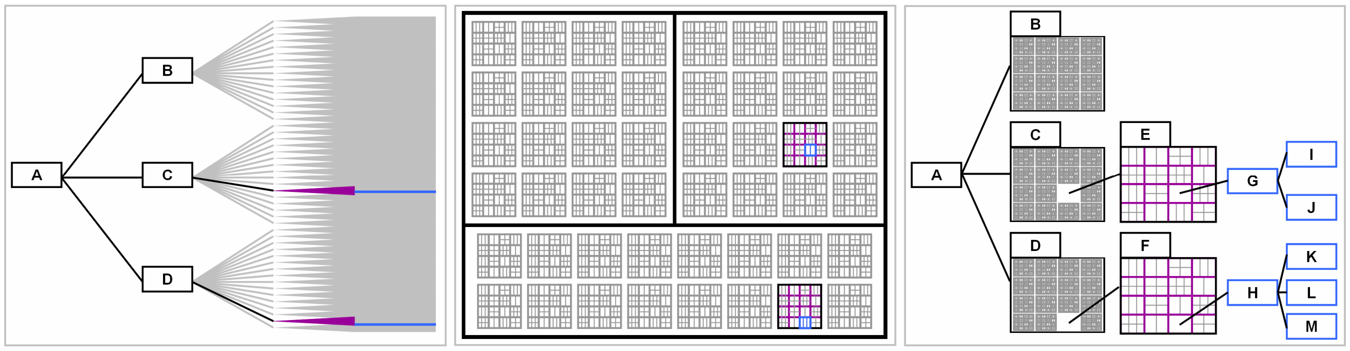
Elastic Hierarchies: Combining Treemaps and Node-Link Diagrams

Shengdong Zhao[1] Michael J. McGuffin[2] Mark H. Chignell[3]

University of Toronto University of Toronto University of Toronto



Node–Link Diagram **Treemap** **Elastic Hierarchy**

Figure 1: an illustration of the same tree drawn in three styles, with certain branches highlighted. Node-Link diagrams show topology clearly, but distribute nodes unevenly, leaving upper level nodes separated by white space, and lower nodes densely packed. Treemaps use space efficiently, but are less familiar and can be difficult to interpret. Elastic Hierarchies combine the two techniques, allowing chosen structures and content to be emphasized and clearly presented in a flexible and space-efficient manner.

Abstract

We investigate the use of *elastic hierarchies* for representing trees, where a single graphical depiction uses a hybrid mixture, or "interleaving", of more basic forms at different nodes of the tree. In particular, we explore combinations of node-link and Treemap forms, to combine the space-efficiency of Treemaps with the structural clarity of node-link diagrams. A taxonomy is developed to characterize the design space of such hybrid combinations. A software prototype is described, which we used to explore various techniques for visualizing, browsing and interacting with elastic hierarchies, such as side-by-side overview and detail views, highlighting and rubber banding across views, visualization of multiple foci, and smooth animations across transitions. The paper concludes with a discussion of the characteristics of elastic hierarchies and suggestions for research on their properties and uses.

CR Categories and Subject Descriptors: I.3.6 [Computer Graphics]: Methodology and Techniques–interaction techniques; E.1 [Data Structures]: trees

Additional Keywords: Elastic Hierarchies, Treemaps, node-link diagrams, hybrids, combinations, overview+detail, multiple views, trees, interaction techniques, interactive visualization

1 Introduction

Trees are a fundamental organizing structure, with menu hierarchies and file directories being prominent examples of their use. The data size of a tree typically grows exponentially with its depth, which raises many challenges for visualization. Showing the structure is space consuming, and the exponential growth in the number of nodes from the root to the leaves creates difficulties for laying out the items of large trees effectively in a given space.

Many tree representations have been proposed in the past. Various styles have unique visual and interactive properties that may be useful in different scenarios, but they also have limiting constraints, creating tradeoffs in their use. For example, the classical node-link diagram [15] is probably the most natural way to display nesting structure, but fails to scale to large datasets. In contrast, Treemaps [17] are space efficient, scaling up to thousands of nodes, but at the cost of making the different levels within the tree harder to perceive and distinguish.

Trees are often complex and can have very different local properties across nodes. In addition, trees are often dynamic, making a single style of representation harder to adjust to variations over time. In this paper we explore the concept of allowing different styles of representation at different places in a tree. The resulting hybrid may allow designers to combine the best features of different representations, enabling a user to view each part of the data in the most effective way. However, hybrids may carry the disadvantage of being less uniform and less familiar to users, making it all the more important to use good visual design. Our research investigates the properties and affordances of such mixed-representation trees, or *elastic hierarchies*, as a first step toward determining when and how to use hybrid tree representations. "Elastic" refers to the flexibility allowed by arbitrarily interleaving representations (right image, Figure 1, Figure 4). I.e., we allow the representation portraying nesting at each point to be chosen independently of the representation choices made at other points in the tree.

Elastic hierarchies, as described below, are a means of exploring the large design space of 2D hierarchical visualization. As multi-representational views where the representational form of each subtree can be modified on the fly by the user, they allow

[1]sszhao@dgp.toronto.edu
[2]mjmcguff@cs.toronto.edu
[3]chignell@mie.toronto.edu

researchers to explore how many representations can usefully be shown within a tree, and what forms of transition should be used between different types of representation. Studies of how people use elastic hierarchies can help researchers determine what tools and interaction techniques should be provided to assist users in transitioning from one representation to another in viewing a large hierarchy.

In this paper, we describe the design space of elastic hierarchies made up of interleaved Treemaps and node-link diagrams, and discuss ways in which such hybrids may be more suitable than traditional visualizations. We describe a prototype that implements our more promising design ideas, including support for visualizing multiple foci. We also identify a set of research questions concerning elastic hierarchies to be addressed in future studies.

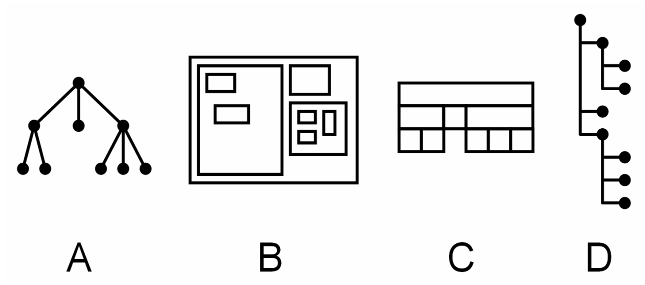


Figure 2: common tree representations, each showing the same tree in a different way. **A:** node-link. **B:** nested containment, or nested enclosure. **C:** use of alignment and adjacency. **D:** indented outline style. (Note that **D** is not simply a variation on **A**. In **D**, the edges are implied by the positioning of the nodes, which is not generally the case in **A**.)

2 Background

The most common representational form for a tree is the node-link diagram [15]. While node-link diagrams show nesting structure very clearly, they use screen space inefficiently, and do not scale well to large datasets. As a result many approaches have been proposed to supplement node-link diagrams. Well known alternatives include Treemaps [17], cone trees [16], and the hyperbolic browser [11].

Since many visualization methods have been proposed, it is useful to organize these into categories. Besides node-link diagrams (Figure 2 **A**), another major style involves nested containment or nested enclosure (Figure 2 **B**). Two other forms use alignment and adjacency (Figure 2 **C**), and the indented outline style (Figure 2 **D**). Most other proposals can be viewed as extensions or combinations of these. For example, the hyperbolic browser [11] arranges a node-link diagram (**A**) radially, with a focus point that can be manipulated interactively by the user. Cone Trees [16] extend **A** into 3D. Treemaps [17] are an example of **B** where the area of nodes encodes additional information associated with them. Information slices [2] and Sunburst [19] are variations of **C** using polar coordinates.

Although hybrid tree forms have not been systematically explored in depth, various hybrid techniques have been used to visualize trees and graphs. Fekete et al. [7] have described a graphical representation for graphs that uses both a Treemap, to represent a spanning tree over the graph, and curved links, to represent the remaining edges. In addition, Harel [8] and Sindre et al. [18] have described representations for graphs that are richer than simple node-link diagrams, and that can encode additional information by using various graphical conventions and symbols. In their work, different types of relationships between nodes are shown simultaneously using different conventions, such as connection (Figure 2 **A**) and containment (Figure 2 **B**). By contrast, in elastic hierarchies, only one type of relationship needs to be shown, that between parent and children nodes, but we allow multiple conventions to be used for this, e.g., connection and containment, creating a range of choices in the graphical layout, and allowing the user to pick the one desired.

Various hybrid representations of trees have been implemented in 3D. Information pyramids [1] combine nested containment (Figure 2 **B**) in 2D with adjacency (Figure 2 **C**) along a 3rd dimension, by stacking nodes in a layered fashion. Balzer and Deussen [3] present a visualization which uses two styles: nested enclosure and linked nodes are shown simultaneously to represent the same tree. In these hybrids, the combination of representations is fixed, and cannot be independently changed at different locations of the tree.

Given all these variations in tree representations, and research that identifies their pros and cons, a single representation style, or even a fixed hybrid form, may not accommodate the needs of complex and dynamic real world problems. Our work investigates improving tree representations using dynamically adjustable hybrids, i.e. elastic hierarchies, and focuses on the case of combining Treemaps and node-link diagrams in a single display.

3 Design Space

In theory, hybrids between any of the forms in Figure 2 might be feasible; however, we note that nested containment (Figure 2 **B**) differs from the 3 others in that child nodes are *inside* parent nodes, whereas they are outside in the other forms. Furthermore, node-link diagrams (Figure 2 **A**) and nested containment are perhaps the most contrasting forms. Node-link diagrams show topological structure well, while Treemaps are space efficient. Treemaps tend to allot more screen space to shallow nodes, making them easier to see, independently of how deep the subtree under a node may be. Treemaps also allow for visual comparison of the relative sizes of nodes. Node-link diagrams, however, are more familiar to users, and perhaps better at showing the different levels and depth of a tree.

Due to their complementary properties, Treemaps and node-link diagrams are especially suited for hybrid combinations that access a continuum of intermediate forms. These two styles can

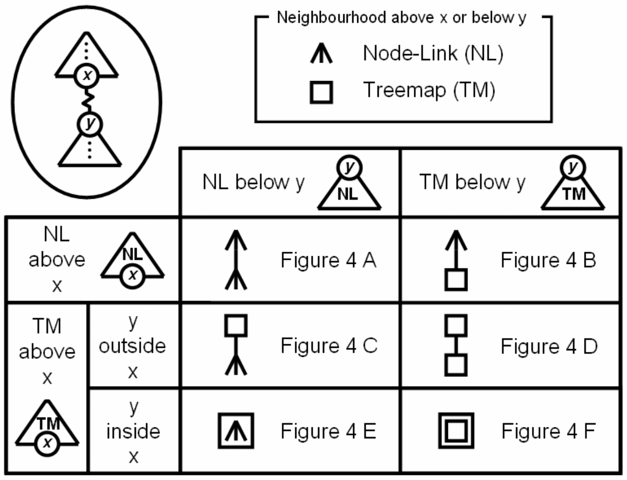


Figure 3: a taxonomy of graphical representations of the relationship between a node x and its child y. The neighbourhood of nodes above x, and the neighbourhood of nodes below y, can each by drawn in node-link (NL) or Treemap (TM) style.

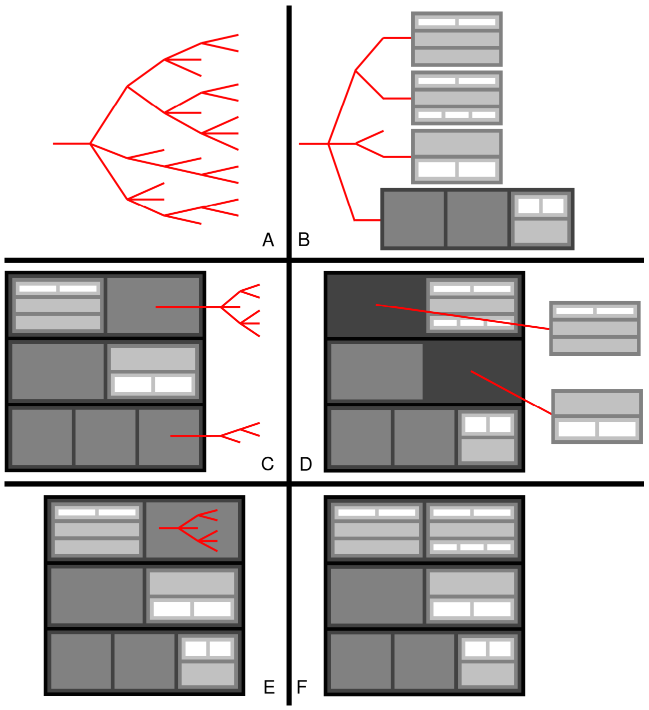


Figure 4: here, the same tree is depicted 6 different ways (illustration): in **A**, with a traditional node-link diagram, in **F**, with a Treemap, and in **B-E**, with mixed, hybrid representations.

be mixed in a straightforward manner without introducing ambiguity, as will be shown.

Figure 3 presents a taxonomy that can be used to generate and explain different methods for graphically combining Treemap and node-link styles of representation within a visualization of a hierarchy. The structure of an elastic hierarchy can be catalogued in terms of the types of transition that occur between different representational styles within the hierarchy.

While child nodes in a node-link diagram are always “outside” the parent node, this is not true for Treemaps, where a child node may be represented as either inside or outside a node (e.g., in Figure 4 **C** and **D** show Treemaps where parent nodes are linked to child nodes that are placed outside the parent, whereas Figure 4 **E** and **F** show child nodes that are nested inside the parent node.)

Based on the above description, six possible transitions can occur between node-link and Treemap styles as documented in Figure 3. The two rightmost columns of Figure 3 distinguish conditions where node-link style is used below some node *y* (left column), from conditions where a Treemap is used below the *y* node (right column). The three rows in the body of the table refer to the parent node *x* that is immediately above node *y* in the tree. The first row shows the case where the portion of the tree above node *x* is shown in node-link form. Thus, in the left column of the first row the entire subtree of interest is shown in node-link form (with node-link style used above node *x* and below node *y*), whereas in the corresponding column on the right a Treemap is shown nested underneath the node-link subtree that is above node *x*, leading to a mixed representation with the node-link style above a Treemap. The remaining two rows of the table show cases where a Treemap is used above node *x*. The middle row shows the situation where node *y* is shown as being outside node *x*, while the final row shows node *y* as being nested inside node *x*.

In both Figure 3 and Figure 4, panels **A** and **F** illustrate pure node-link diagram and Treemap styles, respectively. The remaining four transitional forms (**B**-**E**) use hybrids, and each of them will now be discussed.

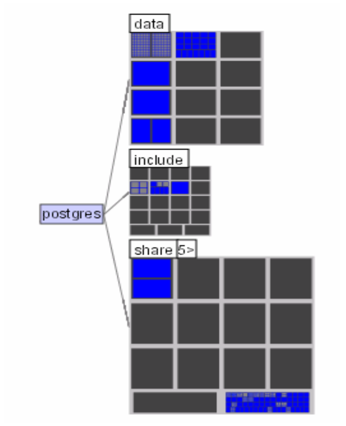


Figure 5: example of Treemap outside of node-link (screenshot). The node-link parent “postgres” has three subtrees (directories): “data”, “include”, and “share”, and each has many children laid out using the Treemap algorithm.

3.1 Treemap outside of Node-Link

In this first transitioning form (Figure 4 **B**, Figure 5) a subtree in a node-link diagram is shown as a Treemap. Since node-link diagrams of large hierarchies typically become more crowded at lower levels, the space saving properties of Treemaps allow more of the hierarchy to be shown within a given space. In contrast to a branch that might have hundreds of nodes that cannot be shown on the screen using nodes and links if fully expanded, a Treemap of those same nodes could be much more compact.

If the topology of the tree is of primary interest, this hybrid technique can show as many of the higher levels as convenient in node-link form. When the node-link style becomes too dense, Treemaps may be used to represent the deeper subtrees, making more information visible.

Small Treemaps can act as previews or thumbnails of the subtrees they contain, but with additional useful properties. First, they are not only previews but also overviews, containing information on the *entire* subtree, rather than just the first few nodes. Second, they are not just views, but also fully functional sub-hierarchies that can be operated on directly using a rich set of interaction techniques.

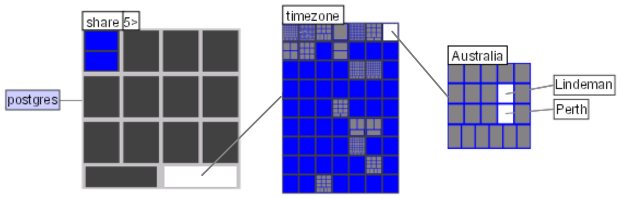


Figure 6: example of drilling down (screenshot). A combination of the Treemap outside of Treemap, and node-link outside of Treemap techniques is used to show deeper nodes.

3.2 Treemap/Node-Link outside of Treemap

In Figure 4 **C** and **D**, subtrees are "pulled out" of the Treemap, and shown as either node-link diagrams or additional Treemaps. This process might become confusing if used with a large number of nodes in a Treemap, but when used with a small number of nodes it may be well suited for focusing on particular subtrees, while retaining surrounding context in a space-efficient manner. This could function as an effective technique for drilling down. For example, in Figure 6, following the links from right to left, the leaf nodes "Lindeman" and "Perth" are node-link children of the Treemap node "Australia", which is itself a child of the Treemap node "timezone". The "timezone" node is a descendant of the "share" directory, and that node is a child of the node-link parent labeled "postgres" (an example of scheme **B**). Each of the Treemap nodes in the figure has numerous children that cannot fit inside the screen when fully expanded. By combining **C** and **D**, we are able to display all the relevant subtrees along the path within the screen. Note that because the edges connecting subtrees to the parent Treemap are overlaid on the Treemap, this results in mild occlusion which may be inconvenient if many subtrees are pulled out.

Having multiple foci in a hierarchy raises challenges for navigation and display. Points of interest could reside at distant locations within the hierarchy (or, in the case of a Treemap, appear very tiny), causing difficulties in showing them simultaneously on the same screen. Using techniques **C** and **D,** multiple foci can likely be shown more effectively than if viewed using non-hybrid representations (Figures 8 and 11).

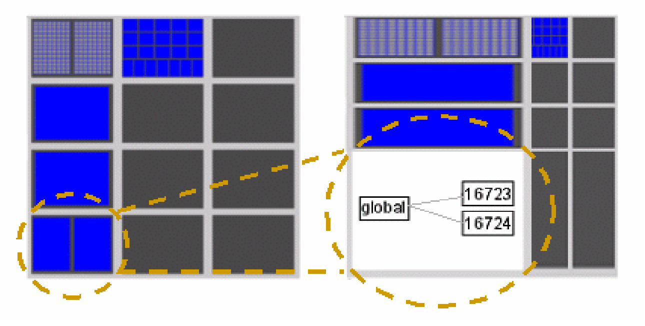


Figure 7: a design sketch showing a node-link diagram inside a Treemap. The subtree within the Treemap on the left is transformed into a node-link diagram on the right. The parent Treemap node adjusts its internal boundaries accordingly.

3.3 Node-Link inside Treemap

In this fourth kind of transition between styles (Figure 4 **E**), a node-link diagram is nested inside an enclosing Treemap. The Treemap acts as a kind of overview, and local features are presented using the node-link diagram. Globally, a space saving representation is used; while a standard node-link diagram is used locally. This can be thought of as a kind of detail-in-context technique with different representations used for the context and detail.

This scheme can be enhanced using an interaction technique where a portion of the Treemap is enlarged (similar to elastic windows [9]), either interactively or automatically, when transformed into a node-link diagram by the user (Figure 7). However, as different portions of the Treemap are expanded, this would change the sizes of subtrees and nodes within the Treemap, distorting substructures within the tree, which may hinder the formation and use of perceptual landmarks. Alternatively, one could limit the amount of distortion/expansion allowed, in which case the amount of space available to show embedded node-link diagrams would also be limited, thereby reducing the scalability of the method. Thus there may be a tradeoff between flexibility of presentation using distortion/expansion, and ease of perceptual orientation and landmark recognition and retention. It is possible that individual differences in perceptual and cognitive ability and preferences may determine how this tradeoff can best be handled for different users.

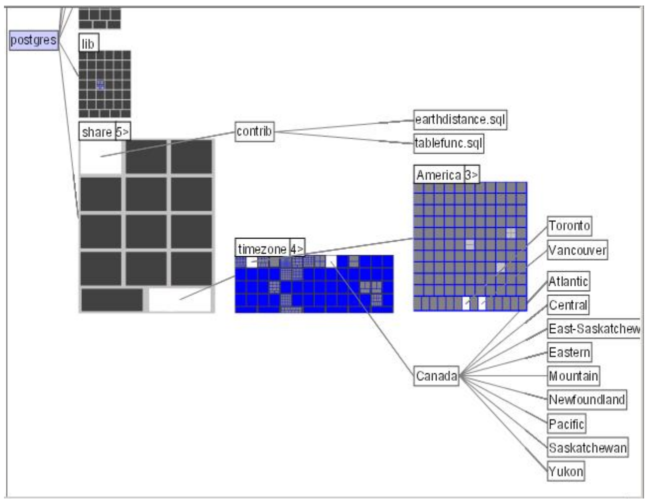


Figure 8: elastic hierarchy techniques allow users to explore and drill-down multiple branches of large trees and still fit much contextual information within a limited screen space (screenshot).

3.4 Combining the different techniques

Finally, these techniques may be used together and combined to create useful visual presentations. Figure 6 (described in section 3.2) and Figure 8 show combinations of **B**, **C**, and **D**. Likewise, **B** and **E** can also be combined. For example, in Figure 7, if the nodes labeled "16723" and "16724" each had many descendants, they could be replaced with two small Treemaps. There are many possibilities for such combinations from the four basic elastic hierarchical hybrid techniques **B**-**E**. Empirical research is needed to determine when different combinations of the basic techniques are useful.

4 Characteristics of Elastic Hierarchies

As already explained, elastic hierarchies are graphical depictions of trees containing multiple representation forms interleaved at different nodes, and that can be modified dynamically and interactively.

Ideally, compatible animation and interaction techniques should be used to facilitate exploration and interpretation of the hybrid structure in an elastic hierarchy. Elasticity can be exploited during interaction, as shown in the following examples. First, users may view different portions of a tree with different representations, and may transition between the different representations at will. This enables the fine-grained differentiation of nodes. For example, children of the same parent may be presented in different styles, with transitional animations being provided to guide or facilitate the transformation. Second, the size of a node (whether of Treemap, or node-link, or other style) could be resized to reflect user interest or structure priorities. Third, various other interaction techniques (coloring, brushing, etc.) could be used to support the perceptual and cognitive operations associated with manipulating and using the tree.

The main advantage of elastic hierarchies is how they allow flexible arrangement and display of contents and structures within a large tree. This is particularly useful for visualizing multiple foci. Typically, display of multiple foci in large trees using conventional methods creates challenges for navigation and display. Points of interest may reside at distant locations within the tree (or, in the case of a Treemap, appear very tiny), causing difficulties in showing them simultaneously on the same screen. This difficulty can be addressed using elastic hierarchies, where multiple foci can be highlighted within a single view (Figures 8 and 11).

In other words, elastic hierarchies generalize how nodes in a tree can be "collapsed". Conventional tree representations often only allow any *subtree* to be collapsed entirely, to "roll up" deeper nodes and save space. However, in an elastic hierarchy, any *connected subgraph* of the tree, such as intermediate levels between shallow and deep nodes, can be collapsed into a Treemap. Given a connected subgraph *S* of a tree, and the shallowest node *N* in *S*, we can display the subtree under *N* as a Treemap, and "pull out" of the Treemap any nodes under *N* that are not in *S*. Thus, distant branches under *S* can be shown pulled out and side-by-side, with the Treemap of *S* providing a compact overview of the context above the branches.

In order to experiment with elastic hierarchies, and investigate appropriate interaction and animation techniques, we created an interactive prototype that allows rapid transitioning between the different representational forms.

5 Implementation

There are many ways in which elastic hierarchies can be constructed and used. Our current prototype implements the transitional forms in Figure 4 **B**, **C**, and **D**, and allows them to be mixed and used together. Ideally, we would eventually like to support all of the possible schemes and allow arbitrary mixing of them, however the current implementation supports the most important possibilities and enables investigation of a large part of the design space.

Figures 5-12 show screenshots of various aspects of the working prototype, except for Figures 7 and 11 which are mocked-up design sketches. The prototype divides the screen vertically into two windows (Figures 9, 11, 12), with an overview on the left and a detail view on the right, to support overview+detail visualization, as discussed in more detail later. The overview shows a Treemap of the entire tree, and the detail view shows an elastic hierarchy of the same tree in which the user may zoom and pan.

Having two side-by-side views of the trees not only enables investigation of overview+detail visualization techniques [10, 12, 13], but also serves the following second purpose. Just as a single elastic hierarchy may help a user learn an unfamiliar tree style (e.g. Treemaps) by interactively transitioning between the unfamiliar style and a more familiar one (e.g. node-link), it may also be true that showing two views side-by-side, i.e. an elastic hierarchy in tandem with a non-elastic view (in this case, the Treemap on the left), could help reinforce a correct mental model in the user of the elastic hierarchy's meaning.

5.1 Platform & Code

The prototype was developed in Java 1.4.2 using the Piccolo Toolkit from the University of Maryland. Piccolo was chosen for its built-in support for zooming, panning, and certain types of animations. For laying out the Treemaps, we used a variant of the Strip Treemap algorithm in the open source Java library written by Ben Bederson and Martin Wattenberg [6]. We chose this algorithm because it preserves the ordering of the nodes, has better readability than the ordered Treemap algorithm, and has a reasonable running speed. The algorithm used for laying out node-link diagrams in 2D is similar to Walker's algorithm [20] and that used in SpaceTree [14]. The prototype can read in a tree described in a file, or can read in the tree structure of a hard drive's file directories (screenshots in this paper mostly show the directory structure of *Postgres*, a database consisting of almost 1000 files and folders installed on a hard disk).

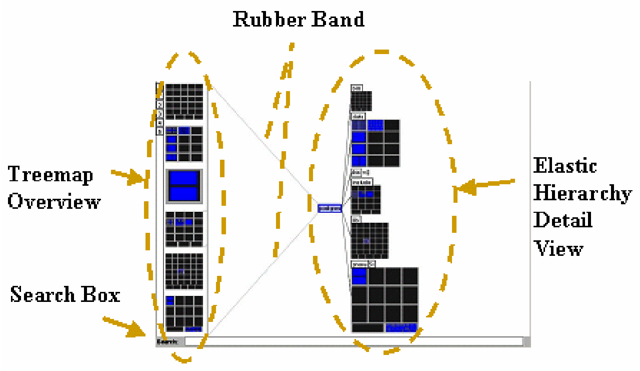


Figure 9: screenshot of the initial state of the prototype after reading in the Postgres dataset. The left window contains a Treemap overview of the tree. The right window shows an elastic hierarchy view of the tree. A search box is at the bottom.

5.2 Data Structure & Algorithms

Internally, the tree is stored in a data structure where each node can take on one of various graphical states corresponding to different representations, allowing the Treemap and node-link styles to be intermixed. Although our internal data structure allows for all the hybrids sketched in Figure 4, the supporting code so far only implements the schemes in Figure 4 **A**, **B**, **C, D,** and **F**, which are the most promising of the transitional forms.

Two complementary approaches are available for choosing representational forms and generating a layout of an elastic hierarchy: (1) automatic methods, which might use heuristics based on local attributes of a tree such as branching factor or depth, and (2) manual interaction, where the user explicitly specifies the representation to use for each node. We tried to strike a balance between these two approaches in the prototype, whereby the software makes a best first guess as to which style to use for each node, and the user may subsequently adjust specific nodes as needed.

When our prototype reads in a tree, it computes a layout that fills the available screen space with a hybrid of the type in Figure 4 **B**. Our intent is to give the user an initial view of the tree that uses the node-link style as much as possible, without excessive crowding, and to use Treemaps to show large subtrees wherever necessary.

Thus, the primary goal of the algorithm is to first maximize the number of higher-level nodes shown in the node-link style. To do this, the algorithm performs a greedy breadth-first traversal of the tree, setting the node type of each node encountered to be node-link, and stopping and/or backtracking when there is no room to continue.

Next, the second goal of the algorithm is to maximize the remaining area allotted to Treemaps used to show lower levels of the tree. To do this, the system first determines the bounding boxes of the Treemaps using a heuristic based on the size and depth of its subtree. The system recalculates the overall space needed, and resizes the Treemaps so that the entire structure is scaled to fit within the screen. In order to guarantee that the detail view has the same visual pattern as the overview, any Treemap appearing in both views is given the same aspect ratio in both, creating visual consistency across the views (Figure 9).

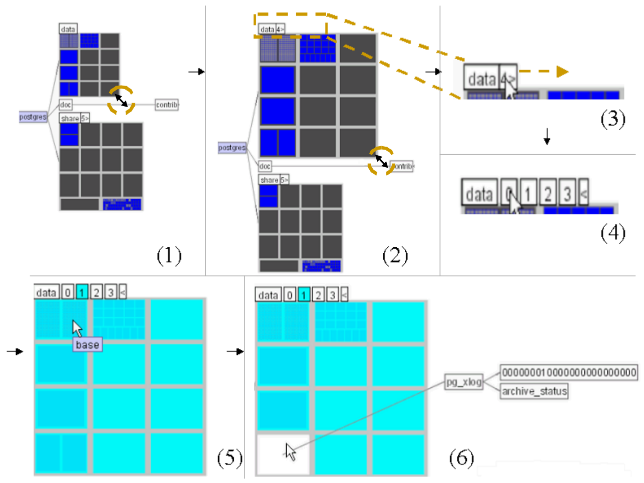


Figure 10: a sequence of user interaction allowed in the prototype (screenshots): (1) initiate resizing; (2) node resized, tabs appeared beside the labels; (3) and (4), click on the "4>" tab to show the labels of layers for the Treemap node; (5) layer 1 is selected; (6) double click to expand a subtree from layer 1.

5.3 User Interaction

The prototype elastic hierarchy system is designed for easy switching between Treemap and node-link views at different points in the tree. Users can right click on the label of a particular node to bring up a popup menu to change the form (Treemap or node-link) of the node. Users can also resize a Treemap node by dragging the bottom left corner of the node (Figure 10, 1-2), after which the tree layout is automatically adjusted. This allows users to examine in more detail the content of a Treemap, and select nodes with in it more easily.

Since the internal (i.e. non-leaf) nodes in a Treemap are graphically covered almost entirely by descendants, these can be difficult to select, even with borders and margins around nodes. Thus, a special selection technique was implemented for unambiguously selecting nodes of Treemaps. A set of tabs is displayed above each Treemap node (Figure 10, 3-4). The user may click on a tab to select the level at which they wish to select a node. Then, the user can rollover the nodes of the Treemap, and see the nodes at the chosen level highlighted (Figure 10, 5), with descendant nodes partially faded. The user can then double click to transform a node (Figure 10, 6) into a node-link subtree.

5.4 Additional Features

Elastic hierarchies can incorporate Treemaps at any point, allowing hybrid representations to scale better to large numbers of nodes than pure node-link diagrams. Ultimately, however, any incremental improvement in scalability will still fail given a sufficiently large data set. Thus we built our prototype to support an overview+detail view of the elastic hierarchy. A pure Treemap is shown in an overview window, and the hybrid Treemap/node-link diagram is shown in a detail window (Figure 9). A Treemap was chosen for the overview rather than a pure node-link diagram due to its space efficiency. The vertical division between the overview and detail windows in our prototype can be dragged to resize windows, accommodating different data sets and changes in the user's focus of attention.

Both elastic hierarchies and overview+detail visualization lend themselves naturally to viewing multiple foci. Although not yet implemented in our prototype, Figure 11 shows a sketch of how the user might select multiple foci in the Treemap overview, and

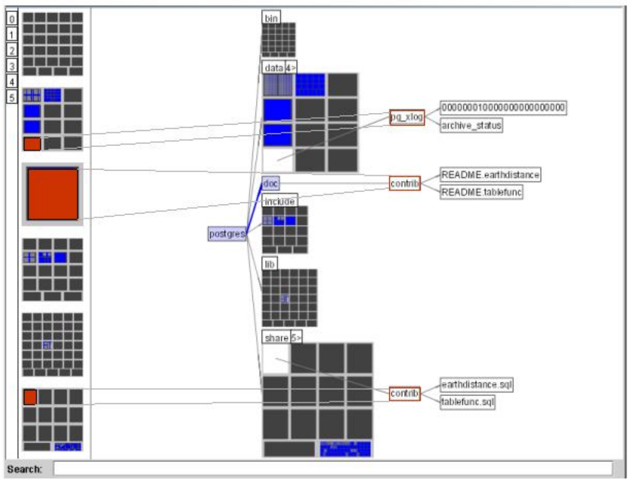


Figure 11: a design sketch of multiple foci in the tree. The foci are highlighted in red in both the overview (Treemap representation on the left), and the detail view (hybrid representation on the right). The hybrid representation allows us to provide much contextual information for the three foci.

in response, the detail view adjusts its layout to show the three foci simultaneously with rubber bands linking between the windows.

The hybrid mixing of multiple representations in elastic hierarchies is unfamiliar to users. This, combined with the fact that elastic hierarchies change form on demand, motivates the need for mechanisms that help the user understand the correspondence between nodes during changes of representation, and across the two views involved in our overview+detail scheme.

To investigate this, our prototype implements multiple strategies. First, we have experimented with using smooth animations to display transitions between representation styles, to help the user maintain context, as has been discussed elsewhere [4, 5, 21]. Second, whenever a Treemap is visible in the detail view, its aspect ratio is constrained to match the aspect ratio of the corresponding (sub)Treemap in the overview window, to make it easier for the user to visually scan for corresponding subtrees. This aspect ratio is maintained even during resizing of Treemaps by the user. Third, rubber bands (Figures 9 and 11) are drawn that connect selected nodes in the overview and detail view. Fourth,

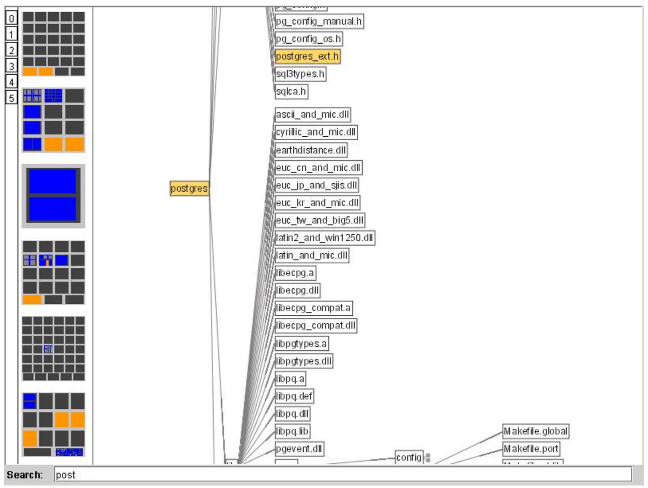


Figure 12: node searching in the prototype (screenshot). The key word is typed in the search box at the bottom. Matches are shown in orange in both the Treemap overview on the left and the detail view on the right.

because the representation in the overview is persistently a Treemap, the user may change a subtree in the detail view from Treemap style to node-link style, and still see the Treemap form of the subtree in the overview.

Thus, the user has a choice of either (a) looking at different representations simultaneously of the same nodes, each in different windows, or (b) toggling in-place between representations of nodes that are shown within a single, integrated view (i.e. the detail view).

6 DISCUSSION AND CONCLUSION

Elastic hierarchies have many valuable properties. They may be useful in dealing with a wider range of information. Since our hybrid tree representations can use compact Treemaps for various portions of the tree, it is possible to introduce space-efficiency as needed. Thus different nesting strategies within elastic hierarchies create a continuum of space efficiency between node-link diagrams at one end of the continuum and Treemaps at the other. A side-benefit of elastic hierarchies is that, by allowing the user to toggle between traditional node-link diagrams and Treemaps at any point in the tree, this may help users to better learn and understand Treemaps (which are themselves unfamiliar to most users) by seeing how they relate to familiar node-link diagrams.

Elastic hierarchies can be combined with other visualization strategies, such as the use of multiple views, or focus+context approaches. While this creates a large design space, it is one where there may be a number of useful “sweet spots” that will only be discovered after there is an opportunity to examine the properties of the space with prototyping and user studies. The provision of multiple representations both within a tree (i.e., using an elastic hierarchy) and between multiple views may help users learn and understand the content and structure of hierarchies better. The ability to see the same tree rendered in different ways, and to see the correspondence between elements in the different views, may encourage the development of more accurate mental models of information structure.

There is a large design space of possible elastic hierarchy implementations of which this paper has considered a small portion. Systematic research is needed to reinforce and challenge the design intuitions that underpin this form of hierarchy visualization. One obvious research issue which overlaps with both the present work and earlier research on hierarchy visualization is the relationship between the number of nodes in a hierarchy and which representational forms should be used at different levels of the hierarchy. Other research questions (a selection from many that could be posed) that may be worth pursuing with reference to the design and use of elastic hierarchies include:

- How and when should users be able to choose which representations to use (vs. having layouts assigned automatically)?
- What elastic hierarchies using representations other than Treemap and node-link diagrams could be designed?
- What types of design cues and strategies can be used to facilitate the formation and use of cognitive/perceptual landmarks within large hierarchies?
- How can smooth animation facilitate the use of multiple views (e.g., overviews, “you-are-here” diagrams, detail views, etc.)?
- How should elastic hierarchies be personalized for different types of user?

One possible future direction for elastic hierarchy prototypes would be to generalize the roles played by the overview and detail views, and to link multiple views with various types of visualization and animation to highlight correspondences.

In our exploratory prototype, the local file system is chosen as the content for the elastic hierarchies. A next step could be using elastic hierarchies for other kinds of real world data, and further investigating their characteristics under these domains.

In conclusion, the research reported in this paper investigated the use of elastic hierarchy representations for trees. A design space for elastic hierarchies was characterized where a single graphical depiction uses a mixture of Treemap and node-link views at different levels of the tree. A prototype was created to demonstrate and explore related design features and to illustrate some of the properties of elastic hierarchies. Empirical research is now needed to determine if, when, and how, elastic hierarchies can be used.

7 ACKNOWLEDGMENTS

Thanks to Ben Bederson and Ben Shneiderman for some detailed feedback and pointers to related work. Thanks to Amy Zhu, Maneesh Agrawala, Joe Laszlo, Jim Chengming Cai, and Noah Lockwood for support and feedback. Thanks to Ravin Balakrishnan and other members of the DGP lab at University of Toronto for feedback during an early stage of this work. Thanks to John Hancock for technical help with making the video. Thanks to University of Maryland’s HCIL for making Piccolo, SpaceTree, and Treemap demos, code, and datasets available to the public. Thanks also to the anonymous reviewers for their constructive feedback and suggestions.

REFERENCES

[1] Andrews, K., *Visual exploration of large hierarchies with information pyramids*. Proceedings of Sixth International Conference on Information Visualisation. 2002: IEEE Computer Society Press. 793-798.

[2] Andrews, K. and Heidegger, H., *Information slices: Visualising and exploring large hierarchies using cascading, semi-circular discs*. Proceedings of IEEE Symposium on Information Visualization, Late Breaking Hot Topics. 1998. 9-12.

[3] Balzer, M. and Deussen, O., *Hierarchy based 3D visualization of large software structures*. Proceedings of the Conference on Visualization Posters Compendium. 2004: IEEE Computer Society. 81-82.

[4] Bartram, L., *Can motion increase user interface bandwidth?* Proceedings of IEEE Conference on Systems, Man, and Cybernetics. 1997. 1686-1692.

[5] Bartram, L., *Perceptual and interpretative properties of motion for information visualization*. Proceedings of the 1997 Workshop on New Paradigms in Information Visualization and Manipulation. 1997, Las Vegas, Nevada, United States: ACM Press. 3-7.

[6] Bederson, B. and Wattenberg, M., *Treemap Java Algorithms.* http://www.cs.umd.edu/hcil/treemap-history/Treemaps-Java-Algorithms.zip.

[7] Fekete, J.-D., Wang, D., Dang, N., Aris, A., and Plaisant, C., *Overlaying Graph Links on Treemaps*. Proceedings of IEEE Information Visualization Symposium Posters Compendium. 2003: IEEE Computer Society.

[8] Harel, D., *On visual formalisms.* Communications of the ACM, 1988. **31**(5): p. 514-530.

[9] Kandogan, E. and Shneiderman, B., *Elastic Windows: a hierarchical multi-window World-Wide Web browser*. Proceedings of the 10th

annual ACM symposium on User interface software and technology. 1997, Banff, Alberta, Canada: ACM Press. 169-177.

[10] Kumar, H.P., Plaisant, C., and Shneiderman, B., *Browsing hierarchical data with multi-level dynamic queries and pruning*. International Journal of Human-Computer Studies, 1997. **46**(1): p. 103-124.

[11] Lamping, J., Rao, R., and Pirolli, P., *A focus+context technique based on hyperbolic geometry for visualizing large hierarchies*. Proceedings of ACM Conference on Human Factors in Computing Systems. 1995. 401-408.

[12] Mukherjea, S., Foley, J.D., and Hudson, S., *Visualizing complex hypermedia networks through multiple hierarchical views*. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 1995, Denver, Colorado, United States: ACM Press/Addison-Wesley Publishing Co. 331-337.

[13] North, C., *A user interface for coordinating visualizations based on relational schemata: snap-together visualization*, Ph.D. Thesis, Computer Science Department, University of Maryland, 2000.

[14] Plaisant, C., Grosjean, J., and Bederson, B.B., *SpaceTree: Supporting exploration in large node link tree, design evolution and empirical evaluation*. Proceedings of IEEE Symposium on Information Visualization. 2002. 57-64.

[15] Reingold, E.M. and Tilford, J.S., *Tidier drawings of trees.* IEEE Transactions on Software Engineering, 1981. **SE-7**(2): p. 223-228.

[16] Robertson, G.G., Mackinlay, J.D., and Card, S.K., *Cone trees: Animated 3D visualizations of hierarchical information*. Proceedings of ACM Conference on Human Factors in Computing Systems. 1991. 189-194.

[17] Shneiderman, B., *Tree visualization with tree-maps: 2-d space-filling approach.* ACM Transactions on Graphics, 1992. **11**(1): p. 92-99.

[18] Sindre, G., Gulla, B., and Jokstad, H.G., *Onion graphs: aesthetics and layout*. Proceedings of IEEE Symposium on Visual Languages. 1993. 287-291.

[19] Stasko, J. and Zhang, E., *Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations*. Proceedings of IEEE Symposium on Information Visualization. 2000. 57-65.

[20] Walker, J.Q., *A node-positioning algorithm for general trees.* Software--Practice and Experience, 1990. **20**(7): p. 685-705.

[21] Woods, D.D., *Visual momentum: a concept to improve the cognitive coupling of person and computer.* International Journal of Man-Machine Studies, 1984. **21**: p. 229-244.