A User Study of Techniques for Visualizing Structure and Connectivity in Hierarchical Datasets

Tommy Dang[1], Paul Murray[2], Ronak Etemadpour[3], and Angus G. Forbes[4]

[1]Texas Tech University, Lubbock, TX, USA
[2]The New York Times Interactive News Desk, New York, NY, USA
[3]Oklahoma State University, Stillwater, OK, USA
[4]University of California, Santa Cruz, CA, USA

Abstract. Many tree layouts have been created for presenting hierarchical data. However, layouts optimized for some tasks are not adequate for others. In this paper, we focus on identifying tree structures and cross-links generated by hierarchical edge bundling. Our key contribution is the introduction of descriptive features that can be used to characterize trees in terms of their structural and connective qualities. We present a user study with 14 subjects that provides an evaluation of our approach in comparison to other popular tree visualization techniques. The results of the study indicate which techniques are more effective for visual analysis tasks that involve identifying and comparing tree and subtree structures and/or visualizing connections using hierarchical edge bundling.

Keywords: Hierarchical edge bundling, tree layouts, user evaluation.

1 Introduction

An important consideration in visualizing complex hierarchical data, such as biological pathways, phylogenic trees, and ontological taxonomies, is clearly showing how particular elements relate to or are influenced by other elements. That is, it can be necessary to highlight relevant interconnected subtrees with a particular directionality, while at the same time making sure not to obscure the structure and hierarchical information represented by the tree. Many existing approaches [1, 15, 21] investigate the use of hierarchical edge bundling techniques [14], and our previous work also introduces *CactusTree* [4], an interactive technique for visualizing the structure and connectivity in nested trees.

In this paper, we explore the effective use of tree layouts to support hierarchical structure recognition and to minimize ambiguity introduced by bundling cross-edges (also called non-hierarchical connections/links). A user study that uses both real-world and synthetic datasets validates the effectiveness of our approach. Additionally, the results of our user study offer preliminary guidelines for identifying or constructing layouts that are appropriate for tasks requiring the analysis of linked data and ontologies.

2 Related Work

Schulz [27] maintains *Treevis.net*, a comprehensive website that describes of a large number of tree layouts (296 in total as of September 2017) gathered from conference

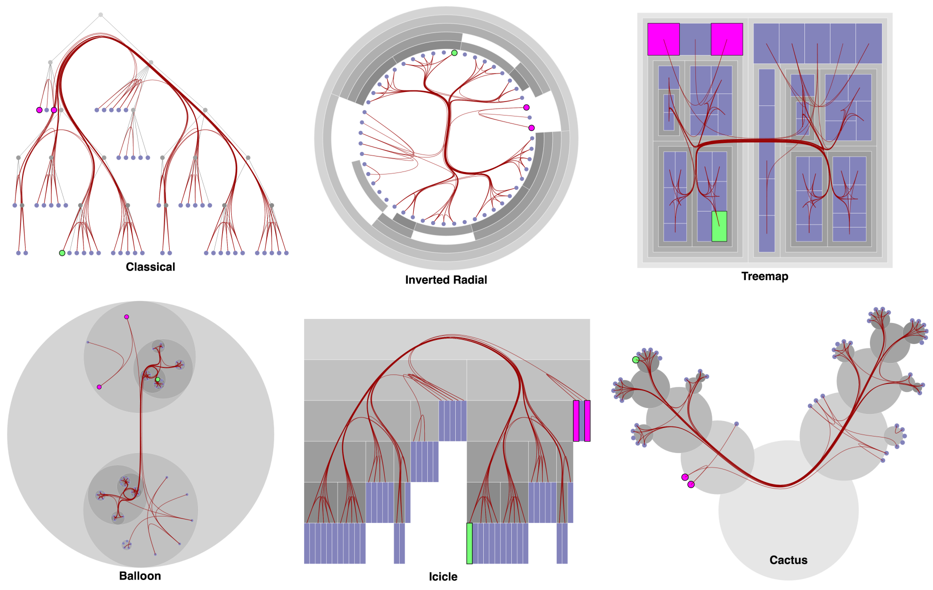

Fig. 1. Examples of hierarchical edge bundling on a hierarchical dataset using different tree layouts. Connectedness between leaf nodes is more visible in some layouts than others: The two purple nodes are connected (via red links) and these purple nodes are disconnected from the green node.

proceedings and journal articles. Each of these layouts have advantages and disadvantages when used for particular tasks. In this section, rather than attempting to survey all related visualization techniques, we instead highlight related work representative of the main approaches to visualize hierarchical datasets.

A TreeMap [30] is a space-filling technique that maps a hierarchical dataset onto a rectangular region. The effective use of space enables comparison of attributes of leaf nodes such as size and color coding, and therefore helps to highlight patterns and outliers in large hierarchies. ArcTrees [24] additionally overlay non-hierarchical links onto TreeMaps [10]. Icicle Plots [17] encode hierarchical data by stacking child rectangles directly on top of parent nodes. This makes it easier to see the hierarchical structure, but also assigns valuable screen space in assigning large areas to intermediate nodes. When using Icicle Plots to represent dense datasets that contain a large number of leaf nodes, the leaf nodes can be pushed close together, making them hard to see. Viegas et al. [32] propose a solution to better utilize the space in a circle packing algorithm by defining a new center point of each "balloon." However, the hierarchical structure can be difficult to interpret for tree datasets with more than a few levels of depth.

Kobsa [16] describes a study to compare several well-known information visualization systems for tree hierarchies in a between-subjects experiment. The study showed a significant difference in completion times and correctness between structure-related versus attribute-related tasks on various tree layouts. McGuffin and Robert [22] present an in-depth survey of tree layouts that introduces a range of metrics to define the information density of different tree layouts. These metrics provide design guidelines for the use of layouts for certain tasks, such as maximizing space-efficiency and supporting labeling.

Hierarchical edge bundling (hereafter, HEB) groups links between adjacent edges by routing them through parent nodes in order to re-enforce the hierarchical structure of the data. HEB is widely used for a range of applications; however, HEB has not been evaluated systematically. In a survey paper on edge bundling techniques, Zhou et al. [36] summarize some studies of HEB, which tend to indicate user preference for visualizations that use HEB in comparison to those that do not. While not explicitly focused on HEB, Xu et al. [35] examine visualizations that use varying degrees of curvature, finding that links with high-curvature can adversely affect how well users interpret data. McGee and Dingliana [21] perform user experiments to evaluate the impact of bundling on user performance on different tasks using a set of randomly generated undirected compound graphs with varying sizes and edge densities. In their study, graphs are presented with a range of different levels of edge bundling using only a simple balloon tree layout. Moreover, the design of McGee and Dingliana's study limits the applicability of their results to 3 layers of depth and the interconnectivity in their generated graphs does not reflect real-world edge densities.

3 Tree Qualities and Visualization Tasks

There are many tasks related to visualizing compound graphs in a range of scientific domains, including those that involve biological pathways [20], ontology alignment [23], and taxonomic classifications [5]. Although there are more basic tasks for tree layouts, such as determining the degree [18] or height [33, 16] of a tree, through in-depth discussions with systems biologists, taxonomists, and ontology researchers, we identified two primary tasks important for visual exploration of hierarchical datasets: characterizing hierarchical structures and identifying connections between nodes in the hierarchy.

T1: Effectively characterize hierarchical structure—It is not uncommon find a pathway which contains more than ten nested levels [31]. Similarly, in taxonomic and ontological alignment domains, hierarchies can get extremely complicated, potentially containing thousands of leaf concepts [5, 11]. More importantly, these classifications (hierarchies) can be contentious, and they may change from year to year as new discoveries or interpretations are made [13]. Having a representation which captures hierarchical structures effectively and allows users to visually identify structures/determine structural changes quickly is highly desirable by domain experts.

T2: Minimize ambiguity introduced by edge bundling—When HEB is applied, tracing a bundled link can lead to the perception of incorrect connectivity if edges are not clearly separated within the bundles [1]. For some tree layouts the loss in detail (i.e., the ability to trace connectivity between two nodes) can be amplified depending on various factors, such as the chosen visual encodings and the overall structure of the tree. An ideal tree visualization technique to best support HEB should minimize this loss.

	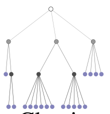	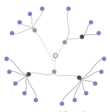	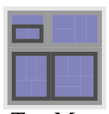	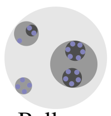	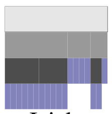	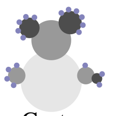
Tree layouts	Classic	Radial	TreeMap	Balloon	Icicle	Cactus
Hierarchical relationship						
Node-link	🟥✓	🟥✓				
Containment			🟨✓	🟨✓		
Stacking					🟩✓	🟩✓
Shape Preservation						
Stable	🟩✓			🟩✓	🟩✓	🟩✓
Malleable		🟥✓	🟥✓			
Space-centric filling						
Root-centric	🟥✓	🟥✓			🟥✓	
Parent centric			🟩✓	🟩✓		🟩✓
Bundling angularity						
Wide				🟩✓		🟩✓
Sharp	🟥✓	🟨✓	🟨✓		🟥✓	

Table 1. Tree layouts and their qualities. Cells are colors based on our hypotheses about whether or not these qualities support the primary tasks in our user study (in Section 4): red means that this quality will not effectively support these tasks; green means that it will be supportive; and yellow means that it falls somewhere in between or is neutral.

3.1 Classifying Layouts to Support Visualization Tasks

Hierarchical edge bundling can be used in conjunction with existing tree visualization techniques (balloon layout and radial cluster tree are the two prominent examples), but each layout introduces certain drawbacks. When we explicitly draw the hierarchy underneath the relationships (in the case of classic tree layout or TreeMaps), the visualization becomes more cluttered and it is difficult to interpret the hierarchical information. When we separate the hierarchy and connectivity (in the case of using outer rings to depict hierarchy in radial tree), it becomes less intuitive because viewers need to interpret them independently.

Schulz [27] classifies tree layouts in terms of three main features, based on their structural layout: *dimensionality* (2D or 3D), *representation* (explicit or implicit), and *alignment* (axis-parallel, radial, and free). We refine Schulz's *representation* categorization by further dividing implicit representations into two sub-categories, *containment* and *stacking*, in addition to *node-link* representations. These categories have distinct features which impact the visual characteristics of HEB. In addition to refining the structural classification of the tree layouts, we also identified how the layouts are either amenable to or inappropriate for edge bundling techniques. We can thus characterize each tree layout in terms of the following four "qualities" (see examples of tree layouts in Table 1). We hypothesize that, in each case, having a particular quality is superior to not having it, at least in terms of supporting our two primary tasks. We test this explicitly in Section 4.

Hierarchical relationship: Node-link vs. containment vs. stacking This describes the encoding of a parent-child relationship by either: (a) drawing a link (*node-link*, such as in Classic trees or Radial trees); (b) nesting children within the parent (*containment*, such as TreeMaps or Balloon layouts); or (c) having a spatial area of the child abut its parent (*stacking*, such as Icicle plots).

Shape preservation: Stable vs. malleable structure How a subtree appears is affected by the structure of the tree (such as the number of leaf nodes) and where it locates in the tree. *Stable* structures may have different scales/rotations, but overall shapes are preserved. A layout with a *malleable* structure means that the exact same subtree may appear very different when positioned in different trees, or in different places within the same tree.

Space-centric filling: Root-centric vs. parent-centric In *root-centric* layouts, all layout operations are made with respect to the tree's root. In *parent-centric* layouts, all layout operations are made with respect to a node's parent [28]. Based on this classification, Classic, Radial, and Icicle are *root-centric* since the subdivision for leaf nodes are computed with respect to the root node.

Bundling angularity: Wide turns vs. sharp turns This quality describes the ease of interpreting HEB overlaid on tree layouts. In general, layouts that have child nodes distributed in a linear manner with regard to their parent center have sharper turns than layouts that have child nodes distributed in a circular manner. For example, Radial trees (having child nodes distributed in a circular manner with regard to their root) have *wider turns* compared to Classic trees and have *sharper turns* compared to Balloon layouts (since Radial trees are *root-centric* while Balloon layouts are *parent-centric*).

4 User Study

For our controlled experiments we recruited 14 subjects (11 males and 3 females), aged between 22 and 38 with normal vision. All of the subjects had some familiarity with tree layouts, and a short training session was provided to give a further overview of each of the six layouts that were used in the study. The layouts shown to the subjects had sizes in the range 900x700 to 1200x1050 pixels. Subjects were allowed to touch the screen or to use a mouse or trackpad to move the mouse cursor, but the experiments involved no interaction (other than to select and confirm answers to the Yes/No questions).

The total length of the study (including two experiments) ran from between 30 minutes to just under one hour depending on the speed of the participant. The subjects were asked to answer the questions as quickly and accurately as possible. We collected accuracy and completion time for each quantitative task (described below); user preference data were also collected. The order of the questions was randomized to avoid bias and learning effects.

For our user study, we chose representative examples of trees with the features described in Section 3.1, including hierarchical relationship, shape preservation, space-centric filling, and bundling angularity. The 6 trees we used are shown in Table 1: Classic, Radial, TreeMap, Balloon, Icicle, and Cactus.

There are many variants of the same tree layout and multiple ways to visual encode them. Through a preliminary pilot study, we identified the best variant and visual encoding in each layout based on user performances. For example, for our representation of TreeMap, we decided to use a TreeMap with margins to better represent hierarchical structures. We also use gray shadings for all 6 tree layouts to encode different levels in the hierarchy; although some layouts may benefit from this encoding more than others. Examples of the layouts and their relevant features are presented in the top row of Table 1, along with an indication of how well we thought the layouts would perform in our user study.

4.1 Experiment 1: Identifying Subtrees

Our first experiment evaluates the ability of participants to identify the existence of a subtree within a larger tree. This task has been confirmed to be valuable by the domain experts we worked with to develop *CactusTree* [4] and previous visualization techniques for representing interconnected hierarchical datasets [3, 5, 6, 8, 9, 7]. For example, the *RAF cascade* pathway (used in Table 1, where it looks like a snowman in *CactusTree* representation) is duplicated four times in the larger *Signalling by NGF* pathway.[1] In general, it can be important to facilitate hierarchical structure recognition tasks within tree layouts. For instance, in taxonomic studies, these taxonomic classifications may change from year to year (or even more often) as new discoveries are made [13].

In this experiment, we used 10 different datasets for subtrees, each of which were synthesized from real-world datasets from different domains, including biological pathways, *flare* source code packages,[2] and mammal hierarchy [26]. We selected these datasets for subtrees because they characterize the inherent complexity of real-world scenarios. These subtrees have from 3 to 5 levels of depth. The depth of a tree is defined as the number of edges on the longest path from the root to a leaf. The degree of a tree is defined as the maximum degree of any of its nodes (the degree of a node is the number of its children). The larger search trees are generated randomly with 6 level of depth and 6 degrees. Since our study does not explicitly evaluate zooming or other interaction techniques, these maximums ensure that the generated search trees are neither too small nor too large, which could cause a layout to be completely obvious or to draw tiny trees that are impossible for any user to recognize.

Our experiment was extended on the following dimensions: 6 tree visualizations x 10 subtree datasets = 60 questions. In particular, for each tree layout we asked 10 questions (associated with 10 subtrees) in which 5 contain a subtree and 5 do not. For those that do contain a subtree, we blend the subtree into a random position in the generated tree. The order of questions and tree layouts are completely randomized.

[1] http://www.reactome.org/PathwayBrowser/#/R-HSA-166520
[2] http://flare.prefuse.org

Hypotheses for the Identifying Subtrees Task Based on our analysis of meaningful features for hierarchical structure recognition tasks, we hypothesized **H1**—Layouts that (a) preserve the shape of the data regardless of where it is positioned in the layout, and that (b) use *stacking* to represent structure will perform better than those that do not, both in terms of completion time and accuracy, for the subtree task (**T1**).

More specifically, we ranked the six layouts according to how these qualities were emphasized. Each tree layout exhibits these qualities to more or less of a degree, but generally, we expect shape preservation to be most important, followed by the type of hierarchical relationship. The following hypotheses about performance rankings are based on this expectation: **H1.1**—*Cactus*, which can uniformly scale and rotate the shape, but do not otherwise modify it, will outperform all other layouts in terms of completion time and accuracy. **H1.2**—*Icicle*, which can narrow the nodes representing data when positioned within a larger tree, but otherwise retain its shape, will not perform as well as Cactus, but perform better than other layouts. **H1.3**—*Classic*, similarly to Icicle, can narrow the nodes representing data when they are positioned within a larger tree, and will not perform as well as Cactus. Classic will also not perform as well as Icicle since it also uses lines rather than shapes to indicate structure. **H1.4**—*Balloon*, although the shape of the data is preserved, nodes in general difficult to see because they are nested within each other, and become hard to view at deeper levels. Thus, Balloon will perform worse than Cactus, Icicle, and Classic, both in terms of accuracy and completion time. **H1.5**—*TreeMap* suffers from both having nested data and from lacking any guarantee that shape will be preserved. Thus, TreeMap will perform worse than Cactus, Icicle, Classic, and Balloon, both in terms of accuracy and completion time. **H1.6**—*Radial*, which does not preserve layout and uses lines to indicate structure, will perform worse than all other layouts, both in terms of accuracy and completion time.

4.2 Experiment 2: Path Tracing

Path tracing is a basic task in network visualization [21, 25]. We asked participants to identify whether or not two highlighted nodes in a tree are connected using edge bundling. In our second experiment, we used 4 different datasets for subtrees drawn directly from real-world data used in different domains. The data was ranked in terms of its complexity, that is, its depth, number of leaf nodes, and number of links used in the edge bundling: *easy* (4 levels, 3 max branches, 49 leaf nodes, 39 links); *medium* (4 levels, 4 branches, 87 leaf nodes, 81 links); *hard* (4 levels, 7 branshes, 105 leaf nodes, 156 links); and *hardest* (4 levels, 7 branches, 103 leaf nodes, 267 links). We used these datasets because they reflect real-world, non-hierarchical link densities (the non-hierarchical links to the number of nodes). We further randomized these data by swapping 2 branches within 3 levels from the root. This generates more trees and ensures that participants will never see the same tree twice; swapping does not effect non-hierarchical link density.

Our experiment was extended on the following dimensions: 6 tree visualizations x 4 datasets x 4 correct answers (2 Yes and 2 No)= 96 questions. Each pair of highlighted nodes was selected randomly. The order of questions and tree layouts are also completely randomized. The examples of the 4 original datasets in Cactus and Classic (without any blending or swapping of subtrees) are presented in Fig. 2, ordered from left to right in terms of difficulty of the increasing complexity of the data.

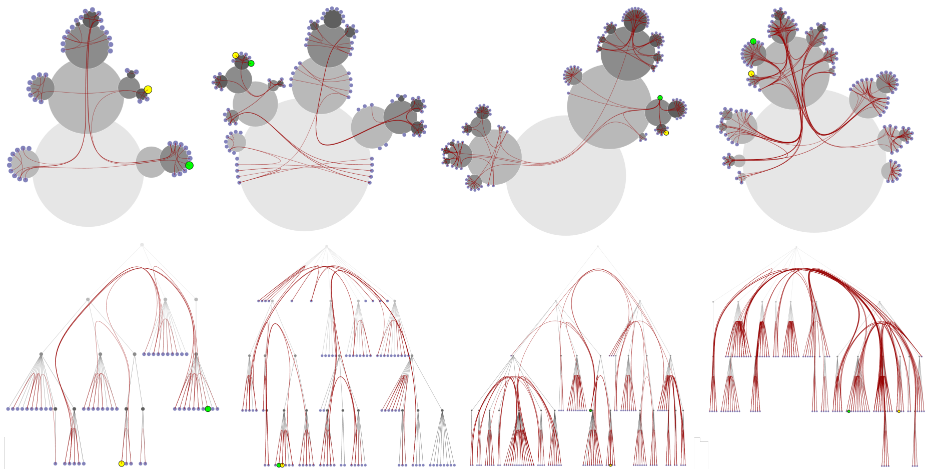

Fig. 2. Visualizing 4 input datasets using HEB on *CactusTree* (top row) and classic tree (bottom row). The difficulty of data increases from left to right. The same pair of nodes is highlighted in both tree representations for each of the four datasets.

Hypotheses for the Path Tracing Task Based on our analysis of meaningful features for tree layouts, we hypothesized that layouts that emphasize a wider bundling angularity and that use a different visual encoding to differentiate hierarchical structure and non-hierarchical connectivity (i.e., containment and stacking) would perform better for the connectivity task: **H2**—Layouts that (a) use different visual encodings to represent hierarchy and connectivity, that (b) reduce sharp turns in edge bundling, that (c) and avoid inward nesting will perform better than those that do not, both in terms of completion time and accuracy, for the connectivity tracing task (**T2**).

As with the first experiment, we ranked the six layouts according to how these qualities were emphasized. Each tree layout exhibits these qualities to more or less of a degree, but generally, we expect hierarchical relationship to be most important, followed by bundling angularity. The following hypotheses about performance rankings are based on this expectation: **H2.1**—*Cactus*, which clearly differentiates between visual encodings and which uses a wide bundling angularity, will outperform all other layouts in terms of completion time and accuracy. **H2.2**—*Icicle*, which clearly differentiates between visual encodings but introduces sharp turns in edge bundling, will not perform as well as *CactusTree*, but will perform better than other layouts. **H2.3**—*Balloon*, which nests inwardly, but which supports wide bundling angularity will perform worse than *CactusTree* and Icicle Plots in terms of both time and accuracy. **H2.4**—*TreeMap*, which nests inwardly, but uses a different visual encoding to differentiate hierarchical structure and non-hierarchical connectivity, will perform worse than Cactus, Icicle, and Balloon. **H2.5**—*Radial*, which does not clearly differentiate between visual encodings, but which tends to have wider bundling angularity will perform worse than all layouts, except *Classic* trees in terms of both time and accuracy. **H2.6**—*Classic*, which does not clearly differentiate between visual encodings, and which introduces sharp turns in edge bundling, will perform worse than all other layouts.

5 Results

Several aspects were considered for the statistical analysis of the results of the user study. First, we compared the six methods for each of the tasks by looking into the mean errors over all subjects and all data sets. Second, we did the same comparisons considering the time it took the participants to fulfill the tasks. For all analyses, we computed means and standard deviation of the errors. To test for statistical significance of the individual results, we first tested the distribution of the error values against normality using the Shapiro-Wilk tests [29]. All of our data had non-normal distribution, thus we applied the non-parametric Friedman test [12] on K related samples when comparing more than two groups, and Kendall's Coefficient of Concordance [19]. If the computed differences were significant, we performed pair-wise comparisons of the groups using a Wilcoxon test [34] on non-parametric two related samples to be able to report which groups particularly differ from each other. For pair-wise comparisons in case of more than two groups we run a series of Holm's sequential Bonferroni adjustment at the 0.05 level.

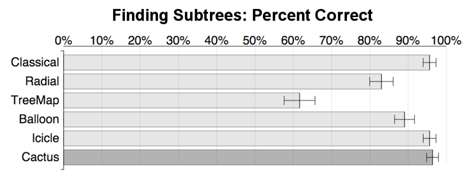

Fig. 3. Percent Correct (higher is better) for identifying subtrees (**T1**) for each of six layouts. Our *CactusTree* is highlighted in darker bars.

5.1 Results of Experiment 1

Fig. 3 summarizes the results of the comparative analysis of the six different methods for finding subtrees (**T1**). The bar charts show the mean error values and the standard error from the mean. The omnibus tests for statistical significance showed that there is statistical significance in the mean errors for some of the tasks. Kendall's Coefficient of Concordance test showed significant difference (Kendall W=0.149, χ^2=102.76, df=5, p<0.01) among six methods. Bonferroni across pair-wise Wilcoxon comparisons showed significant differences between TreeMap and all other techniques, with TreeMap as the least accurate technique for finding sub-tree. Other pair-wise comparisons also showed significant differences: Icicle vs. Radial (Z=-3.343, P<0.0035), Cactus vs. Radial (Z=-3.34, P<0.0035) and Radial vs. Classic (Z=-2.942, P<0.0032). Therefore, according to Bonferroni adjustment, Icicle and Cactus are ranked as the best techniques. Classic is ranked as the second best technique. Balloon and Radial are ranked as the third best techniques and TreeMap is the worst technique.

Fig. 4 summarizes the results of the time analysis of the six different methods for finding subtrees (**T1**). The bar charts show the mean error values and the standard error from the mean. The omnibus tests for statistical significance showed that there is significance in the mean errors for some of the tasks. Kendall's Coefficient of Concordance test indicates significant difference (Kendall W=0.164, χ^2=114.276, df=5, P<0.0001) among six methods. Pairwise comparisons between TreeMap and other techniques other than Radial Tree showed significant difference. TreeMap is one of the slowest techniques while Radial was significantly slower than TreeMap (Z=-2.153, P<0.031). Thus, Radial is ranked as the slowest techniques followed by TreeMap. The significant results from Bonferroni across pair-wise Wilcoxon comparisons between Cactus and Classic (Z=-3.747, P<0.001) and Cactus and Balloon (Z=-2.986, P<0.003) also reveal Cactus to be one of the fastest techniques. Similarly, Icicle is ranked as one of the fastest method as revealed significant difference in comparison with Classic (Z=-4.178, P<0.001) and Balloon (Z=-2.548, P=0.011). These results show a consistent results with accuracy for finding because Icicle and Cactus both ranked as the two most accurate methods, while TreeMap was both the least accurate method and the slowest.

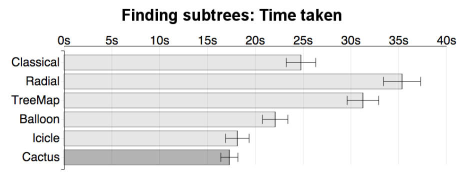

Fig. 4. Time taken (lower is better) for identifying subtrees (**T1**) for each of six layouts. Our *CactusTree* is highlighted in darker bars.

Thus **H1** is confirmed: layouts that use stacking to represent hierarchical structure are better than those that use containment or node-link representations; layouts that tend to preserve shape are better than those that do not. Our hypotheses about the importance of each of these qualities are mostly correct as well. Cactus and Icicle performed best (with no statistically significant difference), followed by Classic, then Balloon and Radial (which showed no statistically significant difference), and finally TreeMaps. Thus, **H1.1–H1.6** are partially supported.

5.2 Results of Experiment 2

Fig. 5 summarizes the results of the comparative analysis of six different methods for path tracing tasks. Again, the bar charts show the mean error values and the standard error from the mean. The omnibus tests for statistical significance showed that there is statistical significance in the mean errors for some of the tasks. The Friedman test showed (χ^2 (5,N=224)=3.250; P<0.001) among six methods. Bonferroni across pairwise Wilcoxon comparisons showed significant differences between TreeMap vs. Cactus (Z=-3.255, P=0.001). Cactus outperforms significantly Icicle (Z=-4.096, P<0.0001). Cactus is also significantly more accurate than Balloon (Z=-4.216, P<0.001). Cactus is ranked as the best technique because it also has significantly better performance compared to Classic (Z=-2.263, P=0.024). Radial is ranked as the second best technique:

TreeMap vs. Radial (Z=-2.967, P=0.003) showed significant difference. Radial vs. Balloon is also finds that Radial is significantly more accurate than Balloon. Classic ranked third (Z=-2.449, P=0.014), while Balloon Layout, TreeMap, and Icicle are ranked as the worst techniques for **T2** regarding percent correct.

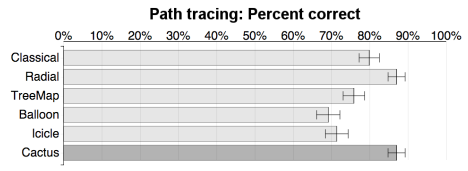

Fig. 5. Percent Correct (higher is better) for path tracing (**T2**) for each of six layouts. Our *CactusTree* is highlighted in darker bars.

Fig. 6 summarizes the results of the comparative analysis of the six different layouts in terms of the time taken to complete the path tracing task. The Friedman test showed significant difference (χ^2 (5,N=192)=26.947; P<0.0001) among six methods. Pairwise comparisons showed significant differences between Cactus vs. Balloon (Z=-3.935,P<0.0001), Cactus vs. Classic (Z=-5.017,P<0.001), and Cactus vs. Icicle (Z=-2.849,P=0.004). However, based on Bonferroni adjustments, other pairwise comparisons did not reveal any significant results. We can therefore conclude that the *CactusTree* layout is significantly faster than all other methods for **T2**.

Thus **H2** is partially confirmed: layouts that use stacking to represent hierarchical structure are better than those that use containment or node-link representations. Our hypotheses about the ranking is only partially correct. Most notably, Cactus performs significantly better than all other layouts, in terms of both time and accuracy. However, our rankings of the others layouts was not as expected. In terms of accuracy, Radial performed second best, followed by Classic and Icicle, with no significant difference between them. The other two layouts performed poorly, with no significant difference between them. Thus, **H2.2–2.5** are not supported. Notably, Icicle was close to the bottom ranking, despite sharing many similar features as Cactus. We believe that this is largely due to the tendency of Icicle to introduce very narrow leaf nodes which are hard to distinguish from each other, especially when edge bundling is used to show connectivity between them.

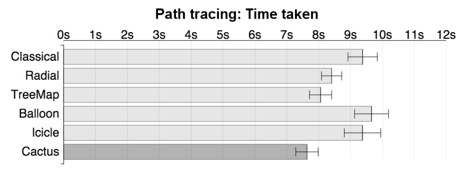

Fig. 6. Time taken (lower is better) for path tracing (**T2**) for each of six layouts. *CactusTree* is highlighted in darker bars.

We also compared the results of the Cactus, Icicle, and Radial layouts across the four different datasets, which had different levels of complexity, from C1 (simplest) to C4 (most complex). For Cactus, the Friedman test did not show any significant differences among all levels of complexities. That is, the Cactus layout performed equally well across all levels of complexity. However, the Friedman test showed significant differences among different levels of complexity for Icicle (χ^2(3,N=56)=22.105; P<0.001). Wilcoxon pairwise comparisons showed significant differences between C3 vs. C1 (Z=-

2.324, P¡0.001), C3 vs. C2 (Z=-2.744, P=0.006), and C4 vs. C3 (Z=-3.530, P=0.001). That is, the accuracy of Icicle tends to get worse as the data becomes more complex. Similarly, the Friedman test showed a significant differences among different levels of complexity for Radial ((χ^2(3,N=56)=27.370; P<0.001). Wilcoxon pairwise comparisons showed significant differences between C3 vs. C1 (Z=-3.578, P<0.001), C3 vs. C2 (Z=-3.411, P=0.001), and C4 vs. C3 (Z=-3.273, P=0.001). As with Icicle, the accuracy of Radial tends to get worse as the data becomes more complex. The other layouts showed no significant differences between the different levels of complexity.

	Classic	Radial	TreeMap	Balloon	Icicle	Cactus
T1	3	4	6	5	1	1
T2	4	2	5	6	3	1

Table 2. Average user preferences on different tree layouts for the two tasks: finding subtrees (**T1**) and path tracing (**T2**), 1 = best, 6 = worst. Icicle and Cactus are both the best for **T1**.

5.3 Qualitative Responses

In addition to recording the participants' time and accuracy of questions related to each of the tasks, we also solicited qualitative responses about each of the layouts. At the end of the study, each user was invited to rank the layouts and to offer comments. User preferences for the two tasks in our study in Section 4 are shown in Table 2.

Despite the general familiarity with the Classic tree layout by the participants, they had mixed responses to it. Some participants indicated that it was among the best techniques with which to identify subtrees, but others noted that it was hard to trace paths between nodes on the tree.

All participants mentioned that they liked the patterns created by the TreeMap layout, but most also indicated that it was frustrating to find structural patterns and that it was confusing to trace connections between the nodes using the TreeMap. This was especially when the links inadvertently crossed the center of a node, as it was hard to tell which level was being indicated. As one user noted, "sometimes it seems as if the connections are connected through an intermediate step." One exception was a user who rated the TreeMap highly despite the visual clutter, saying that "it just seemed easier to recognize connections, even when the data was messy."

Participants also mentioned that they were drawn to the Radial layout, but that, as one user said, "if there were too many layers, it became very complicated to see if the subtrees were in there"— A sentiment echoed by most users. On the other hand, the Radial layout was evaluated much more favorably for the path tracing task. The Balloon layout was perceived the most negatively by participants, as it was acknowledged by all participants that they were difficult to interpret for the more deeply nested trees.

CactusTree and Icicle Plots were described most positively by participants. They found both of these techniques to be more "intuitive" to understand, but some noted that it was harder to read the Icicle Plots in some cases when the density of the tree caused nodes to be narrow: "It was hard to read the connections in the normal [Classic] tree, because the lines seemed stitched together, and the same thing happens with the Icicle tree." Participants were prone to find unintentional patterns in the CactusTree layout, noting that some trees looked, variously, like "snowmen," "clouds," "animals," or "a face." However, this didn't seem to present a distraction, as nearly all participants found it easy to identify subtree patterns in this layout and indicated that it was the most straightforward technique with which to trace paths between nodes.

6 Conclusions and Future Work

The results of our user study indicate that layouts that exhibit particular features aid users when working with complex hierarchies that can be found in real-world scientific datasets. Specifically, we found that a hierarchical structure recognition task was best enabled by trees that used stacking and that preserved the shape of the data representation when contextualized within different datasets. Further, we showed that a layout that used wider turns for edge bundling performed better than a range of other layouts for reasoning about connectivity across complex hierarchies. Our classification of trees using these descriptive features serves as a preliminary guideline for visualization designers to determine if a particular layout is useful for a task, and also to help guide the development of new techniques for the visual analysis of ontologies and linked data.

For future work, we plan to conduct more extensive studies of HEB on different tree layouts and to investigate more involved tasks. For instance, we want to explicitly examine a user's understanding of high level inter-cluster connectivity trends by asking the user to identify which cluster/parent node is most strongly connected to a selected cluster/parent node. We also plan to examine user understanding of low level intra-cluster connectivity trends by testing how well a user can identify the connectivity within a cluster/parent node. Additionally, we also plan to explore how interactions, such as rotating, panning, and zooming, support these tasks (especially for deeply nested trees of up to one hundred levels).

CactusTree [4] is implemented in Javascript using the D3.js library [2]. A demonstration video, the online *CactusTree* application, and evaluation materials, such as screenshots of techniques the test participants saw, questions they were asked, and the actual user study itself are all available on our project page, located at `http://cactustrees.github.io`. Examples of complex, real-world datasets discussed in this paper (along with additional datasets) can also be found on our project page, represented both using *CactusTree* and as compared to a range of other tree layouts.

Acknowledgements

This work was funded in part by DARPA under ARO contract W911NF-14-1-0395.

References

1. B. Bach, N. H. Riche, C. Hurter, K. Marriott, and T. Dwyer. Towards unambiguous edge bundling: Investigating confluent drawings for network visualization. *IEEE Transactions on Visualization and Computer Graphics*, 23(1), 2016.
2. M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.
3. T. N. Dang, H. Cui, and A. G. Forbes. MultiLayerMatrix: Visualizing large taxonomic datasets. In *Proceedings of the 7th EuroVis Workshop on Visual Analytics (EuroVA)*, pages 55–59, Groningen, Netherlands, June 2016.
4. T. N. Dang and A. G. Forbes. CactusTree: A tree drawing approach for hierarchical edge bundling. In *Proceedings of the 10th IEEE Pacific Visualization Symposium (PacificVis)*, Seoul, Korea, April 2017.
5. T. N. Dang, N. Franz, B. Ludäscher, and A. G. Forbes. ProvenanceMatrix: A visualization tool for multi-taxonomy alignments. In *Proceedings of the ISWC Workshop on Visualization and User Interfaces for Ontologies and Linked Data (VOILA)*, volume 1456, CEUR Workshop Proceedings, pages 13–24, Bethlehem, Pennsylvania, October 2015.
6. T. N. Dang, P. Murray, J. Aurisano, and A. G. Forbes. ReactionFlow: An interactive visualization tool for causality analysis in biological pathways. *BMC Proceedings*, 9(6):S6, 2015.
7. T. N. Dang, P. Murray, and A. G. Forbes. PathwayMatrix: Visualizing binary relationships between proteins in biological pathways. *BMC Proceedings*, 9(6):S3, August 2015.
8. T. N. Dang, P. Murray, and A. G. Forbes. BioLinker: Bottom-up exploration of protein interaction networks. In *Proceedings of the 10th IEEE Pacific Visualization Symposium (PacificVis)*, Seoul, Korea, April 2017.
9. T. N. Dang, N. Pendar, and A. G. Forbes. TimeArcs: Visualizing fluctuations in dynamic networks. *Computer Graphics Forum*, 35(3):61–69, June 2016.
10. J. Fekete, D. Wang, N. Dang, A. Aris, and C. Plaisant. Interactive poster: Overlaying graph links on treemaps. In *Proceedings of the IEEE Symposium on Information Visualization Conference Compendium (InfoVis 03)*, pages 82–83, 2003.
11. N. M. Franz, N. M. Pier, D. M. Reeder, M. Chen, S. Yu, P. Kianmajd, S. Bowers, and B. Ludäscher. Taxonomic provenance: Two influential primate classifications logically aligned. *ArXiv e-prints*, Dec. 2014.
12. M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.
13. C. E. Hinchliff, S. A. Smith, J. F. Allman, J. G. B//urleigh, R. Chaudhary, and et al. Synthesis of phylogeny and taxonomy into a comprehensive tree of life. *Proceedings of the National Academy of Sciences*, 112(41):12764–12769, 2015.
14. D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.
15. D. Holten and J. J. van Wijk. A user study on visualizing directed edges in graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2299–2308, 2009.
16. A. Kobsa. User experiments with tree visualization systems. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 9–16, 2004.
17. J. B. Kruskal and J. M. Landwehr. Icicle plots: Better displays for hierarchical clustering. *The American Statistician*, 37(2):pp. 162–168, 1983.
18. B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, pages 1–5, 2006.

19. P. Legendre. Species associations: The Kendall coefficient of concordance revisited. *Journal of agricultural, biological, and environmental statistics*, 10(2):226–245, 2005.
20. A. Lex, C. Partl, D. Kalkofen, M. Streit, S. Gratzl, A. M. Wassermann, D. Schmalstieg, and H. Pfister. Entourage: Visualizing relationships between biological pathways using contextual subsets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2536–2545, Dec. 2013.
21. F. McGee and J. Dingliana. An empirical study on the impact of edge bundling on user comprehension of graphs. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 620–627, 2012.
22. M. J. McGuffin and J.-M. Robert. Quantifying the space-efficiency of 2d graphical representations of trees. *Information Visualization*, 9(2):115–140, June 2010.
23. M. Nasir, O. Hoeber, and J. Evermann. Supporting ontology alignment tasks with edge bundling. In *Proceedings of the 13th International Conference on Knowledge Management and Knowledge Technologies*, pages 11:1–8, 2013.
24. P. Neumann, S. Schlechtweg, and S. Carpendale. ArcTrees: Visualizing relations in hierarchical data. In *Proceedings of the Seventh Joint Eurographics / IEEE VGTC Conference on Visualization*, EUROVIS'05, pages 53–60, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.
25. H. C. Purchase. The effects of graph layout. In *Proceedings of the Australasian Conference on Computer Human Interaction*, OZCHI '98, Washington, DC, USA, 1998.
26. J. Rosindell and L. J. Harmon. OneZoom: A fractal explorer for the Tree of Life. *PLoS Biology*, 10(10):e1001406, 2012.
27. H.-J. Schulz. Treevis.net: A tree visualization reference. *Computer Graphics and Applications, IEEE*, 31(6):11–15, Nov 2011.
28. H.-J. Schulz, Z. Akbar, and F. Maurer. A generative layout approach for rooted tree drawings. In *IEEE Pacific Visualization Symposium (PacificVis)*, pages 225–232, 2013.
29. S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, December 1965.
30. B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on graphics (TOG)*, 11(1):92–99, 1992.
31. L. Strömbäck and P. Lambrix. Representations of molecular pathways: An evaluation of SBML, PSI MI and BioPAX. *Bioinformatics*, 21(24):4401–4407, 2005.
32. F. Viégas, M. Wattenberg, J. Hebert, G. Borggaard, A. Cichowlas, J. Feinberg, J. Orwant, and C. Wren. Google+ ripples: A native visualization of information flow. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1389–1398, 2013.
33. Y. Wang, S. T. Teoh, and K.-L. Ma. Evaluating the effectiveness of tree visualization systems for knowledge discovery. In *Proceedings of the Eighth Joint Eurographics / IEEE VGTC Conference on Visualization*, pages 67–74, 2006.
34. F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.
35. K. Xu, C. Rooney, P. Passmore, D.-H. Ham, and P. H. Nguyen. A user study on curved edges in graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2449–2456, 2012.
36. H. Zhou, P. Xu, X. Yuan, and H. Qu. Edge bundling in information visualization. *Tsinghua Science and Technology*, 18(2):145–156, 2013.