pan. Some tried to use the mouse wheel which was not supported at that time. Three participants made explicit negative comments about panning, and many people sighed when they had to keep panning for the very tall tree for the most complex trial. Nevertheless participants were much more successful completing the task correctly with TreePlus than with GraphPlus.

To enable users to see all children of a node at once without panning, we used multi-column layout when the height of the children of the currently opened node was bigger than the window height (Figure 5.13a). However, as one participant noted "Multi-column layout was useful but confusing." Although it was not needed and did not help, many participants panned anyway in the apparent belief that there were more nodes above or below.

5.4.6.3 Search

With both interfaces, the search results showed only one shortest path between two nodes (Figure 5.6). However, anecdotal evidence suggests that some participants

sometimes thought that the person currently shown on the path was the only one who communicated with both people.

5.4.6.4 Preview panel in TreePlus

In contrast to our expectations, most participants did not seem to use the adjacent nodes preview for the Adjacency task. Instead, people just opened nodes. This might be because opening a node is simple and cheap, and users are not accustomed to the adjacent nodes preview. Among seven participants observed actively using the adjacent nodes preview, two commented that they really liked it. We plan to make the preview panel optional.

Note that, for both interfaces, users could not select other nodes during the animation. However, adjacent nodes for the selected node are positioned at their new position later in the animation sequence with TreePlus than with GraphPlus. We believe that GraphPlus gained 0.6 seconds (1/3 of the total animation time) for the Browsing task because participants were able to start visual scanning earlier. In spite of this disadvantage, TreePlus still worked better than GraphPlus.

5.5 TreePlus Improvements

TreePlus design has been evolved to fix usability problems and to support other types of datasets, which require more features.

5.5.1 Design Improvements after the Usability Study

To fix several issues observed through the usability study, we made many changes to TreePlus from our earlier version [89]. The major changes include:

1) Mouse click instead of mouse hover now changes focus
2) Arrows are now placed in a better location
3) Search can now be limited to one selected direction, so as to result in a valid shortest path for that direction
4) The preview panel was reorganized
5) "+N moved" was added to indicate when a list of children is incomplete because some of them have moved
6) Legends were provided
7) When showing the partial overview, do not zoom out to fit it in window
8) Lines are now curved rather than straight to minimize occlusion by labels

5.5.2 Design Improvements after the Controlled Experiment

To facilitate the Connectivity task an optional connectivity hint was added. A black vertical bar on the left side of the node shows the percentage of the connected nodes among on-screen nodes. For example, among all organisms eaten by "broad-winged hawk," "Puerto Rican coqui frog 1" has the most connections (Figure 5.12). We also added an option to draw the cross links as curved lines to show the overall connectivity of the partial graph on screen.

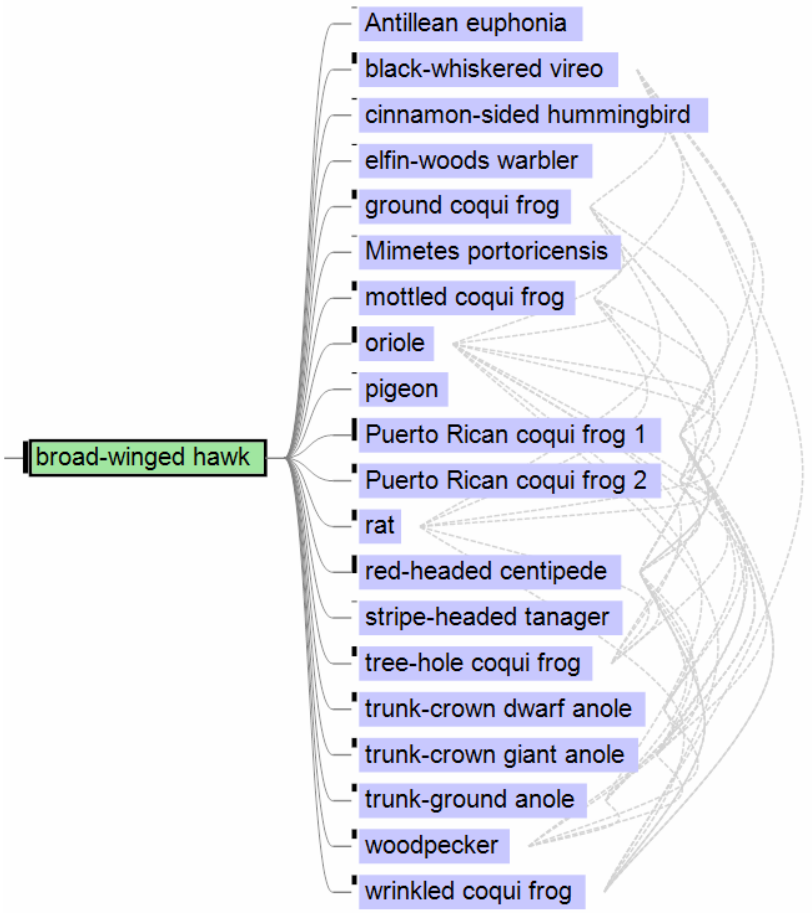


Figure 5.12 Connectivity bar indicated by a black vertical bar on the left side of the node shows the percentage of connected nodes for each node. Cross links are shown on demand with the dotted lines.

To address some of the problem of multicolumn layouts (Figure 5.13a) we now balance the heights of columns, and guarantee a gap between the edge of the window and the multi-column background (Figure 5.13b), making it more obvious that there is no need to pan.

(a)

(b)

Figure 5.13 (a) previous multi-column layout (b) improved multi-column layout

For search, TreePlus now shows all shortest paths between two nodes. Users can confirm the actual shortest paths by turning on the option that draws the cross links (Figure 5.14). Users can also search several keywords (e.g., several peoples' names) separated by semicolon.

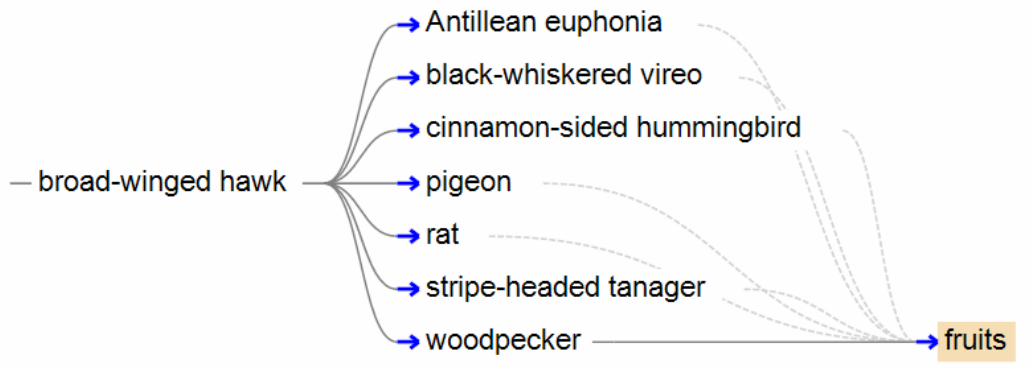


Figure 5.14 A search for "fruits" from "broad-winged hawk" with cross links visible shows all seven shortest paths from "broad-winged hawk" to "fruits."

5.5.3 Design Extension

5.5.3.1 Supporting Other Datasets

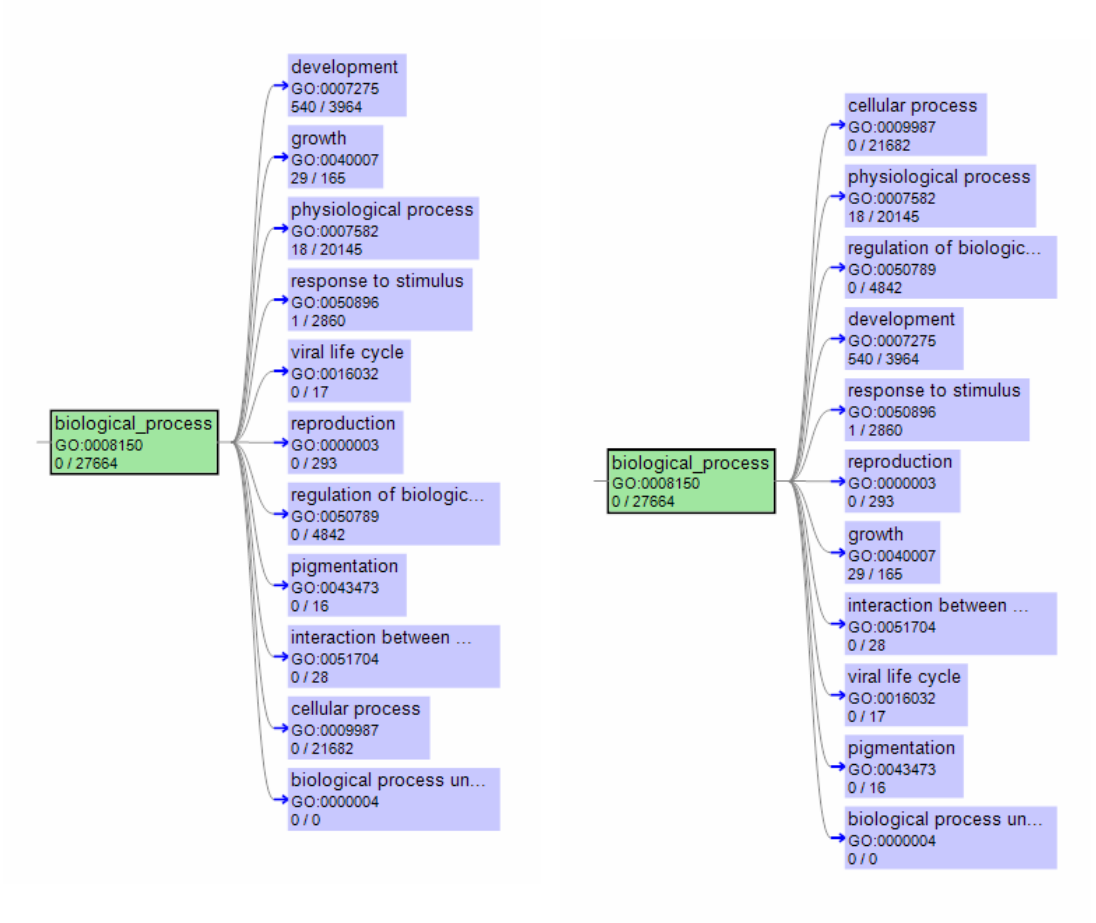
Our techniques were applied to two other real datasets. First, we visualized the co-authorship graph and citation network for the ACM CHI proceedings [87]. These datasets introduced additional requirements. Since each author has four attributes, the screen can be cluttered if all attributes are shown. TreePlus now enables users to specify which attributes to be shown. In addition, TreePlus allows users to show the categorical attribute with colored dots. TreePlus assigns colors based on the frequency of each category value over the whole graph. For the CHI authors, "Unknown" was most frequent since institution information for many authors is

missing (Figure 5.15). "University" was most frequent among the known institutions. Since TreePlus uses only 5 colors, if the number of categories is larger than 5, TreePlus assigns the first four colors to the four most frequent category values respectively and one last color to all other category values. Users can either show or hide the category attribute as a string and the actual value for each color is provided in the legend. In Figure 5.15, you can easily see that "University" is the institution for most coauthors of "Benjamin Bederson" including himself, as seen by its representative orange color.

Figure 5.15 The type of institutions for authors is represented by colored dots.

Since paper titles are usually very long, TreePlus enables users to set the maximum width for nodes. When the label is clipped, an ellipsis at the end indicates that there is more text and the whole label is revealed when users move the cursor over the node.

Second, we visualized annotated GO terms in the gene ontology structure using GOTreePlus, a prototype gene ontology browser. The important information in this dataset is how many times each GO term and its descendents are annotated, which are numeric values. To save space (reduce height), TreePlus allows users to show these numbers in one row separated by a "/" character as shown in Figure 5.16. Furthermore, TreePlus also enables users to specify which attributes are numeric, since numeric sort is different from string sort.



(a) (b)

Figure 5.16 TreePlus enables users to show two attributes in one row and sort children by numeric values. (a) Children are sorted by the number of its own annotations. (b) Children are sorted by the sum of its descendents' annotations.

5.5.3.2 Duplicating Nodes on Demand to Represent Cross Links

For directed acyclic graphs (DAGs), some researchers generated a tree by duplicating the cross-linked nodes then visualized it with a tree visualization system. Since this approach could work well for graphs that have a modest number of cross links, TreePlus now provides an option of duplicating nodes instead of moving to represent cross links. For example, six frogs that are connected to both "broad-winged hawk" and "rat" are duplicated in Figure 5.17. When users click on a node, TreePlus

highlights its duplicates if any to help users find them Figure 5.18. In contrast to earlier works, TreePlus will be applicable to any graphs even if they have cycles because it duplicates nodes on demand.

Figure 5.17 TreePlus can duplicate nodes to represent cross links. When users open "rat," six frogs that are connected to both "broad-winged hawk" and "rat" are duplicated.

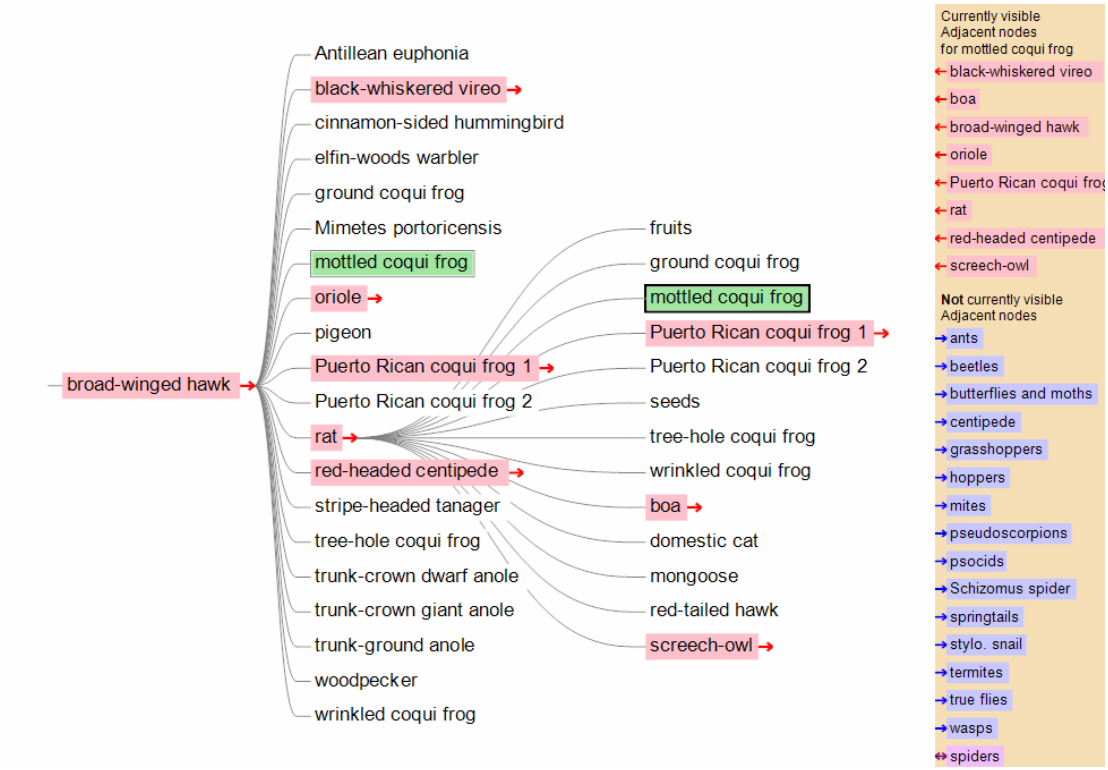


Figure 5.18 When users click on "mottled coqui frog" which was connected to both "broad-winged hawk" and "rat," TreePlus highlights its duplicates.

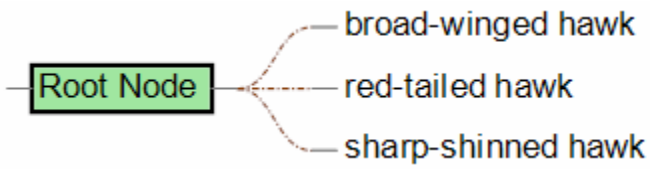
5.5.3.3 Supporting Multiple Roots

While trees, by definition, can have only one root, graphs may consist of several connected components. If a graph has multiple connected components, TreePlus automatically identifies them and a root of the spanning tree for the each connected component. When visualizing the graph, TreePlus adds an arbitrary root named "Root Node" and links connecting the dummy root to the each root node (Figure 5.19b). Furthermore, even if the graph is only one connected component, users might be interested in multiple nodes and want to explore from those nodes. To address this

problem, TreePlus provides a way to specify multiple root nodes in the preferences (Figure 5.19a).

```
<DefaultRoot>Use Dummy</DefaultRoot>
<CurrentRoot>
      <Root id="116236">broad-winged hawk</Root>
      <Root id="116234">red-tailed hawk</Root>
      <Root id="116193">sharp-shinned hawk</Root>
</CurrentRoot>
```

(a)



(b)

Figure 5.19 (a) How to specify multiple root nodes in the preferences. (b) To handle multiple roots, TreePlus adds an arbitrary root named "Root Node" and links connecting the dummy root to each root node.

5.5.3.4 Supporting Multiple Link Types

While there is only one link type in our food web data, graphs often have multiple link types. For example, SPIRE's Food Web Constructor [126] tries to construct a food web by utilizing data contributed by a number of researchers. To test the performance of the system, it reconstructs one of the food webs stored in the database. The results of the reconstructions have four types of links; True Positive, False Positive, True Negative, and False Negative. While TreePlus cannot visualize negative link types because they mean there is no link, visualizing two positive link types differently give an idea of how good the result is and help researchers spot where the main problem is. To help users easily recognize different link types,

TreePlus uses different colors and styles (e.g. solid/dotted) for each link type (Figure 5.20).

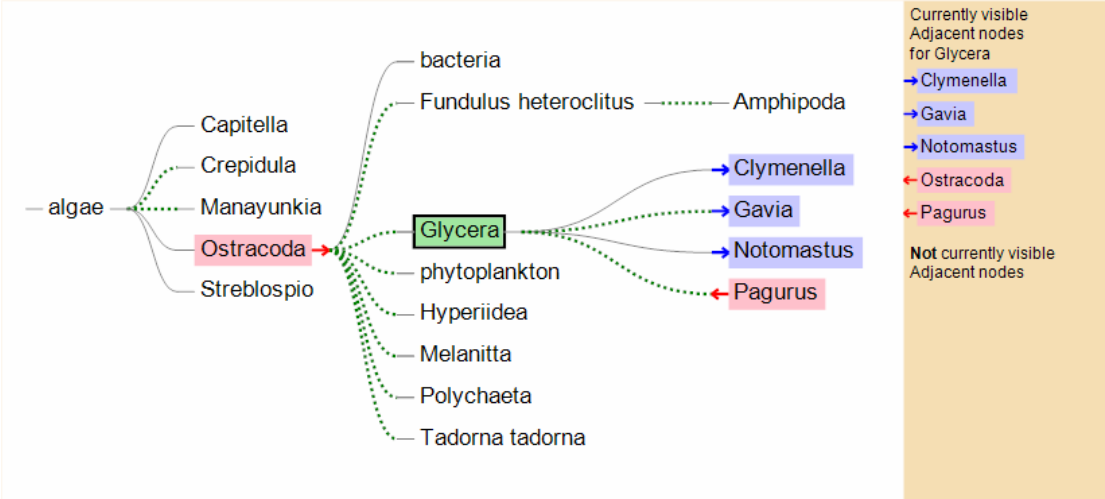


Figure 5.20 Gray solid lines represent true positive and green dotted lines mean false positive.

5.6 Implementation Details

TreePlus is designed as a reusable component to support the rapid creation of graph visualization systems. Implemented in C# with Piccolo.NET, TreePlus is pluggable into any .NET application. It provides an application programming interface (API) and fires events to communicate with the containing application. TreePlus consists of 15 classes and about 7,000 lines.

Data structures and basic operations related to graphs are contained in a separate library called GraphLibrary so they can be reused for other controls, such as GraphPlus. GraphLibrary also contains some utility classes needed for specifying preferences and non-graph related operations.

5.6.1 TreePlus Architecture

Piccolo.NET provides a control called PCanvas, to host a piccolo scene-graph in .NET Windows applications. If we extend this class directly, application developers would need to have knowledge about Piccolo components. They would also need to add references to the Piccolo components in their application. Furthermore, they have unnecessary access to the properties of a PCanvas such as "Root," "Camera," and "Layer." To avoid forcing application developers to have unnecessary knowledge about Piccolo, the TreePlus control is inherited from the UserControl class in .NET rather than PCanvas, and contains the PCanvas as a member variable.

The TreePlus interface consists of the main tree browser on the left and a preview of the adjacent nodes on the right. When users pan the tree, the nodes in the preview should not move, and vice versa. In other words, each of these two views should have separate event handlers registered with two different cameras. The adjacent nodes preview (PPreviewNode) was added to the default layer (PLayer) viewed by the default camera (PCamera) in the PCanvas provided by Piccolo. We created another layer called TreePlusLayer inherited from PLayer to show the tree structure, and added it to PRoot as a child. An internal camera was created and added TreePlusLayer to its view. Figure 5.21 shows this runtime structure of the TreePlus control.

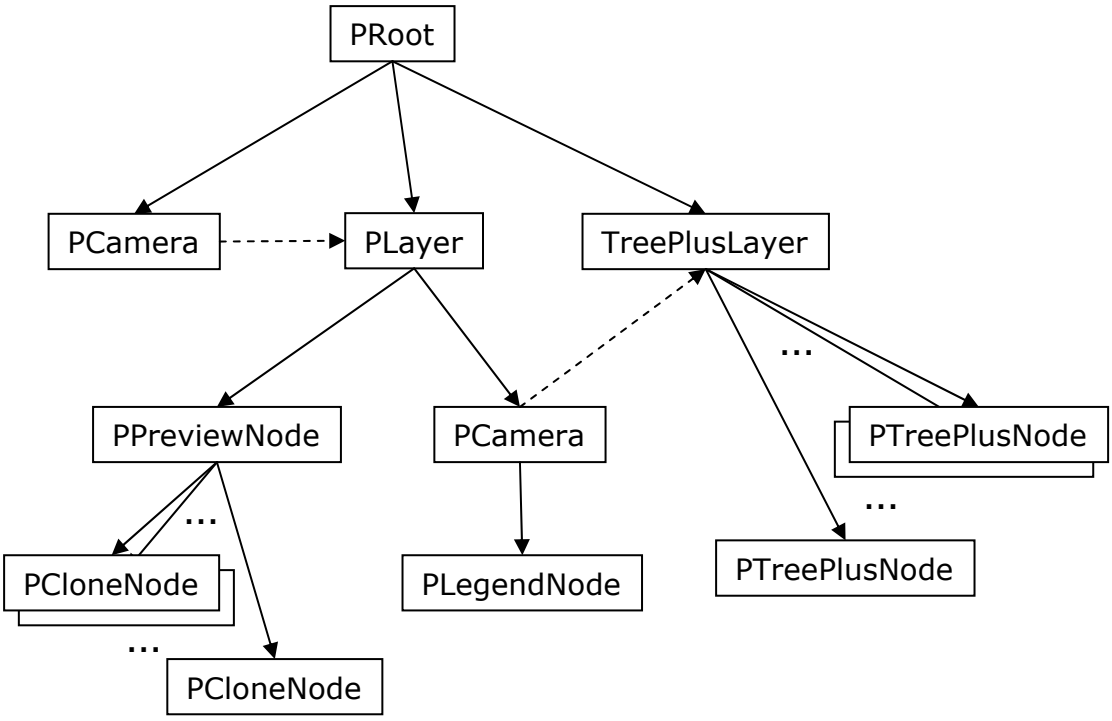


Figure 5.21 TreePlus Control Runtime Structure

Piccolo provides a default event handler for zooming. To zoom in and out, users have to drag the mouse while pressing the right button. While we were developing TaxonTree, we learned that most users have difficulty with the default zoom interaction. Therefore, we disabled Piccolo's default zoom event handler and created a new one, TreePlusEventHandler, which is based on keyboard navigation. It was added as an input event listener to the internal camera looking at TreePlusLayer.

5.6.2 Data File Format

The current GraphLibrary handles data files written in GraphML [62], a markup language to describe the structural properties of a graph. Two additional assumptions are made by the GraphLibrary; 1) An attribute called "nodename" must be provided

to be used as node labels. 2) All node elements should be defined before they are used in the edge elements. The appearance order of the node attributes in TreePlus depends on the order of the attribute definitions in the data file. Figure 5.23 is a GraphML file for the sample undirected graph shown in Figure 5.22.

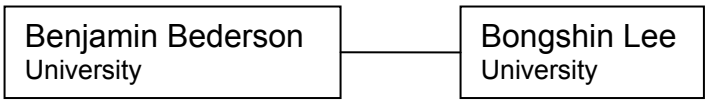


Figure 5.22 A sample undirected graph.

```
<graphml>
      <!-- node attributes are defined using key elements-->
      <key id="nodename" for="node"
            attr.name="nodename" attr.type="string"/>
      <key id="affiliation" for="node"
            attr.name="affiliation" atty.type="string"/>

      <graph id="G" edgedefault="undirected">
            <node id="P28811">
                  <data key="nodename">Benjamin Bederson</data>
                  <data key="affiliation">University</data>
            </node>
            <node id="P438767">
                  <data key="nodename">Bongshin Lee</data>
                  <data key="affiliation">University</data>
            </node>

            <edge source="P28811" target="P438767"/>
      </graph>
</graphml>
```

Figure 5.23 GraphML for the sample graph shown in Figure 5.22 to be handled by the GraphLibrary.

5.6.3 Data Structure

The data graph is read into an instance of the Graph class, which provides functions for loading and saving a graph in the data file format described in the previous

section. This class also contains functions for managing a graph such as adding nodes and edges, and other graph operations such as finding a shortest path.

While the whole graph should be loaded into memory, the tree structure for the graph is incrementally constructed as users browse the graph in TreePlus. Each node in the graph is represented by an instance of the Node class. Each node in the tree is represented by a TreePlusNode object that has a reference to the mapped Node instance. The TreePlusNode class also contains member variables for the tree structure such as parent and an array of TreePlusNode children. If we duplicate all data in each TreePlusNode instance when duplicating nodes to represent cross links, it may waste memory especially for highly connected graphs with many attributes. Instead, we use the original TreePlusNode once and duplicate view nodes represented by PTreePlusNode instances.

The TreePlusNode class contains an array of PTreePlusNode instances, one for each view node. Additional fields include the number of adjacent nodes and whether there is a self cycle. The PTreePlusNode object contains the TreePlusNode instance as a data node. It also contains the tree structure information for layout such as left sibling, left neighbor, parent, and so on.

5.6.4 Preferences

Graphs vary from a simple undirected one to more complex one with many node attributes. When designing an interactive graph visualization system for general purposes, we have many options to control. For example, there are layout options such as gap between siblings and distance between levels. Best configuration sometimes depends on the dataset. As described before, TreePlus allows users to set

the maximum with of nodes. It also provides several ways to specify a root and two ways to represent cross links.

For animation, it is useful to enable users to set the duration for the animation. In fact, the best drawing of a graph often depends on the personal preference. For example, some people want to have a border for a node but others don't. While some people prefer to use background colors of nodes to represent link directions, others want to use the colored arrows, or use both.

Furthermore, the meaning of nodes and links are usually different since it is a property of the graph. For example, while the link in the food web data means "eats" or "eaten by," the link in the graph we used for the user study mean "has an email communication." If we want to provide meaningful legend for the links, it should be configurable. Users may want to see the legend when they first look at the graph but hide it once they are familiar with it. This information for each graph can be specified in the Preferences object, passed to TreePlus, and saved in a file. The full description of the Preference class is provided in the appendix.

5.6.5 How to Use TreePlus

Since TreePlus is a UserControl in C#, it can be plugged into any .NET applications just like other controls. The example shown in Figure 5.24 creates a TreePlus control, sets its bounds, sets up event handlers for drag and drop, and adds it to a Windows Form.

```
TreePlusControl treePlusControl;

private void InitializeControl() {
      // create a TreePlus control
      treePlusControl = new TreePlusControl.TreePlusControl();

      // set bounds
      Rectangle desiredBounds = new Rectanble(0, 0, 1024, 768);
      treePlusControl.Bounds = desiredBounds;

      // setup event handlers
      treePlusControl.AllowDrop = true;
      treePlusControl.DragEnter +=
            new DragEventHandler(treePlusControl_DragEnter);
      treePlusControl.DragDrop +=
            new DragEventHandler(treePlusControl_DragDrop);
      treePlusControl.DragLeave +=
            new EventHandler(treePlusControl_DragLeave);

      // add the control to a form
      Controls.Add(treePlusControl);
}
```

Figure 5.24 Example code to plug TreePlus into a Windows Form.

You can show a graph from a data file using the TreePlus control as shown in Figure 5.25. To load a graph from a file you can use the function LoadGraph provided by the GraphLibrary. You should pass that graph to TreePlus by setting the TreePlusControl.DataGraph property before you call the ShowGraph function. Its parameter is an id for the root node. If you pass an empty string, TreePlus uses DefaultRoot/CurrentRoot information in the preferences. You can setup your own context menu by setting the TreePlusControl.ContextMenu property. You may also want to set the TreePlusControl.Preferences property after setting desired values, such as maximum width of nodes.

```
GraphLibrary.Graph graph;
String rootNodeId = "";
Preferences pref;
ContextMenu contextMenu;

public void ShowGraph(string filename) {
      // load a graph
      graph = new GraphLibrary.Graph();
      graph.LoadGraph(filename);

      // set the ContextMenu property
      treePlusControl.ContextMenu = contextMenu;

      // set the DataGraph property
      treePlusControl.DataGraph = graph;

      // set maximum width of nodes using preferences
      pref = new Preferences();
      pref.MaxWidth = 150;
      treePlusControl.Preferences = pref;

      // show the graph from the rootNodeId
      treePlusControl.ShowGraph(rootNodeId);
}
```

Figure 5.25 Example code to show a graph read from a data file.

The data graph can also be constructed on the fly using GraphLibrary. Figure 5.26 shows how to build the sample undirected graph shown in Figure 5.22 at run time. Although node information is hard-coded here for simplicity, it can be retrieved from other source such as a database in a real situation.

```
public Graph BuildGraph() {
      // create a graph
      Graph g = new Graph();

      // specify whether it is directed or not
      g.Directed = false;

      // specify order of the attributes
      g.AttrOrder.Add("nodename");
      g.AttrOrder.Add("affiliation");

      // create nodes
      Node n1 = new Node();
      n1.Id = "P28811";
      n1.Data["nodename"] = "Benjamin Bederson";
      n1.Data["affiliation"] = "University";

      Node n2 = new Node();
      n2.Id = "P438767";
      n2.Data["nodename"] = "Bongshin Lee";
      n2.Data["affiliation"] = "University";

      // add nodes and the edge to the graph
      g.AddNode(n1);
      g.AddNode(n2);
      g.AddEdge(n1, n2, 1);

      return g;
}
```

Figure 5.26 Example code to create the sample graph shown in Figure 5.22 at runtime.

Since Piccolo's default zoom event handler is disabled, TreePlus provides an API for zooming; Zoom and FitinWindow. You should call these functions after showing a graph using the TreePlus control. The Zoom function allows you to set the zoom factor and the FitinWindow function zooms in/out to fit the tree in the window. The full list of API is provided in the appendix.

5.6.6 Graph Operations

Since we do not have the complete tree structure for the graph, necessary graph operations are applied to the Graph instance in GraphLibrary. To count how many nodes can be reached in each level from a node, we computed all pairs shortest path by using the Floyd-Warshall algorithm [32, Chapter 26]. To get the number of nodes of distance *d* from a node, we counted the nodes whose shortest path length from the node is *d*. Since the time complexity of the algorithm is $O(|N|^3)$, where |N| is the number of nodes, it takes too long when |N| is large. Therefore, we currently disable this feature if |N| is larger than 1,000. To overcome this problem, we should pre-compute these numbers.

For search, as described before, TreePlus shows the shortest paths from the root to the search result nodes. We first apply string-match search over the Graph instance. Next, we collect all the nodes included in the shortest paths from the root to the search results. We then build a tree using these nodes and links between them, and visualize the tree in TreePlus.

5.7 Discussion

A new interactive graph visualization component was developed by applying a tree layout approach to graph visualization. It was based on a guiding metaphor: "Plant a seed and watch it grow." The study results suggest that visualization and interaction techniques can effectively support incremental exploration of a graph, and reveal the graph structure superimposed on a tree structure.

The user study compared TreePlus with a standard graph visualization system (GraphPlus) and found that participants completed the tasks faster and with fewer errors with TreePlus for several tasks. When started designing TreePlus, we suspected it might work well for some tree-like graphs, such as web hierarchies and gene ontologies, but not for high-density graphs that have many cross links. However, once we used TreePlus and GraphPlus with a real food web dataset we found that GraphPlus works well for low-density graphs but suffers more as the density increases. In accordance with our own experience, the study result shows that the benefits of TreePlus increased with density. Participants also reported higher levels of confidence in their answers with TreePlus and most of them preferred TreePlus. For the medium sized graphs used in the study, the benefits of TreePlus increased with density. Although the high density we used is representative of real food webs, it may not be as common in other domains. For large graphs, our low density (15%) might also be considered high and this might have contributed to the good performance of TreePlus overall in this experiment. Further evaluations using lower densities with larger graphs would shed more light on the benefits of TreePlus.

Chapter 6

Task Taxonomy for Graph Visualization

Despite a long history of graph visualization research, only a few graph visualization systems have actually been tested with real users. Furthermore, the tasks that were used were highly domain-specific. To improve the evaluation of information visualization techniques and systems, it is important to have benchmark datasets and tasks [107]. This chapter suggests a list of tasks commonly encountered while analyzing graph data. I worked with Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry.

We first prepared lists of tasks with examples taken from several domains such as food webs, bibliography, and student class assignments. The task taxonomy for tree visualization posted in the InfoVis 2003 contest [74] was used as a starting point. We then reviewed several user studies of graph visualization techniques and extracted the tasks used in those studies.

After making those two lists, we considered the set of low-level Visual Analytics tasks proposed by Amar *et al*. [4]. These tasks were extracted from a corpus of questions about tabular data. We realized that our tasks all seem to be compound tasks made up of Amar *et al*'s primitive tasks applied to the graph objects. When some tasks could not be represented with those tasks and objects, either an object or a low-level task was added. This chapter demonstrates how all complex tasks could be seen as a series of low-level tasks performed on those objects.

6.1 Graph-Specific Objects

A graph consists of two types of primitive elements, nodes and links. A subgraph of a graph G is a graph whose nodes and links are subsets of those of G. There are several meaningful subgraphs such as connected components.

- Nodes

  Nodes by nature have an attribute degree that is the number of links incident to that node. In a directed graph, there are two types of degrees according to the direction; indegree and outdegree. For practical use, nodes also have a special "label" attribute. They often have application-dependent attributes as well. In network analysis, there are various measures used to determine the centrality, or relative importance, of a node within the graph (for example, the importance of a person within a social network). Measures of centrality include betweenness and closeness. There is also a special kind of node called an articulation point, whose removal disconnects a graph.

- Links

  Links can have labels and application specific attributes. For a directed graph, each link also has a "direction" attribute. A bridge is a link whose removal disconnects a graph.

- Paths

  A path is an alternating sequence of nodes and links, often represented as a sequence of just nodes, since there is only one link between two nodes in most cases. If the first and last nodes of the path are the same, it is called a cycle. Shortest path between two nodes is a path in which the sum of the weights of

the constituent links is minimized. If the links are not weighted, instead the number of links in the path was minimized.

- Graphs

  Graphs can be considered to be objects as well as users might want to compare graphs or see how a graph changes over time. Graphs have a "directed" attribute defined by whether or not links in the graph are directed and a "cyclic" attribute defined by whether or not the graph contains any cycles. Graphs can also have some computed attributes such as the number of nodes and links.

- Groups

  A group can be defined as a set of related nodes, such as nodes with common attribute values or nodes of interest to users.

- Connected Components

  A connected component is a maximal connected subgraph.

- Clusters

  A cluster is a set of objects that are spatially close together. For graphs, this is a subgraph of connected components whose nodes have high connectivity. Thus, in our terminology, clusters are based solely on link information, in contrast to a group.

6.2 Low-Level Tasks

Amar's low-level tasks (shown in Table 6.1) are all relevant to graphs. In the task descriptions, a "data case" is an entity in the dataset and an "aggregation function" is

a function that creates a numeric representation for a set of data cases, such as average and sum.

Tasks	Descriptions
Retrieve value	Given a set of cases, find attributes of those cases.
Filter	Given some conditions on attributes values, find data cases satisfying those conditions.
Compute Derived Value	Given a set of data cases, compute an aggregate numeric representation of those data cases. (e.g. average, median, and count)
Find Extremum	Find data cases possessing an extreme value of an attribute over its range within the dataset.
Sort	Given a set of data cases, rank them according to some ordinal metric.
Determine Range	Given a set of data cases and an attribute of interest, find the span of values within the set.
Characterize Distribution	Given a set of data cases and a quantitative attribute of interest, characterize the distribution of that attribute's values over the set.
Find Anomalies	Identify any anomalies within a given set of data cases with respect to a given relationship or expectation, e.g. statistical outliers.
Cluster	Given a set of data cases, find clusters of similar attribute values.
Correlate	Given a set of data cases and two attributes, determine useful relationships between the values of those attributes.

Table 6.1 Ten Visual Analytics tasks proposed by Amal *et al*.

The last three tasks do not have ground truth answers that can be easily compared with users' answers. The "Correlate" task may have a statistical ground truth but we assume that in the field of Information visualization, the intended meaning of "Correlate" is "identify possible correlations."

We propose one graph-specific task and two general tasks that are not covered by the above list.

- Find Adjacent Nodes: Given a node, find its adjacent nodes.
- Scan: Quickly review the list of items.

  This task differs from the "Retrieve Value" task, since it usually requires users to review many items at once but not necessarily to retrieve exact values. For example, if users want to find "Robin Williams" they can immediately move to the next item if it does not start with "R." They can also stop when they find an answer. Depending on the tasks, users may need to continue to review all items. The values may not be specific, for example users may need to scan for foreign names. They need not be text, for example users may need to scan for color-coded information.
- Set Operation: Given multiple sets of nodes, perform set operations on them. For example, find the intersection of the set of nodes.

6.3 Graph Task Taxonomy

This section summarizes a list of tasks commonly encountered while analyzing graph data. These suggested tasks are further categorized into four groups: topology based tasks, attribute based tasks, browsing tasks, and the overview task. Each task has

general descriptions and example scenarios. FOAF, FW, GO, and ARM represent friend-of-a-friend graph, food webs, gene ontology, and airport routing map respectively. In addition, this section shows how each task can be decomposed into low-level tasks, shown in italics, on specified graph objects. While there might be several ways to decompose a task, only one way is presented.

Note that finding a node is a common starting point for many tasks. But this task may not be performed by users when a search feature is provided by the system. While it is described as a component for each task, the task might be excluded from a user study.

6.3.1 Topology-Based Tasks

6.3.1.1 Adjacency (direct connection)

General Descriptions:

- Find the set of nodes adjacent to a node.
- How many nodes are adjacent to a node?
- Which node has a maximum number of adjacent nodes?

Examples:

- (FOAF) Find the names of the direct friends of Eric.

  [*Find* on Nodes + *Find Adjacent Nodes* on Nodes + *Retrieve Value* on Nodes]
- (FW) How many kinds of organisms do golden eagles eat?

  [*Find* on Nodes + *Find Adjacent Nodes* on Nodes + *Filter* on Links + *Count* on Nodes]
- (FOAF) Who is the most popular person?

[*Find Extremum* on Nodes]

6.3.1.2 Accessibility (direct or indirect connection)

Accessibility task can be treated as a repetition of the Adjacency task.

General Descriptions:

- Find the set of nodes accessible from a node.
- How many nodes are accessible from a node?
- Find the set of nodes accessible from a node where the distance is less than or equal to *n*.
- How many nodes are accessible from a node where the distance is less than or equal to *n*?

Examples:

- (FOAF) Who are your friends, your friends' friends, and so on?

  [*Find* on Nodes + repeat (*Find Adjacent Nodes* on Nodes + *Retrieve Value* on Nodes) until no more new adjacent nodes are found]
- (FOAF) How many friends are you connected to in this way?

  [*Find* on Nodes + repeat (*Find Adjacent Nodes* on Nodes) until no more new adjacent nodes are found + *Count* on Nodes]
- (ARM) To what cities can we go from Seoul, Korea by changing planes only once?

  [*Find* on Nodes + repeat (*Find Adjacent Nodes* on Nodes + *Filter* on Links + *Retrieve Value* on Nodes) twice]

6.3.1.3 Common Connection

General Descriptions:

- Given nodes, find a set of nodes that are connected to all of them.

Examples:

- (FOAF) Find all the people who know both John and Jack.

  [*Find* on Nodes + *Find Adjacent Nodes* on Nodes + *Find* on Nodes + *Find Adjacent Nodes* on Nodes + *Set Operation(Intersect)* on Groups]

6.3.1.4 Connectivity

General Descriptions:

- Find the shortest path between two nodes.
- Identify clusters.
- Identify connected components.
- Find bridges.
- Find articulation points.

Examples:

- (ARM) What is the shortest path from Seoul, Korea to Athens, Greece?

  [*Find* on Nodes + repeat(*Find Adjacent Nodes* on Nodes in a breadth-first manner) until find the path]

- (FOAF) Count the number of clusters.

  [*Scan* on Graphs to count clusters]

- (FW) There may be subgraphs independent of each other. Count the number of connected components in the graph.

[*Scan* on Graphs to count connected components]

- (FOAF) Who is the person whose removal from the graph results in an unconnected graph?

  [*Scan* on Graphs to find an articulation point]

- (FW) Which is the eating link whose removal from the graph results in an unconnected graph?

  [*Scan* on Graphs to find a bridge]

6.3.2 Attribute-Based Tasks

All the previous topology tasks can be repeated with added filter, compute, range, or distribution tasks (opposed to solely count tasks) on the attributes either on nodes or on links.

6.3.2.1 On the Nodes

General Descriptions:

- Find the nodes having a specific attribute value.
- Review the set of nodes.

Examples:

- (FOAF) Who do you know from the people currently shown on screen?

  [*Filter* on Nodes + *Retrieve Value* on Nodes]

- (FOAF) How many people do you know from the ones currently shown on screen?

  [*Count* on Nodes while *Scan* on Nodes]

- (FOAF) Are there any foreign-sounding names?

[*Scan* on Nodes until find an answer]

6.3.2.2 On the Links

General Descriptions:

- Given a node, find the nodes connected only by certain types of links.
- Which node is connected by a link having the largest/smallest value?

Examples:

- (GO) Find the nodes connected by "is-a" relationships from the "Biological Process" node.

  [*Find* on Nodes + *Find Adjacent Nodes* on Nodes + *Filter* on Links + *Retrieve Value* on Nodes]

- (FW) If a link has an attribute representing the percentage of the diet, what is main food of American crow?

  [*Find* on Nodes + *Find Adjacent Nodes* on Nodes + *Find Extremun* on Links + *Retrieve Value* on Nodes]

6.3.3 Browsing Tasks

6.3.3.1 Follow Path

General Descriptions:

- Follow a given path.

Examples:

- (FOAF) A user looks into A's friend B, B's friend C, and C's friend D.

[*Find* on Nodes + repeat (*Find Adjacent Nodes* on Nodes + *Scan* on Nodes) three times]

- (FW) Follow the flow of energy from grasses, to a rabbit that eats grass, to a carnivore that eats the rabbit, and to a carnivore that eats that carnivore.

  [*Find* on Nodes + repeat (*Find Adjacent Nodes* on Nodes + *Scan* on Nodes) three times]

6.3.3.2 Revisit

General Descriptions:

- Return to a previously visited node.

Examples:

- (FOAF) After they follow a path in the above task, they may want to see A's other friends.

  [*Scan* on Nodes + *Find Adjacent Nodes* on Nodes]

- (FW) Find another carnivore that eats the same rabbit.

  [repeat (*Scan* on Nodes) twice to find + *Find Adjacent Nodes* on Nodes]

6.3.4 Overview Task

This is a compound exploratory task to get estimated values quickly. For example, we might ask users to estimate the size of the social network. Note that sometimes it is more important to be able to estimate the answer than to get an accurate one. Some of the topology tasks can be done easily using an overview of the graph as well. For example, using particular layout algorithms, it is easy to see whether or not there are clusters and connected components. Overview also helps to find patterns.

6.4 High-Level Tasks

There are high-level tasks that are not covered by the above tasks.

- When comparing two or more food webs, we can ask the following questions: What do they have in common? What are the differences among those food webs? Is there any missing or conflicting information?
- Due to errors in the data, several nodes may represent the same entity. For example, the co-authorship graphs often have duplicate author nodes. One important task is to identify whether two or more nodes represent the same person.
- How has the graph changed over time?

6.5 Discussion

Evaluating complex interfaces is a challenge, especially in the field of Information Visualization [107]. A list of tasks for graph visualization would be helpful to designers who want to improve their systems and to evaluators who want to compare graph visualization systems. The development of three graph visualization systems for several different domains provided a set of tasks that can serve as a starting point. User studies of graph visualization techniques were reviewed to extract any missing tasks. Finally, those tasks were incorporated into the Amar's list [4]. This enables us to define graph-specific objects and demonstrate how all complex tasks could be seen as a series of low-level tasks performed on those objects. We believe that our taxonomy, associated with benchmark datasets and specific tasks, would help evaluators generalize results collected through a series of controlled experiments.

Chapter 7

Applications

As explained above, TreePlus is a pluggable software component. This chapter presents three applications that use TreePlus to show graph structures. I developed two of them myself to prove TreePlus can be used to rapidly develop graph visualization systems for various datasets. The third (NetLens) was built by Hyunmo Kang at UMD. This chapter also describes how these applications interact with TreePlus to accomplish sample tasks.

7.1 GOTreePlus: Gene Ontology Browser

I developed GOTreePlus (shown in Figure 7.1) to allow biologists to easily identify GO terms of importance and then visualize them in the gene ontology structure. I worked with Jinwook Seo, who collected the task requirements and prepared the dataset.

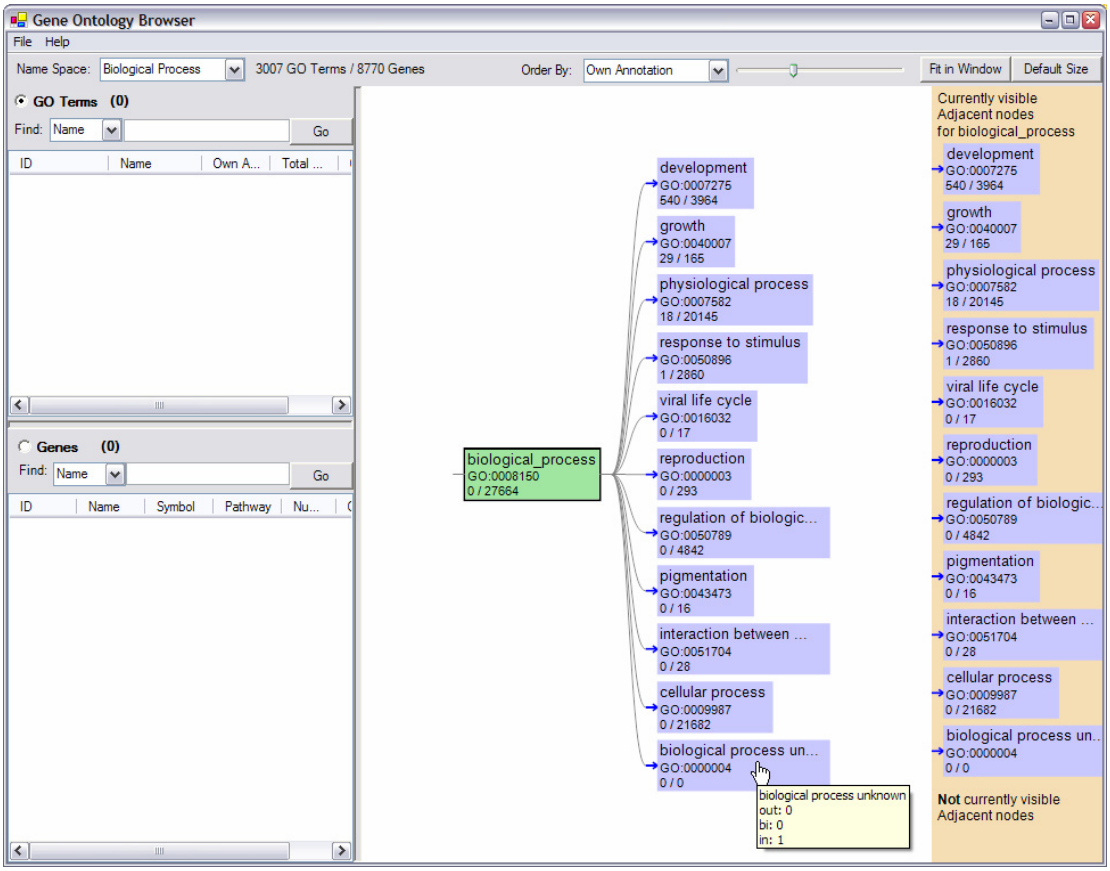


Figure 7.1 GOTreePlus consists of two lists (GO term list and gene list) on the left and TreePlus on the right.

7.1.1 Gene Ontology

Ontologies are specifications of a relational vocabulary. They provide a vocabulary for representing and communicating knowledge about a topic, and a set of relationships that hold among the terms of the vocabulary. The terms in a given vocabulary are likely to be restricted to those used in a particular field.

The Gene Ontology (GO) project [59] is a collaborative effort by the Gene Ontology Consortium to address the need for consistent descriptions of gene products in different databases. The GO collaborators are trying to provide three structured

networks of defined terms. Their goal is to describe gene product attributes by their associated biological processes, cellular components and molecular functions in a species-independent manner. A gene product will have one or more molecular functions, participate in one or more biological processes, and might be associated with one or more cellular components. For example, the gene product "cytochrome c" can be described by the molecular function term "electron transporter activity," the biological process terms "oxidative phosphorylation" and "induction of cell death," and the cellular component terms "mitochondrial matrix" and "mitochondrial inner membrane."

GO terms are organized as directed acyclic graphs (DAGs). This means that a child term can have many parent terms. For example, the biological process term "hexose biosynthesis" has two parents - "hexose metabolism" and "monosaccharide biosynthesis." This is because "biosynthesis" is a subtype of "metabolism," and a "hexose" is a type of "monosaccharide." If the child term describes the gene product, then all its parent terms must also apply to that gene product. When any gene involved in "hexose biosynthesis" is annotated to this term, it is automatically annotated to both "hexose metabolism" and "monosaccharide biosynthesis."

GO has two kinds of relationships: "is-a" and "part-of." The is-a relationship describes necessary specialization relations between properties (e.g., "a eukaryotic cell is a cell"). The part-of relationship means "can be a part of, not is always a part of." In addition, the part-of relationship is intended to behave transitively. GO uses part-of for representation of parts of both substances and processes (e.g., "activation"

is part-of "fertilization") and of functions/activities. Part-of appears also in each of the following kinds of statements:

- membrane part-of cell, intended to mean "a membrane is a part-of any cell"
- flagellum part-of cell, intended to mean "a flagellum is part-of some cells"
- replication fork part-of cell, intended to mean: "a replication fork is part-of the cell (nucleoplasm) only during certain times of the cell cycle"

AmiGO (shown in Figure 7.2) [5] is an HTML-based browser for the gene ontology. Users can browse and search both the terms and the gene product annotations. As you can see in Figure 7.2, "bud" is a part of "cell", which is a "cellular component." Gene ontology is managed by the GO Consortium [59] and available in many different formats including text files, XML files, and MySQL database.

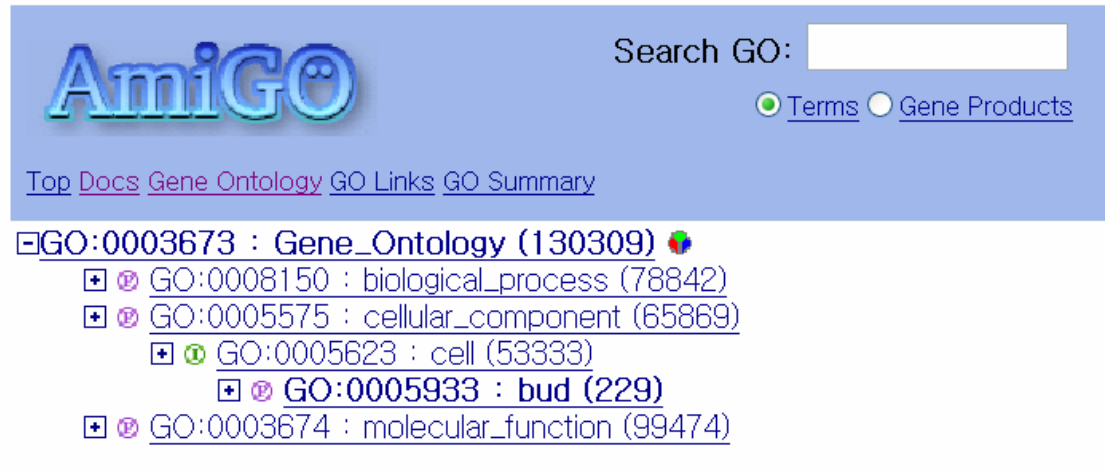


Figure 7.2 AmiGO browser; P represents "part-of" and I represents "is-a."

7.1.2 System Description

GOTreePlus (Figure 7.1) consists of the GO term list, the gene list, and the TreePlus control. When users open a file containing a list of genes annotated with GO terms, the number of annotations for each GO term is computed and shown in the gene ontology structure. In TreePlus, each node representing a GO term has four attributes: name, id, and the number of its own annotations and sum of its descendents' annotations. Since the nodes in TreePlus within the system, by default, are sorted by the number of its own annotations, users can easily see which GO term is most often annotated. For example, among the GO terms that have an "is-a" relationship with "biological process," "development" is annotated the most and "biological process unknown" is not annotated at all (Figure 7.1).

Since three name spaces – Biological Process, Cellular Component, and Molecular Function – are disjoint from each other, GOTreePlus visualizes one name space at a time. Users can select one name space among three using the combo box above the GO term list. The number of the selected genes and the number of their associated GO terms are also shown right next to the combo box. The search for GO terms is performed within the selected name space.

GOTreePlus provides a way to search for specific GOTerms – a simple substring match either by name (e.g., "cell differentiation") or by id (e.g., "GO:0007242"). Search results are shown in the GO term list. When users select a GO term from the list, the selected term is shown in the gene ontology structure in TreePlus. If the "GO Term" radio button is selected, the gene list is updated with the

genes associated with the selected GO terms. The number of genes in the gene list is also updated and shown by the 'Genes' radio button.

Similarly, users can also search genes by name (e.g., "placental growth factor"), symbol (e.g., "Newrod4"), and pathway information ("Purine metabolism"). Labels in the gene list are grayed out if there is no GO term associated with the gene in the current name space. When users select a gene from the list, all GO terms related to the selected gene are shown in the gene ontology structure in TreePlus. For example, the "160312_at" gene is annotated by two GO terms, "sensory perception and chemotaxis" which share "response to stimulus" in the path from the root node, "biological_process" (Figure 7.3). If the 'Genes' radio button is selected, the GO term list is updated with the GO terms associated with the selected genes. The number of GO terms in the list is also updated and shown by the "GO Term" radio button.

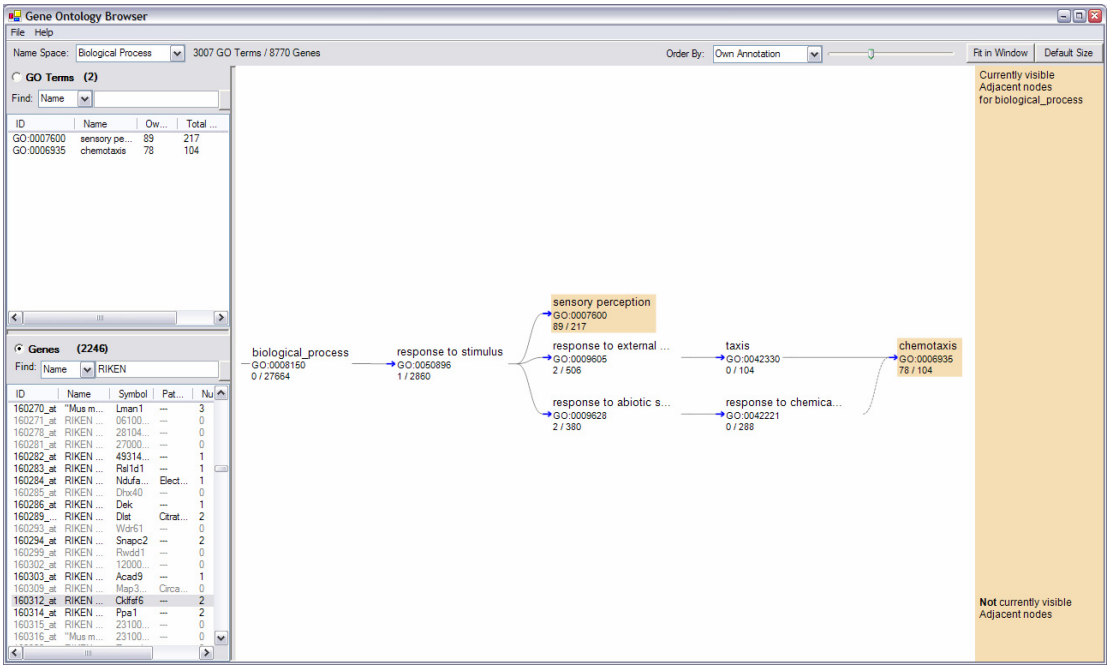
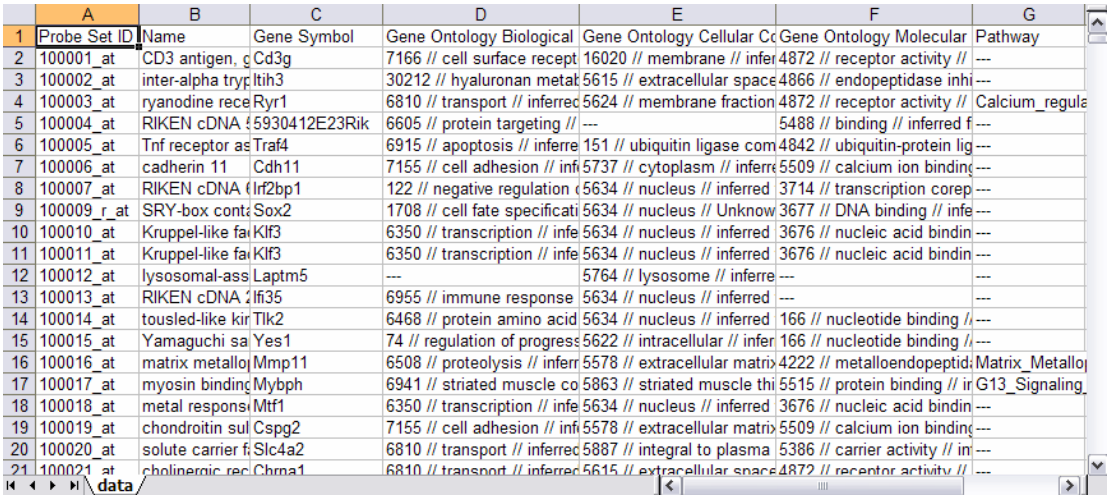


Figure 7.3 When users select a gene from the gene list, its associated GO terms are shown both in the GO term list and in the gene ontology structure in TreePlus.

7.1.3 Implementation Details

GOTreePlus was implemented in C# using the TreePlus control. It consists of only 4 classes and about 1,640 lines of code. GOTreePlus needs two input files; gene ontology in the GraphML file format and user data usually generated from an experiment in a tab-separated text file. The gene ontology file is loaded into memory once at startup since it is a fixed structure. Once the program is launched, the second file is opened and read into memory by users. To help users read the data file, the first row serves as a header showing the column information (Figure 7.4). Each row represents one probe set.



	A	B	C	D	E	F	G
1	Probe Set ID	Name	Gene Symbol	Gene Ontology Biological	Gene Ontology Cellular Co	Gene Ontology Molecular	Pathway
2	100001_at	CD3 antigen, g	Cd3g	7166 // cell surface recept	16020 // membrane // infer	4872 // receptor activity //	---
3	100002_at	inter-alpha tryp	Itih3	30212 // hyaluronan metab	5615 // extracellular space	4866 // endopeptidase inhi	---
4	100003_at	ryanodine rece	Ryr1	6810 // transport // inferred	5624 // membrane fraction	4872 // receptor activity //	Calcium_regula
5	100004_at	RIKEN cDNA 5	5930412E23Rik	6605 // protein targeting //	---	5488 // binding // inferred f	---
6	100005_at	Tnf receptor as	Traf4	6915 // apoptosis // inferre	151 // ubiquitin ligase com	4842 // ubiquitin-protein lig	---
7	100006_at	cadherin 11	Cdh11	7155 // cell adhesion // infe	5737 // cytoplasm // inferre	5509 // calcium ion binding	---
8	100007_at	RIKEN cDNA 1	Irf2bp1	122 // negative regulation o	5634 // nucleus // inferred	3714 // transcription corep	---
9	100009_r_at	SRY-box cont	Sox2	1708 // cell fate specificati	5634 // nucleus // Unknow	3677 // DNA binding // infe	---
10	100010_at	Kruppel-like fa	Klf3	6350 // transcription // infe	5634 // nucleus // inferred	3676 // nucleic acid bindin	---
11	100011_at	Kruppel-like fa	Klf3	6350 // transcription // infe	5634 // nucleus // inferred	3676 // nucleic acid bindin	---
12	100012_at	lysosomal-ass	Laptm5	---	5764 // lysosome // inferre	---	---
13	100013_at	RIKEN cDNA 4	Ifi35	6955 // immune response	5634 // nucleus // inferred	---	---
14	100014_at	tousled-like kir	Tlk2	6468 // protein amino acid	5634 // nucleus // inferred	166 // nucleotide binding /	---
15	100015_at	Yamaguchi sa	Yes1	74 // regulation of progress	5622 // intracellular // infer	166 // nucleotide binding /	---
16	100016_at	matrix metallo	Mmp11	6508 // proteolysis // infer	5578 // extracellular matrix	4222 // metalloendopeptid	Matrix_Metallo
17	100017_at	myosin binding	Mybph	6941 // striated muscle co	5863 // striated muscle thi	5515 // protein binding // in	G13_Signaling_
18	100018_at	metal respons	Mtf1	6350 // transcription // infe	5634 // nucleus // inferred	3676 // nucleic acid bindin	---
19	100019_at	chondroitin sul	Cspg2	7155 // cell adhesion // infe	5578 // extracellular matrix	5509 // calcium ion binding	---
20	100020_at	solute carrier f	Slc4a2	6810 // transport // inferred	5887 // integral to plasma	5386 // carrier activity // in	---
21	100021_at	cholinergic rec	Chrna1	6810 // transport // inferred	5615 // extracellular space	4872 // receptor activity //	---

data

Figure 7.4 Sample user data file for GOTreePlus opened with the Microsoft Excel.

7.2 EcoLens: Exploring Food Webs

I developed EcoLens (shown in Figure 7.5) to allow biologists to browse through a collection of food webs, find webs of interest, and then visualize an individual food web. I worked with Cynthia Sims Parr and Benjamin B. Bederson.

Food web datasets consist of several elements such as food webs, taxa, and habitats. Inspired by our successful experience with PaperLens, EcoLens provides an abstract overview and tightly couple multiple views to show relationships among data elements. Within each food web, food relationships between taxa are important as well. Therefore, our design combines an overview technique with TreePlus' guiding metaphor, "Plant a seed and watch it grow," described before. Through the overview users can easily find not only interesting patterns in the dataset but also particular webs of interest. Once they find desired webs to look at, they can investigate an individual food web using TreePlus.

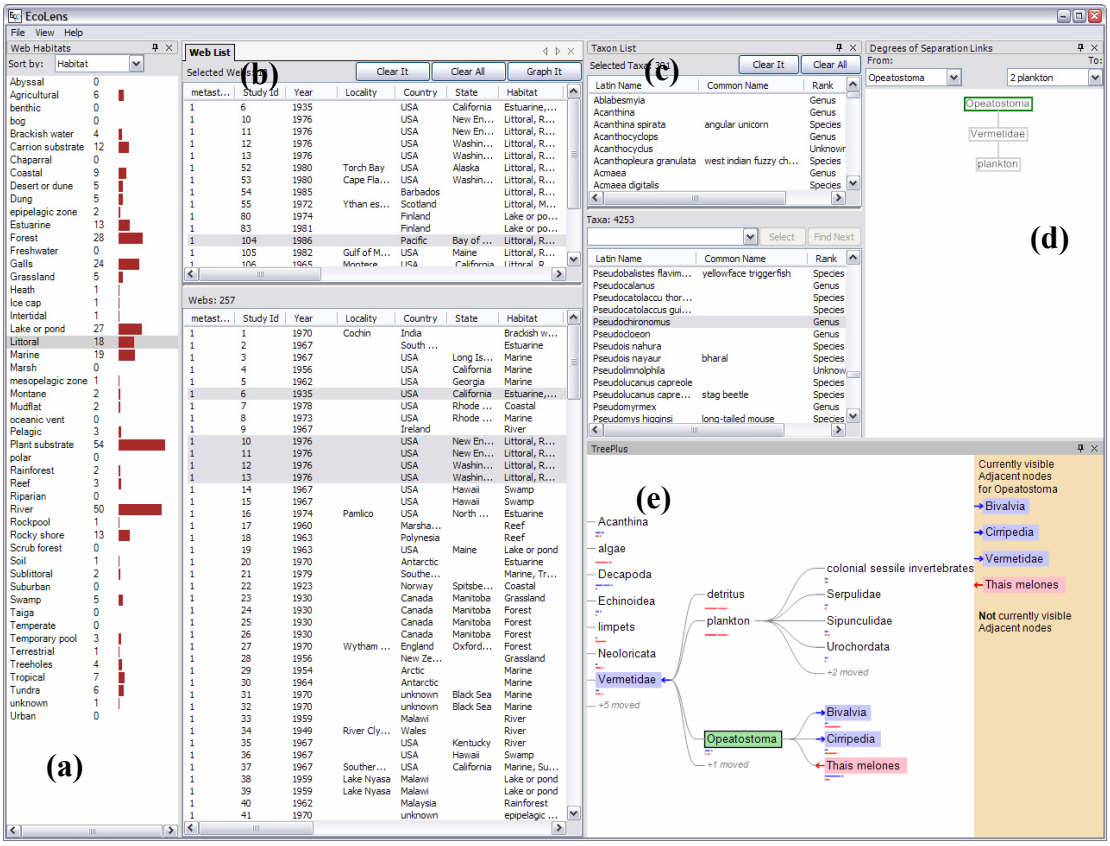


Figure 7.5 EcoLens enables biologists to explore a collection of food webs; (a) Web Habitats (b) Web List (c) Taxon List (d) Degrees of Separation Links (e) TreePlus.

7.2.1 Food Web

A food web is a system of relationships between plants, animals, and energy. It shows the food relationships among organisms in a community. These connections are very important to the understanding of any ecosystem.

Food webs are actually made up of two or more interconnected food chains. A food chain, explains what an organism might eat, and what might eat that organism in a specific scenario. Most animals are part of more than one food chain and eat more than one kind of food in order to meet their food and energy requirements.

These interconnected food chains form a food web. For example, one food chain could be grasses → grasshopper → herring → bald eagle and another could be grasses → grasshopper → salmon → harbour seal → killer whale, as shown in Figure 7.6. Most food chains have no more than five links. There cannot be too many links due to the energy lost at each step in the chain. Food webs are directional and can be cyclic.

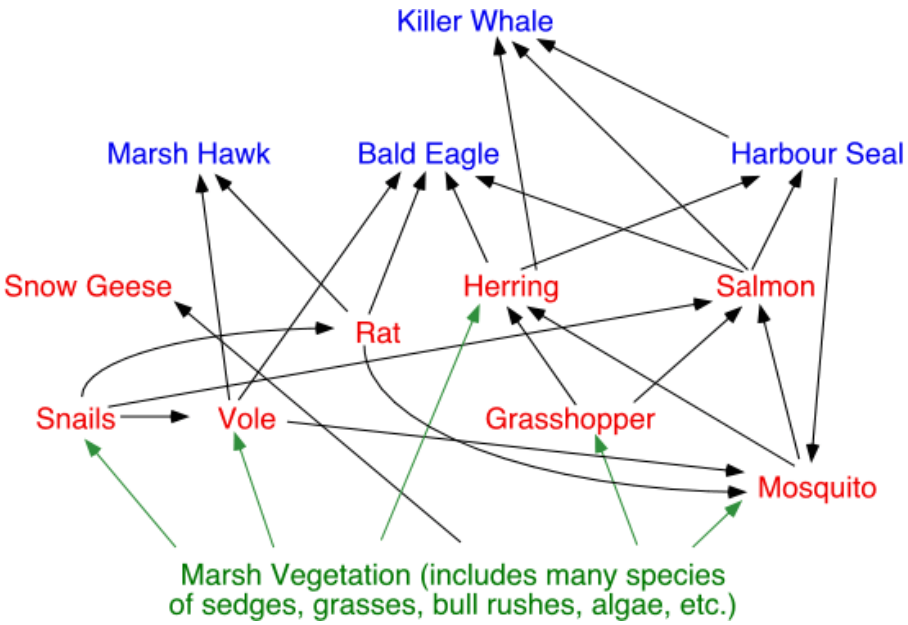


Figure 7.6 Sample food web

A change in the size of one population in a food chain will affect other populations in that chain. This interdependence of the populations within a food chain can contribute to stability but also can lead to far-reaching consequences if part of the web is disturbed. For example, when there are too many grasshoppers, there will not be enough grass for all of them to eat. Many grasshoppers will starve and die. Fewer grasshoppers mean more time for the grass to grow and less food for the

herring to eat. Some of the herring will also die. When there are fewer herrings, the grasshopper population will again increase.

7.2.2 System Description

As shown in Figure 7.5, EcoLens consists of five main views: a) web habits; b) web list; c) taxon list; d) degrees of separation links (DOSL); and e) TreePlus. The web habitats view provides an overview of the habitats by showing the list of habitats with the number of food webs in each habitat. Users can sort the view either by habitat name or the number of webs. The bottom of the web list view shows all the webs in the database. When some webs are selected either by users or by the system, they are shown in the Selected Webs list at the top of the web list view. Similarly, the taxon list view contains all the taxa in the database and the currently selected taxa are shown in the Selected Taxa list. When users double click on a taxon in the Selected Taxa list, EcoLens opens a dialog box to show the information of studies that contain the selected taxon (Figure 7.7). The TreePlus view visualizes an individual food web as a node-link diagram and the DOSL view shows one of the food chains from one taxon to the other in the web currently being visualized by the TreePlus view.

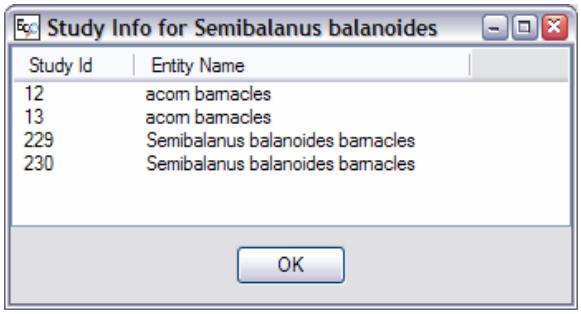


Figure 7.7 When users double click on a taxon, "Semibalanus balanoides," in the Selected Taxa list, EcoLens opens a dialog box to show the information of studies that contain the selected taxon.

These views are tightly coupled. When users select a habitat in the web habitats view, all the webs from the selected habitat are highlighted in the Webs list. Furthermore, they are displayed in the Selected Webs list for easy access. In addition, all the taxa in these selected webs are highlighted in the Taxa list and displayed in the Selected Taxa list. For these three views - web habits, web list, and taxon list - user interactions are symmetric. For example, users can select webs from the Webs list to see habitats for particular food webs or get lists of taxa. Habitats of the selected webs are highlighted in the web habitats view and taxa in the selected webs are shown in the taxon list view. Users can also copy reference information of the selected food webs to the clipboard by selecting the Copy Reference menu option after right clicking on the selected food webs.

Users may visualize an individual food web in TreePlus to see trophic links among taxa by double clicking on a web either in the Selected Webs list or in the Webs list. They can also press the "Graph It" button after clicking on a web in the Selected Webs list. EcoLens then builds a food web from the database and visualizes

it within TreePlus. Since it uses the default root selection mechanism in TreePlus, the taxon with the most connections with others is chosen to be the root. EcoLens also adds all taxa in the visualized web to the "From" combo box in the degrees of separation links view. Once a taxon is selected from the "From" combo box, EcoLens displays all the taxa reachable from the selected taxon through valid food chains in the "To" combo box with the corresponding degrees of separation. When a taxon is selected from the "To" combo box, EcoLens displays one of the shortest food chains between two taxa. When users click on a node in the degrees of separation links view, EcoLens opens the selected taxon within TreePlus. Similarly, when users click on a node in the TreePlus view, EcoLens highlights the selected taxon within the degrees of separation links view if it is already displayed.

7.2.3 Implementation Details

EcoLens is implemented in C# and consists of 40 classes and about 6,800 lines of code. To visualize each food web, it uses the TreePlus control. The web habitats and degrees of separation links views are implemented with Piccolo.NET. EcoLens accesses the MySQL server using a MySQL data provider, which links a data source and .NET code. A current MySQL database schema is shown in Figure 7.8 and the data are currently being used by the SPIRE project [126]. To make each window dockable, it uses the DockPanel Suite [36], an open source docking library. EcoLens is now available for download at http://www.cs.umd.edu/biodiversity/#EcoLens.

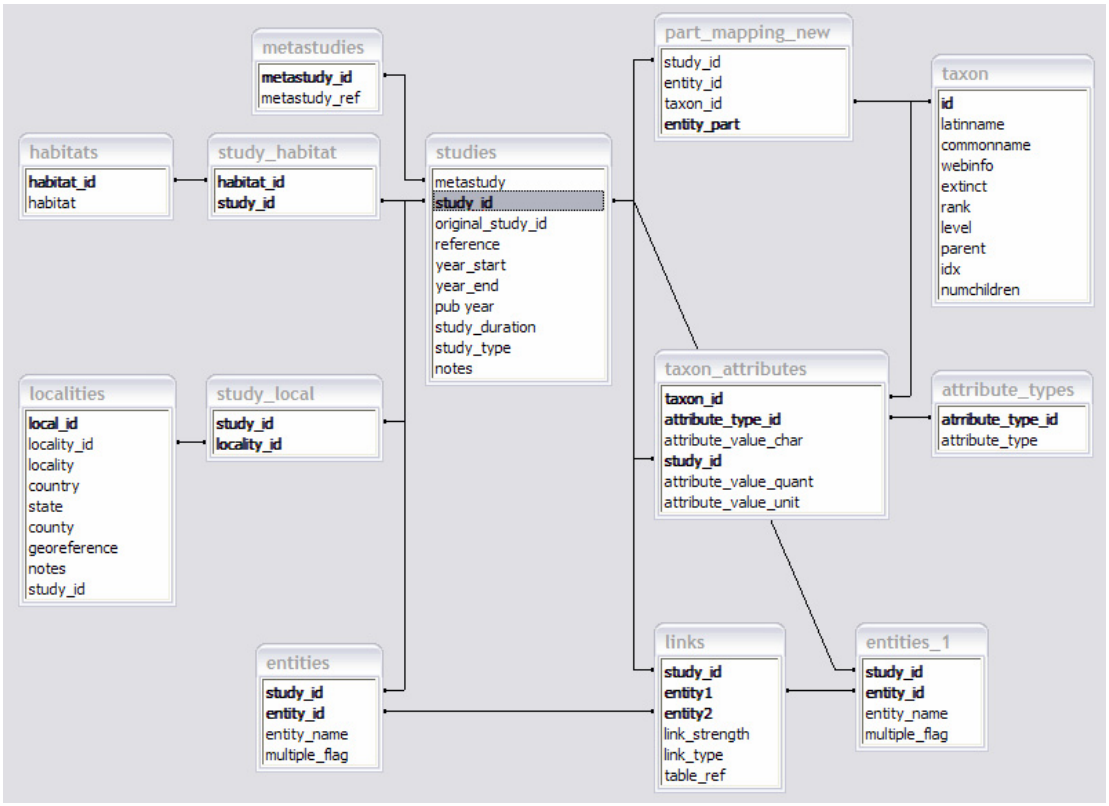


Figure 7.8 Database schema for EcoLens

The Model-View-Controller (MVC) architecture [25] was used to separate the data model, user interface, and control logic. The model object is the domain-specific representation of the information. It notifies the view object when information changes and handles the state change requests from the controller object. The view object renders the model into a graphical or textual representation suitable for interaction. The controller object is the means by which users interact with the application. It responses to events, typically from users, and informs the model and view objects to perform actions.

7.3 NetLens

NetLens (Figure 3.12) is a prototype that demonstrates a general and scalable approach to analyzing common kinds of network data such as conference publications with citations and author information. I was a part of the project team designing the interface that generalizes PaperLens. But, it was implemented by Hyunmo Kang, a member of HCIL.

NetLens is general in that it applies to any dataset that can be represented as a bipartite graph. While this means that there must be just two entity types, this restriction can be relaxed. Each entity has attributes and relationships within and between those entities. For example, in conference publications, the two entity types are papers and people. Papers have attributes such as titles, abstracts, and keywords. They cite and are cited by other papers, and are authored by people. Those people have attributes such as institution and fields of interest, and can be connected to papers through authorship relations and to other people through co-authorship relations. This basic structure applies equally to digital photo collections and email collections.

NetLens is also scalable because it applies to a standard relational database and the interface is built of common simple components such as histograms and lists. Together, they offer surprisingly rich support for real tasks. By avoiding visual overviews of the entire dataset that display a visual element for each entity instance, it is possible to avoid immediate problems of scalability.

The interaction is based on a visual user interface that lets users incrementally explore the dataset and refine their queries. In doing so, NetLens supports realistic

but traditionally difficult queries such as determining trends of each sub-fields are hot and which are on the decline, finding appropriate experts to review a paper or serve as an expert witness in patent litigation, or even determining a good place to go on a sabbatical based on an analysis of one's publications.

While NetLens is particularly good at showing distributions, filtering, and sorting, it does not provide good support for browsing through a graph structure in the dataset and showing relationships between two objects that are not directly connected. Thus, NetLens was enhanced by integrating it with TreePlus. Users can first find a set of authors of interest to them using NetLens and then start browsing from those authors using TreePlus. They can also see how those authors are connected to another author as shown in Figure 7.9 and continue exploring.

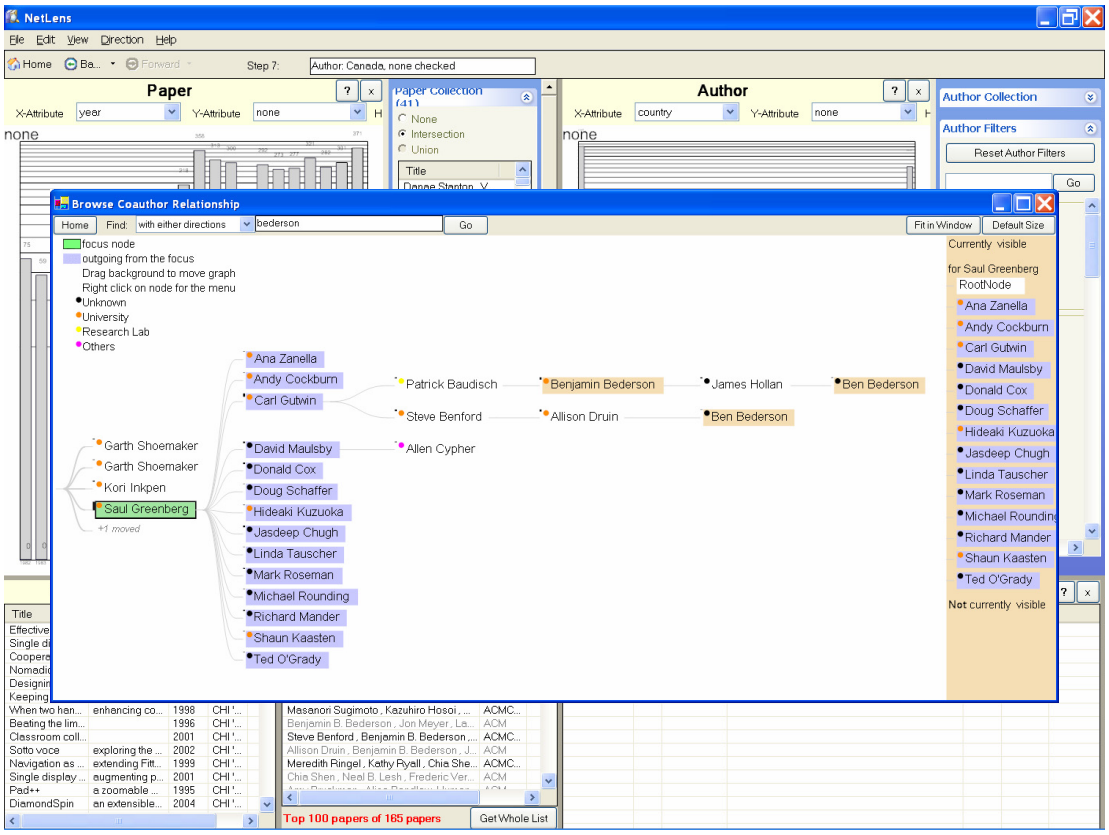


Figure 7.9 TreePlus is integrated with NetLens to show co-authorship graphs.

7.4 Discussion

Developed as a reusable component with a well defined API, TreePlus makes it possible to rapidly build visualization systems that need to show graph structures whether it is directed or not. Although only three visualization systems using TreePlus to show graph structures were presented, TreePlus is applicable to other datasets as well.

Furthermore, a set of preferences enable TreePlus to be configured to best show the graph structures according to the characteristics of the datasets and main task requirements. For example, GOTreePlus sets a root defined by gene ontologies and sorts nodes by the number of its own annotations. NetLens uses a dummy root if

users want to start from a set of authors and sets the categorical attribute to show the institution information.

EcoLens and NetLens demonstrate not only how well TreePlus can complement the overview first approaches, but also shows that the PaperLens concept is useful for analyzing common network data. The overview and rich filtering of NetLens help users find a set of interesting items. The TreePlus view in NetLens enables users to capture relationships among several authors that cannot be easily captured with NetLens itself.

Chapter 8

Conclusion

Due to the broad applicability of graphs, a vast amount of graph visualization research has been done over the last few decades, mostly on node-link representations. However, it is still difficult to produce effective interactive layouts for large graphs. Dense layout and occlusion make food webs, ontologies, and social networks difficult to understand and interact with. To tackle this problem, I created and applied several design principles to various graph visualization domains in novel ways. This dissertation also presented three graph visualizations I developed.

PaperLens is a tool developed to visualize conference proceedings. It provides an abstract overview of the full dataset and shows relationships within a complex network through interactive highlighting. PaperLens received a first place award in the InfoVis 2004 Contest and has been scaled up to the larger set of CHI conference papers. Furthermore, this visualization technique has been applied to other datasets as well.

TaxonTree was developed to visualize the taxonomic hierarchy of animal names. This dissertation discussed the decisions behind and lessons learned from TaxonTree design and concluded that interactive tree visualization could be applied to the biodiversity domain for a broad audience to help users understand the data. TaxonTree has been deployed at University of Michigan's Animal Diversity Web (http://animaldiversity.ummz.umich.edu). The development and evaluation gave us valuable insights for designing a general interactive graph visualization component.

TreePlus was designed by applying the lessons learned from TaxonTree to the broader problem of graph visualization. Our approach involves transforming a graph into a tree plus cross links (i.e. the additional links that are not represented by the spanning tree) and using visualization and interaction techniques to reveal the missing graph structure while preserving readability of the labels. A guiding metaphor of "Plant a seed and watch it grow" allows users to start with a node and expand the graph as needed, which complements the classic overview techniques that are effective at - but often limited to - revealing clusters.

8.1 Contributions

The main contributions of this dissertation are the:

- Creation and application of several design principles to various graph visualization domains. Tightly-coupled and highly customized views were used for graph visualization in a novel way. A new tree layout approach was proposed with appropriate visualization and interaction techniques. When visualizing graphs as trees, a guiding metaphor "Plant a seed and watch it grow" was used to support information gathering and detailed exploration of the graph's local structure.
- Design and implementation of PaperLens, a novel visualization system that enables users to reveal trends, connections, and activity throughout a conference community. The interface offers a compelling alternative to more common node-link diagram visualizations.

- Design and implementation of TaxonTree, a visualization system for animal classification. The extension of an existing tool enabled support for database access and web deployment, while accommodating domain-specific requirements.
- Qualitative evaluation of TaxonTree. The description of how users in the biodiversity domain approach information retrieval provides a rich demonstration of the value of interactive tree visualization with integrated searching and browsing.
- Design and implementation of a new interactive graph visualization component called TreePlus based on a tree-style layout. This includes the development of special visualization and interaction techniques to efficiently visualize graphs as trees.
- Empirical evaluation of TreePlus. The controlled experiment comparing TreePlus to a traditional graph visualization shows that, in general, the advantage of TreePlus over the traditional interface increases as the density of the displayed data increases.
- Development of a taxonomy of tasks for graph visualization.
- Examples of the use of these techniques. The integration of TreePlus with the next generation of PaperLens will offer great potential to the field of information visualization.

8.2 Lessons Learned

Here are some conclusions learned from all three visualizations. First, there are good alternatives to node-link graph visualizations aside from matrix representations. While other graph visualizations provide overviews of the entire dataset, displaying a visual element for each node and link, PaperLens provides an abstract overview and shows relationships through interactive highlighting. PaperLens enabled discovering many interesting patterns and relationships which could not have been revealed using existing tools. PaperLens and NetLens can also facilitate many tasks that require showing a distribution of items and filtering and sorting items, especially when reading of labels and attributes is important. Second, tree layout has strong potential to visualize general graphs. The TaxonTree study provided further evidence for the value of interactive tree visualization and integrated searching and browsing in information retrieval and understanding. In addition to the stability of the tree layout, alignment, grouping, and sorting of children are definite plus over the traditional graph layout. Finally, while it is often ignored in the overview-first approaches, label readability is very important for users to perform many tasks that require interpreting graph data.

NetLens and TreePlus each have their own place. The filtering, grouping, and sorting features of NetLens enable users to find an area of interest from a large graph. This can serve as a starting point for TreePlus. And the incremental exploration provided by TreePlus helps users understand the local structure of the graph. The information found in the local structure can be sent back to NetLens to find other

related areas. When combined, these approaches can make large graphs easier to understand and interact with.

8.3 Future Work

Given the difficulty faced by just scaling PaperLens from InfoVis to the larger CHI dataset, an excellent research exercise would be to move toward another order of magnitude in terms of the number of documents, authors and references. At that point, the tool could be used more as a portal to move in and out of different journals or conferences. NetLens has been developed by generalizing PaperLens to handle other datasets. The next step is to scale NetLens to a much larger dataset of documents such as the ACM Digital Library with many other kinds of metadata.

TreePlus was a next step of TaxonTree that gave us valuable insights. Now that TaxonTree is deployed at the Animal Diversity Web, it would be interesting to compare TaxonTree with the web classification interface represented by indented hierarchies.

While TreePlus was implemented as a research prototype, the promising result of the controlled experiment and successful integration with other tools present numerous possible future directions. The following sections summarize potential future work for TreePlus.

8.3.1 Scaling

The current implementation of TreePlus is general in that it supports an abstract graph data structure that can be represented in GraphML under the assumption that there can be at most one link between two nodes. However, it is not scalable because

TreePlus requires the entire graph to be loaded into memory to perform necessary graph operations. Scaling TreePlus would require that data be incrementally loaded from standard databases. It would also require pre-computing some values, such as a shortest path length between two nodes, to reduce the number of accesses to databases. For example, all pairs shortest paths could be pre-computed by using the Floyd-Warshall algorithm. This algorithm stores only one predecessor for each node, generating only one shortest path among many possible ones. Since it is needed to know all the shortest paths between two nodes, the algorithm should be modified so that it stores all predecessors for each node.

8.3.2 Coupling with Overview

TreePlus' incremental graph exploration can complement the classic overview techniques that are effective at revealing clusters and interesting patterns. There are two ways to accomplish the integration of TreePlus and overviews. First, as shown in Chapter 7, TreePlus can be plugged into other tools that support overviews such as EcoLens and NetLens. While these tools provide abstract overviews using histograms and tables, the containing application can provide a regular node-link overview. Second, the classical overview can be incorporated into TreePlus itself. For example, an overview can be added to the left of the tree browser. In either case, it is essential to provide tight coupling between the overview and the tree browser.

8.3.3 Further Evaluation

At the start of the user study our biggest concern was that it might be confusing to look at graphs as trees. The encouraging results of the controlled experiment lead us

to wonder if this might actually be a natural way for people to interpret graphs. Rather than thinking about the whole graph, users can remain focused on the adjacent node relationships, which are well represented as parent-children links in a tree. Most participants who preferred TreePlus said it was not confusing to look at the tree representations. One participant said that "While I was doing the tasks, I did not think of [TreePlus] as a tree." Other comments included "I was very comfortable using it because I am used to the hierarchical structure" and "I think trees are logical and ordered arrangements of the graphs." Many participants really liked the tree layout and the alignment, grouping, and sorting of the children. Obviously further evaluations are needed to investigate this hypothesis, which could have important and testable implications for interactive graph visualization.

Now that TreePlus was tested as a general graph visualization component, it would be useful to conduct another qualitative evaluation of TreePlus in a domain-specific context, as we did with TaxonTree.

As described before, one way to make the tree layout completely stable is to force adjacent nodes to be close to each other by duplicating the cross-linked nodes. TreePlus now provides an option to duplicate nodes instead of moving them to represent cross links. This approach could work well for graphs that have a modest number of cross links. It would be interesting to see which representation of cross links work better as graph density varies.

8.3.4 Additional Functionality

Adding the following features would increase the utility of TreePlus.

- While current TreePlus assume a homogeneous node type, graphs often have different types of nodes. Visualizing multiple types of nodes would help users analyze graphs.
- Visualization of link attributes including labels would help users better understand graphs with such attributes.
- As explained in Section 7.2, there are cases where we need to support multiple links between two nodes.
- It would also be useful to enable users to interact with links, for example by selecting a link.
- Given the fact that TreePlus changes the tree structure to show adjacent nodes close to the selected node, it is often difficult to keep track of several nodes of interest. Thus, providing a way to mark important nodes may help users analyze the graph data.
- The tree representation helps users trace back the path they followed. However, if that path is long it may require a lot of panning. If we show the path to the root at the top of the tree browser, it would be easier for users not only to read the path but also to select nodes along it.
- It would also be useful to better support multiple node selection. Currently users can give focus to multiple nodes by clicking them while pressing the control key. This is cumbersome if they have to select many nodes at once. Since dragging pans the tree, we can support marquee selection when users drag while holding the control key.

8.3.5 Better Support for Large Fan-out

As discovered in the user study, panning is a big issue with TreePlus. The improved multi-column layout helps users see children of the currently opened node at once without panning. However, if the number of children is very large, users still have to pan horizontally. Long labels aggravate this problem. One possible way to tackle this problem is to use a bifocal technique as shown in the mockup (Figure 8.1).

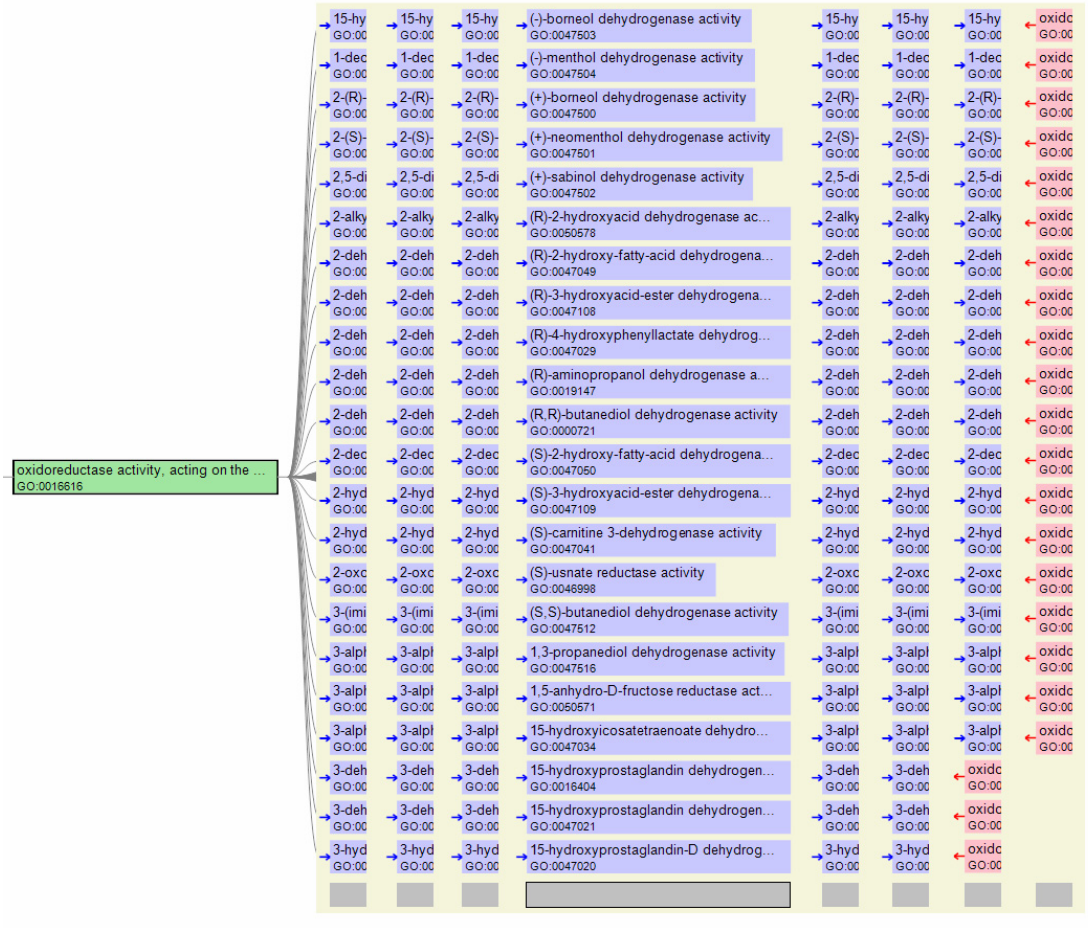


Figure 8.1 Mockup for a possible bifocal technique to avoid panning in the multi-column layout. The fourth column is expanded since it has the focus.

8.3.6 Graph Editor

Assuming that we are using the GraphML format or system-dependent XML file format, it is possible to create a graph using a text editor and then visualize it with a tool such as TreePlus. However, it would be difficult to manipulate large graphs this way. While the simplest solution is to combine a text editor with TreePlus, it would be most desirable to extend TreePlus to support editing capabilities. For example, we could enable users to create nodes and edit labels and other attributes within TreePlus. Links between nodes could be created by drag-and-drop.

8.3.7 Web Deployment

Due to the popularity of online graph resources such as social networks and gene ontologies, web deployment would increase the utility of TreePlus. Since TreePlus is implemented as a .NET control, it can be embedded in Internet Explorer. However, there are some issues to contend with since the .NET security model places additional restrictions on embedded controls. For example, the default security settings on most machines restrict keyboard input and will not allow users to load external dlls. To get around these issues, the client could fully trust our site or decrease their security settings, but that might not be desirable. Therefore, additional work remains to get TreePlus to work over the web for most browsers, as described in the Developer's FAQ for piccolo [106].

Appendix A

TaxonTree Database

A.1 Taxon Table

- id: identifier
- latinname: scientific name
- commonname: common name
- webinfo: existence of external web pages
  4 bits are used to specify four available web sites
- BSCI224
- BSCI224leaf
- extinct
- rank
- level: level in a tree (start 0 for the root)
- parent: parent id
- idx: index among siblings
- numchildren: number of children

A.2 WebInfo Table

- taxon_id
- url: web page address
- source: id for the website
  0: University of Michigan's Animal Diverstiy Web
  1: UC Museum of Paleontology
  2: Tree of Life
  3: BSCI224

A.3 Synapomorphy Table

- taxon_id
- synapomorphy

Appendix B

Tasks for the TreePlus Controlled Experiment

B.1 Task 1 – Find

Find a person that is currently displayed. The person might be off screen.

You will be given 2 training trials (they are not timed and do not count; and you can ask questions) and 5 timed trials.

Task 1.1 (Training): Find and click on "Isaac Williams."

Task 1.2 (Training): Find and click on "Steven Jones."

Task 1.3 (Timed): Find and click on "Julian Jones."

Task 1.4 (Timed): Find and click on "Olivia Davis."

Task 1.5: Find and click on "Timothy Jones."

Task 1.6: Find and click on "Matthew Smith."

Task 1.7: Find and click on "Matthew Smith" again.

B.2 Task 2 – Browse

Follow a path.

2 training trials and 4 timed trials.

Task 2.1 (Training): Follow the path "Michael Smith" -- "Jesus Jones" -- "Alexander Smith."

Task 2.2 (Training): Follow the path "Cameron Williams" -- "Jonathan Johnson" -- "David Smith." Now find and select "Cameron Williams" again.

Task 2.3 (Timed): Follow the path "Sarah Davis" -- "Ian Jones" -- "Jayden Jones."

Task 2.4 (Timed): Follow the path "Anthony Smith" -- "Madison Davis" -- "Gabriel Johnson." Now find and select "Anthony Smith" again.

Task 2.5 (Timed): Follow the path "Jonathan Johnson" -- "Evan Johnson" -- "Carter Brown."

Task 2.6 (Timed): Follow the path "Ella Miller" -- "Vanessa Moore" -- "Ashley Davis." Now find and select "Ella Miller" again.

B.3 Task 3 – Adjacency

Among all those who communicate with a specific person, count those with a given characteristic.

2 training trials and 4 timed trials.

Task 3.1 (Training): Among the people who are communicating with "James Smith," count how many have the last name Smith.

Task 3.2 (Training): Among the people who sent an email to "Luis Williams," count how many people live in Maryland (MD).

Task 3.3 (Timed): Among the people who are communicating with "Mason Williams," count how many have the last name Smith.

Task 3.4 (Timed): Among the people who received emails from "Wyatt Brown," count how many people live in California (CA).

Task 3.5 (Timed): Among the people who are communicating with "Megan Miller," count how many have the last name Moore.

Task 3.6 (Timed): Among the people who sent an email to "Adrian Jones," count how many people live in Florida (FL).

B.4 Task 4 – Accessibility

Count people with a given characteristic within two links (distance 2) of a given person.

2 training trials and 3 timed trials.

Task 4.1 (Training): Among those who are one or two incoming email links away from "Cody Brown" (less than or equal to distance 2), count how many people live in Texas (TX).

Task 4.2 (Training): Among those who are one or two outgoing email links away from "Aaron Williams" (less than or equal to distance 2), count how have the last name Smith.

Task 4.3 (Timed): Among those who are one or two incoming email links away from "Bryan Jones" (less than or equal to distance 2), count how many people live in Illinois (IL).

Task 4.4 (Timed): Among those who are one or two outgoing email links away from "Maya Taylor" (less than or equal to distance 2), count how many have the last name Brown.

Task 4.5 (Timed): Among those who are one or two outgoing email links away from "Alexander Smith" (less than or equal to distance 2), count how many people live in Maryland (MD).

B.5 Task 5 – Common Connection

Find a person who has been in direct email communication with two given people.

2 training trials and 4 times trials.

Task 5.1 (Training): Point with your finger to all the people directly communicating with both "Ryan Smith" and "Connor Johnson."

Task 5.2 (Training): Point with your finger to all the people directly communicating with both "Carlos Jones" and "Jake Brown."

Task 5.3 (Timed): Point with your finger to all the people directly communicating with both "Grace Davis" and "Owen Jones."

Task 5.4 (Timed): Point with your finger to all the people directly communicating with both "Jonathan Johnson" and "Adrian Jones."

Task 5.5 (Timed): Point with your finger to all the people directly communicating with both "Chase Jones" and "Katelyn Wilson."

Task 5.6 (Timed): Point with your finger to all the people directly communicating with both "Jordan Williams" and "Sydney Miller"

B.6 Task 6 – Connectivity

Find who has the most email relationships with other people in a group.

2 training trials and 4 timed trials.

Task 6.1 (Training): Of all the people who email with "Dominic Brown," click on the one who is in email contact with the most of the others.

Task 6.2 (Training): Of all the people who email with "Devin Brown," click on the one who is in email contact with the most of the others.

Task 6.3 (Timed): Of all the people who email with "Kimberly Moore," click on the one who is in email contact with the most of the others.

Task 6.4 (Timed): Of all the people who email with "Seth Brown," click on the one who is in email contact with the most of the others.

Task 6.5 (Timed): Of all the people who email with "Colin Brown," click on the one who is in email contact with the most of the others.

Task 6.6 (Timed): Of all the people who email with "Hannah Davis," click on the one who is in email contact with the most of the others.

Appendix C

Description of the Classes in Graph Library

C.1 Public Properties of TreePlus

- `public ContextMenu ContextMenu { get; set; }`
- `public Preferences Preferences { get; set; }`
- `public Graph DataGraph { get; set; }`

C.2 TreePlus API (Application Programming Interface)

- ```
  public void Clear();
  ```

- ```
  public void Home();
  ```

- ```
  public ArrayList GetRootNodes();
  ```

- ```
  public ArrayList GetFocusedNodes();
  ```

- ```
  public void OpenNodesWithinDistance(
     int distance,
     bool outgoing
  );
  ```

- ```
  public void ResetSearch();
  ```

- ```
  public void Search(
     Hashtable keywords,
     bool outgoing,
     bool incoming
  );
  ```

- ```
  public void SelectNode(
     string nodeId
  );
  ```

- ```
  public void ShowGraph(
     string rootNodeId
  );
  ```

- ```
  public void ShowGraph(
     ArrayList rootNodeIds
  ```

```
);
```

- ```
  public void FitinWinodw();
  ```
- ```
  public void Relayout();
  ```
- ```
  public void UpdatePreferences(
     Preferences pref,
     bool redraw
  );
  ```
- ```
  public void Zoom (
     int zoomFactor
  );
  ```

C.3 Public Properties for the Preferences Class

- ```
  public bool ShowLegend { get; set; }
  ```
- ```
  public string FocusDesc { get; set; }
  ```
- ```
  public string OutDesc { get; set; }
  ```
- ```
  public string InDesc { get; set; }
  ```
- ```
  public ArrayList VisibleAttrs { get; set; }
  ```
- ```
  public string Category { get; set; }
  ```
- ```
  public int ConBarWidth { get; set; }
  ```
- ```
  public int MaxWidth { get; set; }
  ```
- ```
  public string CrossLinks { get; set; }
  ```
- ```
  public ArrayList RootNodes { get; set; }
  ```
- ```
  public string DrawLinks { get; set; }
  ```
- ```
  public bool AdjustLayout { get; set; }
  ```
- ```
  public string CurrentRoot { get; set; }
  ```
- ```
  public string DefaultRoot { get; set; }
  ```
- ```
  public string SortOrder { get; set; }
  ```
- ```
  public string OrderBy { get; set; }
  ```
- ```
  public bool UseArrow { get; set; }
  ```
- ```
  public string UseColor { get; set; }
  ```
- ```
  public bool UseCurve { get; set; }
  ```
- ```
  public int SiblingSeparation { get; set; }
  ```
- ```
  public int SubtreeSeparation { get; set; }
  ```
- ```
  public int LevelSeparation { get; set; }
  ```
- ```
  public int BorderWidth { get; set; }
  ```
- ```
  public int CountBarThickness { get; set; }
  ```
- ```
  public int TickSize { get; set; }
  ```
- ```
  public long LayoutAnimationDuration { get; set; }
  ```

Bibliography

1. J. Abello and J. Korn, MGV: A System for Visualizing Massive Multigraphs, *IEEE Trans. Visualization and Computer Graphics*, Vol. 8, No, 1, pp. 21-38, 2002.

2. ACM Digital Library, http://portal.acm.org

3. E. Adar, GUESS: A Language and Interface for Graph Exploration, To appear in *Proceedings of the Conference on Human Factors in Computing Systems (CHI '06)*, 2006.

4. R. Amar, J. Eagan, and J. Stasko, Low-Level Components of Analytic Activity in Information Visualization, *Proceedings of the Symposium on Information Visualization (InfoVis '05)*, pp. 111-117, 2005.

5. AmiGO, http://www.godatabase.org/cgi-bin/go.cgi

6. An Atlas of Cyberspaces, http://www.cybergeography.org/atlas

7. K. Andrews and H. Heidegger, Information Slices: Visualising and Exploring Large Hierarchies using Cascading, Semicircular Disks, *Proceedings of the Symposium on Information Visualization (InfoVis '98)*, pp. 9-12. 1998.

8. D. Auber, Tulip : A Huge Graph Visualization Framework, Graph Drawing Softwares, Mathematics and Visualization, Berlin: Springer-Verlag, pp. 105-126, 2003.

9. E. H. Baehrecke, N. Dang, K. Babaria, and B. Shneiderman, Visualization and Analysis of Microarray and Gene Ontology Data with Treemaps, *BMC Bioinformatics*, Vol. 5, Available at http://www.biomedcentral.com/1471-2105/5/84, June 2004.

10. M. Q. W. Baldonado, A. Woodruff, and A. Kuchinsky, Guidelines for Using Multiple Views in Information Visualization, *Proceedings of the working conference on Advanced Visual Interfaces (AVI '00)*, pp. 110-119, 2000.

11. J. Barnes and P. Hut, A Hierarchical O(N log N) Force-Calculation Algorithm, *Nature*, Vol. 324, pp. 446-449, December 1986.

12. C. Batini, L. Furlani, and E. Nardelli, What is a Good Diagram? A Pragmatic Approach, *Proceedings of the International Conference on Entity-Relationship Approach*, pp. 312-319, 1985.

13. P. Baudisch, N. Good, and P. Stewart, Focus Plus Context Screens: Combining Display Technology with Visualization Techniques, *Proceedings of the Symposium on User Interface Software and Technology (UIST '01)*, pp. 31-40, 2001.

14. B. B. Bederson, J. Grosjean, and J. Meyer, Toolkit Design for Interactive Structured Graphics, *IEEE Transactions on Software Engineering*, Vol. 30, No. 8, pp. 535-546, August 2004.

15. B. B. Bederson, J. D. Hollan, K. Perlin, J. Meyer, D. Bacon, and G. Furnas, PAD++: A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics, *Journal of Visual Languages and Computing*, Vol. 7, No. 1, pp. 3-31, 1996.

16. F. Bertault, A Force–Directed Algorithm that Preserves Edge Crossing Properties, *Proceedings of the Symposium on Graph Drawing (GD'99)*, Lecture Notes in Computer Science 1731, pp. 351–358, Springer–Verlag, 1999

17. J. Blythe, C. McGrath, and D. Krackhardt, The Effect of Graph Layout on Inference from Social Network Data, *Proceedings of the Symposium on Graph Drawing (GD '95)*, Lecture Notes in Computer Science 1027, pp. 40-51, Springer-Verlag, 1995.

18. R. A. Botafogo, E. Rivlin, and B. Shneiderman, Structural Analysis of Hypertexts: Identifying Hierarchies and useful Metrics, *ACM Transactions on Information Systems*, Vol. 10, No. 2, pp. 142-180, April 1992.

19. F. Boutin, J. Thièvre, and M. Hascoët, Multilevel Compound Tree – Construction Visualization and Interaction, *Proceedings of the International Conference on Human-Computer Interaction (INTERACT '05)*, Lecture Notes in Computer Science 3583, pp. 847-860, Springer-Verlag, 2005.

20. F. Boutin, J. Thièvre, and M. Hascoët, Focus-based Filtering + Clustering Technique for Power Law Networks with Small World Phenomenon, *Proceedings of the Conference on Visual Data Analysis (VDA '06)*, Vol. 6060, 2006.

21. K. Börner and C. Chen, Visual Interfaces to Digital Libraries: Motivation, Utilization, and Socio-Technical Challenges. *Visual Interfaces to Digital Libraries [JCDL 2002 Workshop]*, pp. 1-12, Springer-Verlag, 2002.

22. F. J. Brandenburg, Nice Drawing of Graphs are Computationally Hard, *Visualization in Human-Computer Interaction*, Lecture Notes in Computer Science 439, pp. 1-15, Springer-Verlag, 1988.

23. B.-J. Breitkreutz, C. Stark, and M. Tyers, Osprey: a Network Visualization System, Genome Biology, Vol. 4, No. 3, Available at http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&list_uids=12620107, February 2003.

24. I. Bruß and A. Frick, Fast Interactive 3-D Graph Visualization, *Proceedings of the Symposium on Graph Drawing (GD '95)*, Lecture Notes in Computer Science 1027, pp. 99-110, Springer-Verlag, 1995.

25. F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-oriented Software Architecture, Volume 1, A System of Patterns*. Chichester; New York: Wiley, 1996.

26. S. K. Card, J. D. MacKinlay, and B. Shneiderman, *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufmann Publishers, California: San Francisco, 1999.

27. M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia, Three-Dimensional Pliable Surfaces: For Effective Presentation of Visual Information, *Proceedings of the Symposium on User Interface Software and Technology (UIST '95)*, pp. 217-226, 1995.

28. J. Carriere and R. Kazman, Research Report: Interacting with Huge Hierarchies: Beyond Cone Trees, *Proceedings of the Symposium on Information Visualization (InfoVis '95)*, pp. 74-81, 1995.

29. C. Chen and L. Carr, Trailblazing the Literature of Hypertext: Author Co-Citation Analysis, *Proceedings of the 10th ACM Conference on Hypertext and hypermedia (Hypertext '99)*, pp. 51-60, 1999.

30. E. H. Chi, J. Pitkow, J. Mackinlay, P. Pirolli, R. Grossweiler, and S. K. Card, Visualizing the Evolution of Web Ecologies, *Proceedings of the Conference on Human Factors in Computing Systems (CHI '98)*, pp. 400-407, 1998.

31. A. Cockburn and B. McKenzie, An Evaluation of Cone Trees, *People and Computers XIV: Proceedings of the British Computer Society Conference on Human Computer Interaction*, 2000.

32. T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, McGraw-Hill, 1993.

33. R. Davison and D. Harel, Drawing Graphs Nicely using Simulated Annealing, *ACM Transactions on Graphics*, Vol. 15, No. 4, pp. 301-331, 1996.

34. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, Algorithms for Drawing Graphs: an Annotated Bibliography, *Computational Geometry: Theory and Applications*, Vol. 4, No. 5, pp. 235-282, 1994.

35. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice Hall, New Jersey: Englewood Cliffs, 1999.

36. DockPanel Suite, http://sourceforge.net/projects/dockpanelsuite/

37. U. Doğrusöz, B. Madden, and P. Madden, Circular Layout in the Graph Layout Toolkit, *Proceedings of the Symposium on Graph Drawing (GD '96)*, Lecture Notes in Computer Science 1190, pp. 92-100, Springer-Verlag, 1996.

38. P. Dömel, Webmap: A Graphical Hypertext Navigation Tool, *Proceedings of the Second International World-Wide Web Conference (WWW '94)*, 1994.

39. A. Druin, Cooperative Inquiry: Developing New Technologies For Children With Children, *Proceedings of the Conference on Human Factors in Computing Systems (CHI '99)*, pp. 223-230, 1999.

40. P. Eades, A Heuristic for Graph Drawing, *Congressus Numerantium*, Vol. 42, pp. 149-160, 1984.

41. P. Eades, Drawing free trees, *Bulletin of the Institute of Combinatorics and its Applications*, Vol. 5, pp. 10-36, 1992.

42. P. Eades, M. E. Houle, and R. Webber, Finding the Best Viewpoints for Three-Dimensional Graph Drawings, *Proceedings of the Symposium on Graphing Drawing (GD '97)*, Lecture Notes in Computer Science 1353, pp. 87-98, Springer-Verlag, 1997.

43. P. Eades and X. Lin, Spring Algorithms and Symmetry, *Theoretical Computer Science*, Vol. 240, No. 2, pp. 379-405, 2000.

44. P. Eades and K. Sugiyama, How to Draw a Directed Graph, *Journal of Information Processing*, Vol. 13, No. 4, pp. 424-434, 1990.

45. P. W. Eklund, N. Roberts, and S. P. Green, OntoRama: Browsing an RDF Ontology using a Hyperbolic-like Browser, *The First International Symposium on CyberWorlds (CW '02)*, pp.405-411, 2002.

46. K. Fairchild, S. Poltrok, and G. Furnas, SemNet: Three-Dimensional Graphic Representations of Large Knowledge Bases, *Cognitive Science and its Applications for Human-Computer Interaction*, Lawrence Erlbaum Associates, New Jersey: Hillsdale, pp. 201-233. 1988.

47. J.-D. Fekete, D. Wang, N. Dang, A. Aris, and C. Plaisant, Overlaying Graph Links on Treemaps, *Posters Compendium of the Symposium on Information Visualization (InfoVis '03)*, pp. 82-83, 2003.

48. A. Formella and J. Keller, Generalized Fisheye Views of Graphs, *Proceedings of the Symposium on Graph Drawing* (GD '95), Lecture Notes in Computer Science 1027, pp. 242-253, Springer-Verlag, 1995.

49. A. Frick, A. Ludwig, and H. Mehldau, A Fast Adaptive Layout Algorithm for Undirected Graphs, *Proceedings of the Symposium on Graph Drawing (GD '94)*, Lecture Notes in Computer Science 894, pp. 388-403, Springer-Verlag, 1994.

50. M. Fröhlich and M. Werner, Demonstration of the Interactive Graph Visualization System da Vinci, *Proceedings of the Symposium on Graph Drawing (GD '94)*, Lecture Notes in Computer Science 894, pp. 266-269, Springer-Verlag, 1994.

51. T. M. J. Fruchterman and E. Reingold, Graph Drawing by Force-Directed Placement, *Software-Practice and Experience*, Vol. 21, No. 11, pp. 1129-1164, November 1991.

52. G. W. Furnas, Generalized Fisheye Views, *Proceedings of the Conference on Human Factors in Computing Systems (CHI '86)*, pp. 18-23, 1986.

53. G. W. Furnas and J. Zacks, Multitrees: Enriching and Reusing Hierarchical Structure, *Proceedings of the Conference on Human Factors in Computing Systems (CHI '94)*, pp. 330-336, 1994.

54. S. Gamard, M. J. Schoelles, C. Kofila, V. D. Veksler, and W. D. Gray, CogWorks Visualization Architecture: Cognitively Engineering Next Generation Workstations for Decision Makers, *ACT-R Work-shop*, 2005.

55. E. R. Gansner, E. Koutsofois, S. C. North, and K.-P. Vo, A Technique for Drawing Directed Graphs, *IEEE Transactions on Software Engineering*, Vol. 19, No. 3, pp. 214-229, March 1993.

56. E. R. Gansner and S. C. North, Improved Force-Directed Layouts, *Proceedings of the Symposium on Graph Drawing (GD '98)*, Lecture Notes in Computer Science 1547, pp. 364-373, Springer-Verlag, 1998.

57. M. R. Garey and D. S. Johnson, Crossing Number is NP-complete, *SIAM Journal on Algebraic and Discrete Methods*, Vol. 4, No. 3, pp. 312-316, 1983.

58. A. Garg and R. Tamassia, GIOTTO3D: A System for Visualizing Hierarchical Structures in 3D, *Proceedings of the Symposium on Graph Drawing (GD '96)*, Lecture Notes in Computer Science 1190, pp. 193-200, Springer-Verlag, 1996.

59. Gene Ontology Consortium, http://www.geneontology.org

60. M. Ghoniem, J.-D. Fekete, and P. Castagliola, A Comparison of the Readability of Graphs using Node-Link and Matrix-Based Representations, *Proceedings of the Symposium on Information Visualization (InfoVis '04)*, pp. 17-24, 2004.

61. M. Ghoniem, N. Jussien, and J.-D. Fekete, VISEXP: Visualizing Constraint Solver Dynamics using Explanations, *Proceedings of the Seventh International Florida Artificial Intelligence Research Society Conference (FLAIRS '04)*, 2004.

62. GraphML File Format, http://graphml.graphdrawing.org/

63. M. C. Hao, M. Hsu, U. Dayal, and A. Krug, Web-Based Visualization of Large Hierarchical Graphs using Invisible Links in a Hyperbolic Space, *Proceedings of the Fifth Working Conference on Visual Database Systems (VDB5)*, pp. 83-94, 2000.

64. D. Harel and Y. Koren, Graph Drawing by High-Dimensional Embedding, *Proceedings of the Symposium on Graph Drawing (GD '02)*, Lecture Notes in Computer Science 2528, pp. 207-219, Spring-Verlag, 2002.

65. J. Heer and D. Boyd, Vizster: Visualizing Online Social Networks, *Proceedings of the Symposium on Information Visualization (InfoVis '05)*, pp. 33-40, 2005.

66. I. Herman, G. Melançon, and M. S. Marshall, Graph Visualization and Navigation in Information Visualization: a Survey, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 6, No. 1, pp. 24-43, 2000.

67. I. Herman, G. Melançon, M. M. de Ruiter, and M. Delest, Latour – A Tree Visualisation System, *Proceedings of the Symposium on Graph Drawing (GD '99)*, Lecture Notes in Computer Science 1731, pp. 392-399, Springer-Verlag, 1999.

68. C. P. Hickman, L. S. Roberts, and A. Larson, *Animal Diversity*, 3rd Edition, McGraw-Hill, 2003.

69. K. Hornbæk and E. Frøkjær, Reading of Electronic Documents: the Usability of Linear, Fisheye, and Overview + Detail Interfaces, *Proceedings of the Conference on Human Factors in Computing Systems (CHI '01)*, pp. 293-300, 2001.

70. B. Huffaker, E. Nemeth, and K. Claffy, Otter: A General Purpose Network Visualization Tool, Available at http://www.caida.org/tools/visualization/otter/paper/, 1999.

71. IEEE Xplore, http://ieeexplore.ieee.org

72. T. Igarashi and K. Hinckley, Speed-dependent automatic zooming for browsing large documents, *Proceedings of the Symposium on User Interface Software and Technology (UIST '00)*, pp. 139-148, 2000.

73. Information Visualization Benchmarks Repository, http://www.cs.umd.edu/hcil/InfovisRepository/benchmark.shtml

74. InfoVis 2003 Contest: Visualization and Pair Wise Comparison of Trees, http://www.cs.umd.edu/hcil/iv03contest

75. InfoVis 2004 Contest: The History of InfoVis, http://www.cs.umd.edu/hcil/iv04contest

76. Integrated Taxonomic Information System (ITIS), http://www.itis.usda.gov, 2003.

77. T. J. Jankun-Kelly and K.-L. Ma, MoireGraphs: Radial Focus+Context Visualization and Interaction for Graphs with Visual Nodes, *Proceedings of the Symposium on Information Visualization (InfoVis '03)*, pp. 59-66, 2003.

78. B. Johnson and B. Shneiderman, Tree–maps: A Space–Filling Approach to the Visualization of Hierarchical Information Structures, *Proceedings of the IEEE Conference on Visualization (Visualization '91)*, pp. 275-282, 1991.

79. JUNG: Java Universal Network Graph Framework, http://jung.sourceforge.net

80. T. Kamada and S. Kawai, An Algorithm for Drawing General Undirected Graphs, *Information Processing Letters*, Vol. 31, No. 1, pp. 7-15, 1989.

81. H. Kang, C. Plaisant, B. Lee, and B. B. Bederson, Exploring Content-Actor Paired Network Data using Iterative Query Refinement with NetLens, To appear in *Proceedings of the Joint Conference on Digital Libraries* (*JCDL '06)*, demonstration, 2006.

82. K. Kaugars, J. Reinfelds, and A. Brazma, A Simple Algorithm for Drawing Large Graphs on Small Screens, *Proceedings of the Symposium on Graph Drawing (GD '94)*, Lecture Notes in Computer Science 894, pp. 278-281, Springer-Verlag, 1995.

83. T. A. Keahey and E.L. Robertson, Techniques for Non–Linear Magnification Transformations, *Proceedings of the Symposium on Information Visualization (InfoVis '97)*, pp. 38-45, 1997.

84. R. Kincaid, VistaClara: An Interactive Visualization for Exploratory Analysis of DNA Microarrays, *Proceedings of the Symposium on Applied Computing (SAC '04)*, pp. 167-174, 2004.

85. C. Klein and B. Bederson, Benefits of Animated Scrolling, *Extended Abstracts of Human Factors in Computing Systems (CHI '05)*, pp. 1965-1968, 2005.

86. J. Lamping and R. Rao, The Hyperbolic Browser: A Focus + Context Technique for Visualizing Large Hierarchies, *Journal of Visual Languages and Computing*, Vol. 7, No. 1, pp. 33-55, 1996.

87. B. Lee, M. Czerwinski, G. Robertson, and B. B. Bederson, Understand Research Trends in Conferences using PaperLens, *Extended Abstracts of Human Factors in Computing Systems (CHI '05)*, pp. 1969-1972, 2005.

88. B. Lee, C. S. Parr, D. Campbell, and B. B. Bederson, How Users Interact with Biodiversity Information Using TaxonTree, *Proceedings of the working conference on Advanced Visual Interfaces (AVI '04)*, pp. 134-140, 2004.

89. B. Lee, C. S. Parr, C. Plaisant, and B. B. Bederson, Visualizing Graphs as Trees: Plant a Seed and Watch It Grow, *Proceedings of the Symposium on Graph Drawing (GD '05)*, Lecture Notes in Computer Science 3843, pp. 516-518, Springer-Verlag, January 2006.

90. Y. K. Leung and M. D. Apperley, A Review and Taxonomy of Distortion-Oriented Presentation Techniques, *ACM Transactions on Computer-Human Interaction*, Vol. 1, No. 2, pp. 126-160, 1994.

91. J. D. Mackinlay, R. Rao, and S. K. Card, An Organic User Interface for Searching Citation Links, *Proceedings of the the Conference on Human Factors in Computing Systems (CHI '95)*, pp. 67-73, 1995.

92. J. D. Mackinlay, G. G. Robertson, and S. K. Card, The Perspective Wall: Detail and Context Smoothly Integrated, *Proceedings of the Conference on Human Factors in Computing Systems (CHI '91)*, pp. 173-179, 1991.

93. R. Maddison (ed.), Tree of Life, http://www.tolweb.org, 2003.

94. M. J. McGuffin and R. Balakrishnan, Interactive Visualization of Genealogical Graphs, *Proceedings of the Symposium on Information Visualization (InfoVis '05)*, pp. 17-24, 2005.

95. G. J. McLachlan and K. E. Basford, *Mixture Models: Inference and Applications to Clustering*, Marcel Dekker, New York, 1988.

96. S. Moen, Drawing Dynamic Trees, *IEEE Software*, Vol. 7, No. 4, pp. 21-28, July 1990.

97. P. M. Mullins and S. Treu, A Task-Based Cognitive Model for User-Network Interaction: Defining a Task Taxonomy to Guide the Interface Designer, *Interacting with Computers*, Vol. 5, No. 2, pp. 139–166, 1993.

98. T. Munzner, H3: Laying Out Large Directed Graphs in 3D Hyperbolic Space, *Proceedings of the Symposium on Information Visualization (InfoVis '97)*, pp. 2-10, 1997.

99. T. Munzner, Exploring Large Graphs in 3D Hyperbolic Space, *IEEE Computer Graphics and Applications*, Vol. 18, No. 4, pp. 18-23, July/August 1998.

100. T. Munzner, Drawing Large Graphs with H3Viewer and Site Manager, *Proceedings of the Symposium on Graph Drawing (GD '98)*, Lecture Notes in Computer Science 1547, pp. 384-393, Springer-Verlag, 1998.

101. M. A. Nascimento, J. Sander, and J. Pound, Analysis of SIGMOD's CoAuthorship Graph, *SIGMOD Record*, Vol. 32, No. 3, September 2003.

102. L. Nigay and F. Vernier, Design Method of Interaction Techniques for Large Information Spaces, *Proceedings of the working conference on Advanced Visual Interfaces (AVI '98)*, pp. 37-46, 1998.

103. L. T. Nowell, R. K. France, and D. Hix, Exploring Search Results with Envision, *Extended Abstracts of Human Factors in Computing Systems (CHI '97)*, pp. 14-15, 1997.

104. ParaCite, http://paracite.eprints.org

105. C. S. Parr, B. Lee, D. Campbell, and B. B. Bederson, Tree Visualizations for Taxonomies and Phylogenies, *Bioinformatics*, Vol. 20, No. 17, pp. 2997-3004, 2004.

106. Piccolo.NET, http://www.cs.umd.edu/hcil/piccolo

107. C. Plaisant, The Challenge of Information Visualization Evaluation, *Proceedings of the working conference on Advanced Visual Interfaces (AVI '04)*, pp. 109-116, 2004.

108. C. Plaisant, D. Carr, and B. Shneiderman, Image-Browser Taxonomy and Guidelines for Designers, *IEEE Software*, Vol. 12, No. 2, pp. 21-32, March 1995.

109. C. Plaisant, J. Grosjean, and B. B. Bederson, SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation, *Proceedings of the Symposium on Information Visualization (InfoVis '02)*, pp. 57-64, 2002.

110. H. C. Purchase, Which Aesthetic Has the Greatest Effect on Human Understanding?, *Proceedings of the Symposium on Graph Drawing (GD '97)*, Lecture Notes in Computer Science 1353, pp. 248-261, Springer-Verlag, 1997.

111. H. C. Purchase, R. F. Cohen, and M. James, An Experimental Study of the Basis for Graph Drawing Algorithms, *Journal of Experimental Algorithms*, Vol. 2, No. 4, 1997.

112. H. C. Purchase, R. F. Cohen, and M. James, Validating Graph Drawing Aesthetics, *Proceedings of the Symposium on Graph Drawing (GD '95)*, Lecture Notes in Computer Science 1027, pp. 435-446, Springer-Verlag, 1995.

113. D. P. Reagan and R. B. Waide, *The Food Web of a Tropical Rain Forest*, University of Chicago Press, Illinois: Chicago, 1996.

114. M. Reingold and J. S. Tilford, Tidier Drawing of Trees, *IEEE Transactions on Software Engineering*, Vol. 7, No. 2, pp. 223-228, 1981.

115. J. Rekimoto and M. Green, The Information Cube: Using Transparency in 3D Information Visualization, *Proceedings of the Third Annual Workshop on Information Technologies and Systems (WITS '93)*, pp. 125-132, 1993.

116. K. Risden, M. Czerwinski, T. Munzner, and D. B. Cook, An Initial Examination of Ease of Use for 2D and 3D Information Visualizations of Web Content, *Internal Journal of Human Computer Studies*, Vol. 53, No. 5, pp. 695-714, 2000.

117. G. Robertson, K. Cameron, M. Czerwinski, and D. Robbins, Polyarchy Visualization: Visualizing Multiple Intersecting Hierarchies, *Proceedings of the Conference on Human Factors in Computing Systems (CHI '02)*, pp. 423-430, 2002.

118. G. Robertson, J. D. Mackinlay, and S. K. Card, Cone Trees: Animated 3D Visualizations of Hierarchical Information, *Proceedings of the Conference on Human Factors in Computing Systems (CHI '91)*, pp. 189-194, 1991.

119. M. Sarkar and M. H. Brown, Graphical Fisheye Views of Graphs, *Proceedings of the Conference on Human Factors in Computing Systems (CHI '92)*, pp. 83-91, 1992.

120. M. Sarkar and M. H. Brown, Graphical Fisheye Views, *Communications of the ACM*, Vol. 37, No. 12, pp. 73-84, December 1994.

121. Scientific Literature Digital Library, http://citeseer.ist.psu.edu/cs

122. Y. Shiloach, *Arrangements of planar graphs on the planar lattice*. Ph.D. Thesis, Department of Applied Mathematics, Weizmann Institute of Science, Israel, 1976.

123. B. Shneiderman, The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations, *Proceedings of the Symposium on Visual Languages (VL '96)*, pp. 336-343, 1996.

124. B. Shneiderman, D. Feldman, A. Rose, and X. F. Grau, Visualizing Digital Library Search Results with Categorical and Hierarchical Axes, *Proceedings of the fifth ACM conference on Digital libraries*, pp. 57-66, 2000.

125. A. F. Smeaton, G. Keogh, C. Gurrin, K. McDonald, and T. Sødring, Analysis of Papers from Twenty-Five Years of SIGIR Conferences: What Have We Been Doing for the Last Quarter of a Century?, *ACM SIGIR Forum*, pp. 49-53, 2003.

126. SPIRE: Applying Semantic Web Technologies in Research EcoInformatics, http://spire.umbc.edu/

127. K. Sugiyama, S. Tagawa, and M. Toda, Methods for Visual Understanding of Hierarchical Systems, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 11, No. 2, pp. 109-125, February 1981.

128. K. J. Supowit and E. M. Reingold, The Complexity of Drawing Trees Nicely, *Acta Information*, Vol. 18, pp. 377-392, 1982.

129. TouchGraph, http://www.touchgraph.com

130. P. Uetz (ed.), European Molecular Biology Laboratory Reptile Database, http://www.reptile-database.org, 2003.

131. University of Michigan Museum of Zoology, http://www.ummz.umich.edu/birds, 2003.

132. F. van Ham, Using Multilevel Call Matrices in Large Software Projects, *Proceedings of the Symposium on Information Visualization (InfoVis '03)*, pp. 227-232, 2003

133. J. J. van Wijk and H. van de Wetering, Cushion Treemaps: Visualization of Hierarchical Information, *Proceedings of the Symposium on Information Visualization (InfoVis '99)*, pp. 73-78, 1999.

134. Visual Complexity, http://www.visualcomplexity.com

135. R. Vdovjak, P. Barna, and G.-J. Houben, EROS: Explorer for RDFS-based Ontologies, *Proceedings of the International Conference on Intelligent User Interfaces (IUI '03)*, pp. 330-330, Demo paper, 2003.

136. J. Q. Walker II, A Node-Positioning Algorithm for General Trees, *Software – Practice and Experience*, Vol. 20, No. 7, pp. 685-705, 1990.

137. C. Ware and G. Franck, Evaluating Stereo and Motion Cues for Visualizing Information Nets in Three Dimensions, *ACM Transactions on Graphics*, Vol. 15, No. 2, pp. 121-140, 1996.

138. C. Wetherell and A. Shannon, Tidy Drawing of Trees, *IEEE Transactions on Software Engineering*, Vol. 5, No.5, pp. 514-520, 1979.

139. G. J. Wills, NicheWorks – Interactive Visualization of Very Large Graphs, *Proceedings of the Symposium on Graph Drawing (GD '97)*, Lecture Notes in Computer Science 1353, pp. 403-414, Springer-Verlag, 1997.

140. G. J. Wills, NicheWorks – Interactive Visualization of Very Large Graphs, *Journal of Computational and Graphical Statistics*, Vol. 8, No. 3, pp. 190-212, June 1999.

141. U. Wiss, D. A. Carr, and H. Jonsson, Evaluating Three-Dimensional Information Visualization Designs: A Case Study of Three Designs, *Proceedings of the International Conference on Information Visualization (IV '98)*, pp.137-144, 1998.

142. P. C. Wong, B. Hetzler, C. Posse, M. Whiting, S. Havre, N. Cramer, A. Shah, M. Singhal, A. Turner, and J. Thomas, IN-SPIRE InfoVis 2004 Contest Entry, *Posters Compendium of the Symposium on Information Visualization (InfoVis '04)*, 2004.

143. K.-P. Yee, D. Fisher, R. Dhamija, and M. Hearst, Animated Exploration of Dynamic Graphs with Radial Layout, *Proceedings of the Symposium on Information Visualization (InfoVis '01)*, pp. 43-50, 2001.

144. J. Ziegler, C. Kunz, and V. Botsch, Matrix Browser – Visualizing and Exploring Large Networked Information Spaces, *Extended Abstracts of Human Factors in Computing Systems (CHI '02)*, pp. 602-603, 2002.