PaisleyTrees: A Size-Invariant Tree Visualization

Katayoon Etemad[1], Dominikus Baur[1], John Brosz[1], Sheelagh Carpendale[1] and Faramarz F. Samavati[1]

[1]University of Calgary

Abstract

Squeezing large tree structures into suitable visualizations has been a perennial problem. In response to this challenge, we present PaisleyTrees, a size-invariant tree visualization. PaisleyTrees integrate node-of-interest focus with tree-cut presentations to support rapid tree navigation without resorting to zooming and panning. This visualization offers the ability to work with trees of arbitrary depth and breadth, and maintains legibility for displayed elements. These advantages are achieved by using a hybrid layout, inspired by traditional Paisley patterns, that combines node-link, nested and adjacency-based tree layout techniques, and offers both depth and breadth elision.

Keywords: Information Visualization, Tree Layout, Hybrid Layout, Mobile, Graphs.

1. Introduction

Modern computers are increasingly efficient at storing and processing large data-sets. Despite this, visualization of large data-sets is still a challenging task. This is apparent with visualizations of tree data structures, as can be seen with linguistic and genealogical hierarchies. As information must eventually be read, extracted and understood by a human, an attractive and aesthetic visualization of large trees can prove vital in performing this challenging task.

While node-link diagrams (Battista [1]) are a valuable method for tree visualization, an informative and aesthetically pleasing arrangement of nodes and edges is required. In addition, the scalability of these types of diagrams to large trees remains a challenge. As a tree grows, node-link diagrams do not necessarily provide the best solution. At some point, zooming and panning operations start to take up a large share of all interaction. As an alternative, treemaps [2] usually come with fixed screen size requirements, but they adjust to growing trees by shrinking their internal components. The problem of scalability is particularly acute on touch-enabled displays and high-resolution, small-screen devices. Keeping the problem of increasingly large trees on small-screen devices in mind, we explore the creation of a tree visualization with several attributes: (1) an aesthetically pleasing structure using a layout inspired by Paisley patterns, (2) support for trees of arbitrary breadth and depth by using an appropriate tree-cut, (3) a resulting visualization that is size-invariant and adjustable for various screen sizes from mobile to desktop, and (4) support for the common task of tree traversal through quick navigation. In this paper we introduce a novel tree visualization called PaisleyTree (see Figure 1).[1] To keep PaisleyTrees size-invariant while supporting arbitrarily large trees, we use a tree-cut based on a hybrid layout that combines aspects of node-link, adjacency and nesting layouts.

The remainder of this paper is structured as follows. An overview of relevant previous research is provided in Section 2. The design of PaisleyTrees, including the layout, visualization, interaction and implementation is explained in Section 3. In Section 4, we discuss our design decisions and rationales. In Section 5 we provide some examples, and Section 6 contains a discussion. We conclude the paper in Section 7.

2. Related Work

Tree visualizations usually fall into one of two categories: they show the connections between nodes either *explicitly* by drawing edges or *implicitly* by using the position of node elements [4] (see Figure 2).

Explicit approaches include the most common tree layouts: the node-link diagram and its variations. The main methods used to reduce the display footprint involve changing the layout algorithm (e.g., [5, 6]), adjusting the representation of edges (e.g., *Cheops* [7]), or using the third dimension (*ConeTrees* [8]).

*Corresponding author. Email: publications@icst.org

[1]An early concept of PaisleyTree was mentioned in GRAND NCE poster by our group presented at GRAND [3].

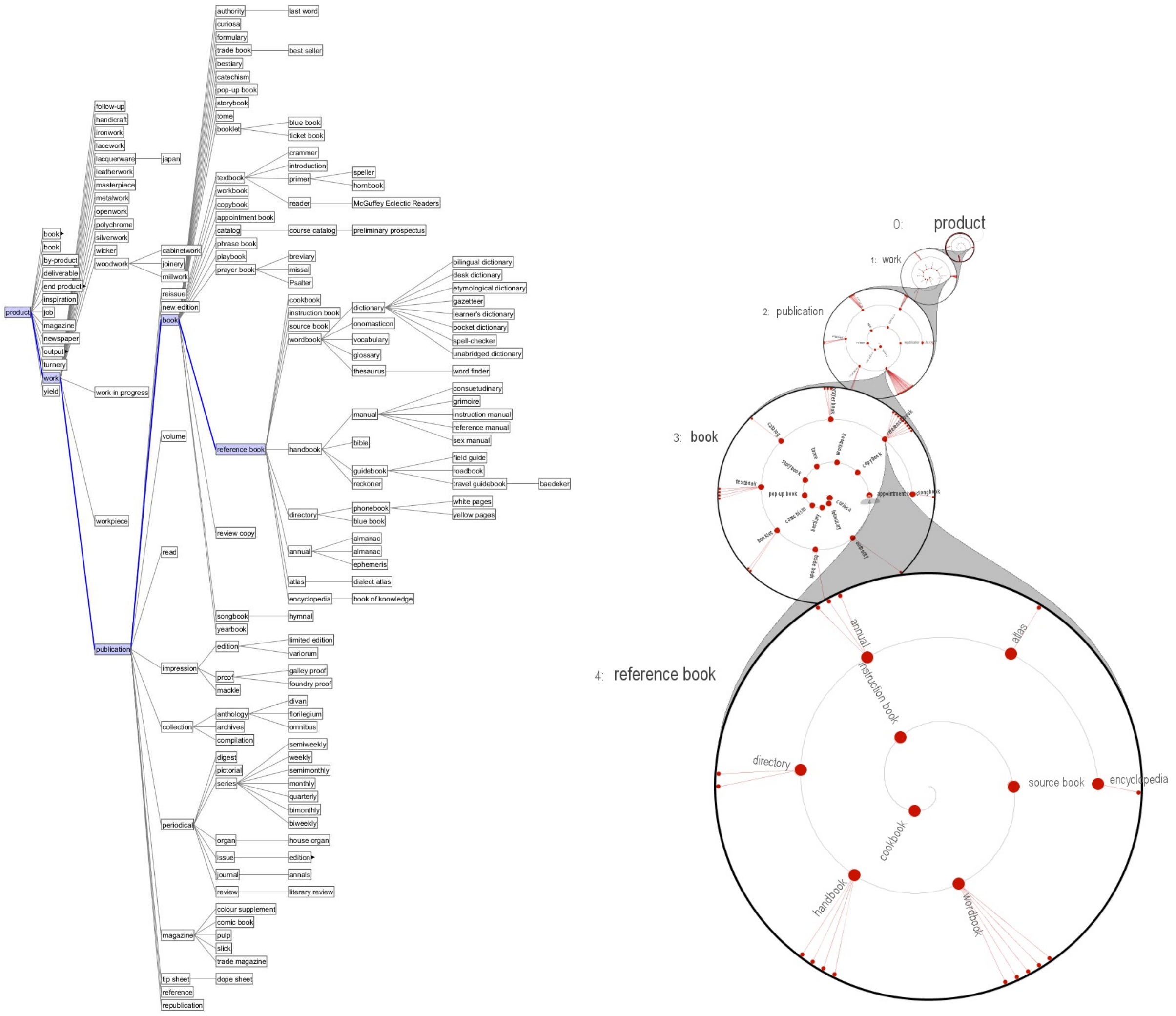

Figure 1. Left: SpaceTree, Right: PaisleyTree. Both trees show the subtree of Wordnet rooted at the word "Product" with 197 nodes and 9 levels.

Neumann et al. increased possible layout size by borrowing optimal node configurations from nature in *Phyllotrees* [9]. Still, all these adjustments cannot prevent the increases in tree size causing a resulting increases in the visualization's space requirements.

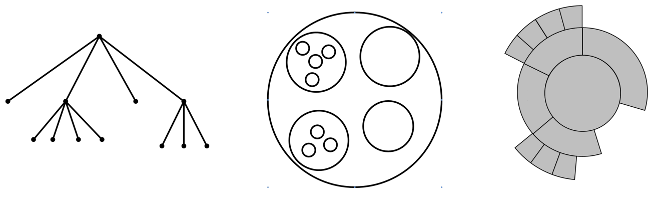

Figure 2. *Tree layouts, Node-link (explicit), Nested (implicit) and Adjacency (implicit).*

Implicit approaches to encoding connections eliminate the need for explicit visual edges. The spatial relationship between two elements sufficiently illustrates in which way they are related. Treemaps [2] and their derivatives (e.g. *Cascaded Treemaps* [10] and *Bubble Trees* [11]) use *nesting* to encode parent-child-relationships. In ShamsehTree [12] a sequence of concentric circles are used to present the ancestral path for a tree-cut. The underlying layout of this tree visualization is designed based on symmetric arrangement of nodes, inspired by Persian floral patterns. Another implicit approach relies on *adjacency* (e.g., *Docuburst* [13], *Icicle Tree* [14], *Sunburst* [15] and *Information Slices* [16]). Implicit techniques are often used when scale is an issue since they save screen space by skipping edge drawing.

The scaling problem of tree visualizations can be partially overcome by other methods. The family of *SpaceTree* [17] and *Degree-Of-Interest Trees* [18, 19] visualize a tree-cut (a connected subset of a the tree), thus using the screen space more efficiently by focusing only on nodes-of-interest. This idea of tree-cut originates in linguistics where thesaurus trees require scale reduction [20]. In Degree-Of-Interest Trees, a specific DOI value is assigned to each node and the size of the node in visualization is a function of that value. DOI value of each node is the importance and its distance from the node of interest. In PaisleyTree we also use a tree-cut which is defined in the proximity of the node of interest (NOI). However, we use a cut from NOI's ancestor nodes up to the root and its descendant nodes down to a specified level. This specific cut is defined to fit to our hybrid aesthetic layout inspired by Paisley patterns and to ease navigation.

Displaying trees in a non-uniform fashion using hybrid layouts provides an excellent opportunity to address the scale challenge. In addition, it can help us to create more pleasing layouts. Examples include *Elastic Hierarchies* [21] and *Linked Treemaps* [22], that use both node-links and treemaps. In Elastic Hierarchies some nodes are presented in treemap layout and connected with links. Linked Treemaps take advantage of the third dimension, presenting the treemap with links added in the 3D space. PaisleyTrees is a 2D layout that uses all three of these approaches — node-link, nesting and adjacency — in one visualization to address both the large scale and aesthetical design challenges.

Finally, the problem of growing visualizations becomes especially obvious on mobile devices with small screens. Mobile visualizations therefore have to take more creative approaches for displaying trees. *Tablorer* [23] converts the underlying tree into a nested table, while *Magic Eye View/Rectangular View* [24] combines distortion with dropping branches of the tree. The *Radial Edgeless Tree* [25] uses a variation of a Voronoi diagram and adjacency. PaisleyTree is a size invariant visualization that works for both large and small screens.

3. PaisleyTrees

In this section, we describe the basics of the PaisleyTrees visualization. The design rationales and possible variations are discussed in Section 4. First, in Section 3.1, we discuss the layout of PaisleyTrees, followed by an overview of the visualization of PaisleyTrees in Section 3.2. We then present a discussion about the interaction ideas in Section 3.3, and give an overview of the implementation in Section 3.4.

3.1. PaisleyTree Layout

According to Roger Scruton, "Styles may change, details may come and go, but the broad demands of aesthetic judgement are permanent." [26] One of the objectives of our visualization is to use an aesthetically pleasant layout. However, the challenge is how to design such a layout whilst meeting our other visualization objectives (see Section 1). In fact "Aesthetics can be considered an added bonus, or a by-product when striving for readability and effectiveness." [27]

Combinations of the node-link, nested and adjacency layouts can be used for designing an aesthetically pleasing layout (see Figure 2). Node-link layout relies on drawing edges in an explicit fashion, which tends to lead to clutter. In contrast, nested and adjacency layouts avoid this by implicitly representing edges. One method for handling a large tree visualization is to consider a node-of-interest (NOI) to be the center of attention. This could be accomplished by shrinking ancestor and descendant nodes as they get further from the NOI (see Figure 3a). By definition, one characteristic of nested layouts is that the largest nodes are the further ancestor nodes. In our design, we use adjacency layout for the NOI and its ancestral nodes to the root, and nested and node-link layouts for descendant subtrees of the NOI.

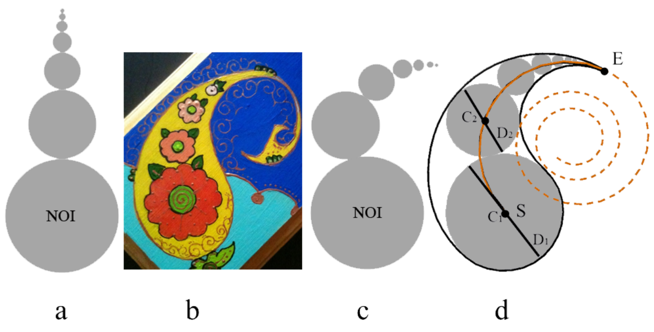

Figure 3. *(a) Shrinking ancestral chain. (b) A traditional Paisley pattern. (c) The ancestral chain arranged on a spiral curve. (d) The node circles filling the Paisley outline.*

As a strategy to create an aesthetic design, we have looked at many patterns used in the traditional arts from different cultures. We chose Paisley patterns used in traditional Persian and Indian designs (see Figure 3b). The main motivation behind our use of this leaf-like pattern is that, as they shrink, nodes further away from the NOI fit nicely within the Paisley outline, which follows a spiral curve (see Figure 3c). For our implementation, we use Fermat's spiral [28] in polar coordinates

$$r = k\sqrt{\theta}, \quad \theta_0 \le \theta \le \theta_1$$

where k is a constant controlling the curvature of the spiral. As shown in the Figure 3d, we denote the start and the end points of the spiral by S and E. To arrange the nodes on the spiral, we represent each node with a circle, denoted by $n_1, n_2, n_3, \ldots$. In this arrangement the node n_{i+1} is tangent to the node n_i and has a smaller size. Each circle n_i is specified by (C_i, D_i) where

C_i is the center and D_i is the diameter of the circle. The basic idea is that the sum of diameters roughly approximate the arc-length L of the spiral from S to E. We use a numerical approximation for finding L using the sum of small lines formed by a discrete sample of the curve's points. The diameter of the first circle ($D = D_0$) is usually known, and we need to find the diameter of the other circles. Since the circles are shrinking, we need to calculate their contraction factor α. Thus, the sum of the diameters (except the NOI circle which is 1/2 of D_0, as shown in Fig 3d) is approximately the same as L:

$$\frac{1}{2}D + \alpha D + \alpha^2 D + \ldots + \alpha^{n-1} D = L. \tag{1}$$

To determine α we find a numeric solution to the non-linear Equation 1. Notice that an approximate value for α is sufficient.

Now we can determine the distribution of circles on the spiral. To do this, we need to know the position of each circle. Note that the first circle is positioned at $C_1 = S$ and its radius is $R_1 = \frac{D}{2}$. The second circle has contraction factor α and must be tangent to the first circle as shown in Figure 3d. Thus, the arc-length of the spiral from C_1 to C_2 (the center of the second circle) must be $R_1 + R_2$ where $R_2 = \alpha R_1$. Consequently, we find C_2 by traveling the distance of $R_1 + \alpha R_1$ along the spiral from C_1. The centers of the other circles are determined similarly.

3.2. PaisleyTree Visualization

PaisleyTree (see Figure 1 right) is our constructed tree visualization based on the above layout and on the requirements outlined in Section 1. To address the challenge of large tree visualization, we use tree-cuts, hybrid layout and elision to make the amount of visual elements manageable.

The main approach of the PaisleyTree is to use a specific tree-cut (i.e. a connected subset of the tree) for achieving our requirements. This tree-cut presents the NOI along with its hierarchically proxemic nodes, including its ancestor nodes up to the root and its descendant nodes down to a specified level. Our implementation includes two levels of descendants from the NOI. The main representation of each node is a circle (as discussed in Section 3.1). Node circles are arranged from bottom to top, with the node-of-interest at the bottom, the root node at the top and intermediate nodes in between. Circles and their contents further up the spiral are diminished in size, reflecting their distance from the NOI.

As shown in the Figure 4, each node circle displays two levels of its descendant sub-tree: all of the node's direct children are placed as dots (nested layout) on a spiral within the node circle while grandchildren

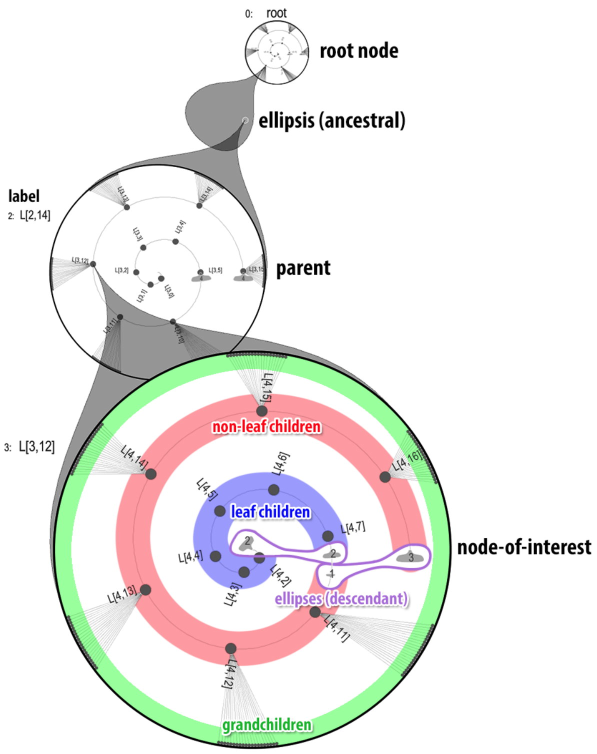

Figure 4. *Visual organization of a PaisleyTree. In each node grandchildren are positioned on the perimeter of the circle. Non-leaf children on the outer ring of the spiral and leaf children are positioned on the inner ring of the spiral.*

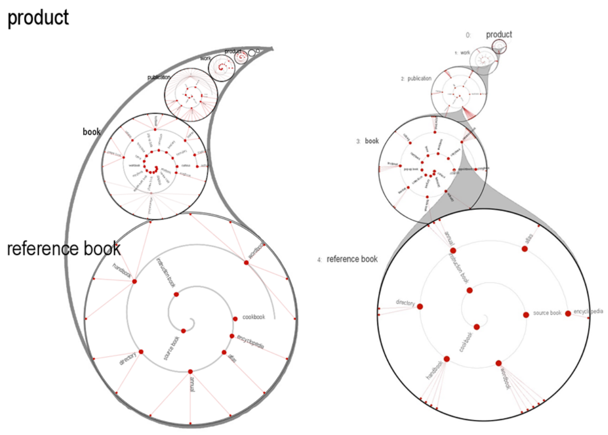

Figure 5. *Left: PaisleyTree with the paisley outline. Right: PaisleyTree without the paisley outline.*

are placed on the perimeter of the node circle (this region is highlighted with green) using node-link layout. To reduce cluttering, the children are divided into two groups, leaf children and non-leaf children. Leaf children are placed on the spiral's inner ring (highlighted with blue), non-leaf children on the outer (highlighted with red). The children are

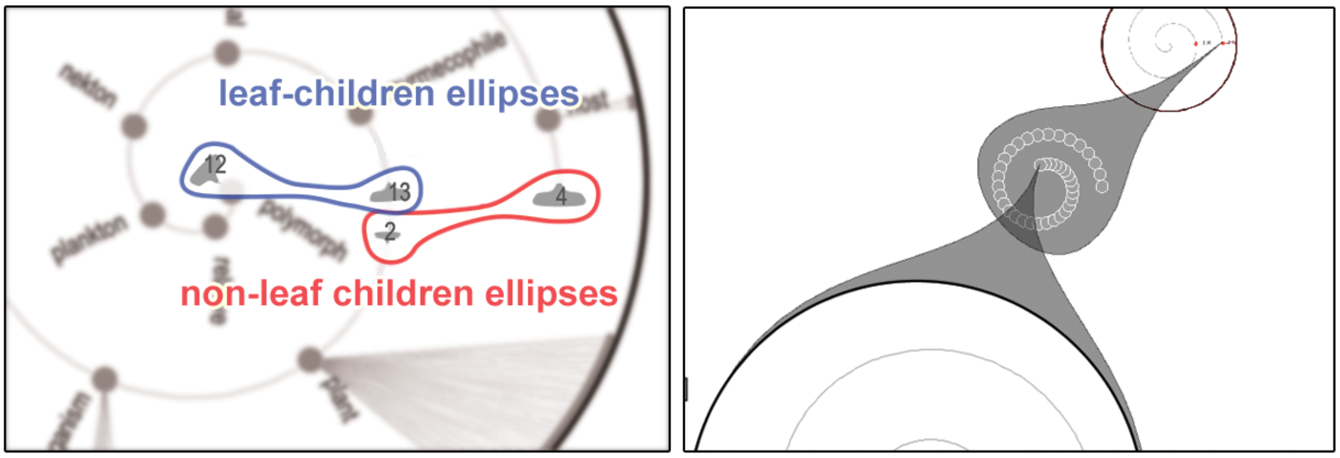

Figure 6. Types of elision in PaisleyTrees: descendant ellipses (left), an ancestral ellipsis (right).

distributed at identical distances to support legibility and interactivity in the available spiral space. All grandchildren are shown on the perimeter of the node circle and arranged close to their respective parent (i.e., a non-leaf child). The perimeter of the circle is divided into equal sections for each set of grandchildren.

Within each node circle the children's labels are displayed in a radial angular orientation. The angle of orientation for each label is perpendicular to the spiral's tangent. Depending on a label's angle, its text direction can change between left-to-right and right-to-left to make it easier to read. Additionally, a pop-out version of the label is shown when hovering over the node.

The use of an outline for PaisleyTree is optional (See Figure 5). Using the outline makes a stronger resemblance to the traditional pattern, but with the price of possible cluttering with labels. The gray cones are removed when the outline is drawn. The gray cones have the advantage of explicitly indicating the node's location in its ancestor.

Each of the intermediary nodes follows the same internal layout although slightly smaller. Each node is an expansion of a child within its parent node. To make this relationship more obvious, each node is connected to its own representation in its parent's node by a slightly transparent grey cone (Figure 1 and 4).

One of PaisleyTree's design goals was to support trees with arbitrary depth and breadth while preserving size and readability. The layout described above, however, breaks for certain cases. If the tree has more levels than there are spaces available for node circles on the Paisley curve, then not all levels can be displayed. Similarly, when the number of children or grandchildren grows large, they are placed very close to one another, eventually becoming unreadable and impossible to interact with. For these reasons, we provide *ellipses*, to act as placeholders for parts of the tree. These ellipses exist in two forms: *ancestral ellipses* (see Figure 6, right) that contain levels of the tree that are not shown as node circles; and two *descendant ellipses* (see Figure 6, left) that contain excess leaf- and non-leaf children.

Ancestral ellipses lie on the Paisley spiral curve, taking the place of one or more node circles, and are displayed as grey shapes pointing towards the origin of the first node circle they contain. All contained nodes are displayed as small circles arranged in a spiral shape. The first contained circle (i.e., the closest to the root) is placed at the right end of the spiral (see Figure 6, right). Descendant ellipses hide excess child nodes within a node circle. They are displayed as grey shapes at the ends of both the leaf- and non-leaf regions of the spiral. Numeric labels indicate the number of contained nodes. Interaction provides access to nodes in either type of ellipses.

3.3. Interaction

In this section we describe interaction with PaisleyTrees in desktop environments. Adaptations for touch-based devices are described in section 5.3. Interaction in PaisleyTrees falls into the two main categories: tree navigation and adjusting ellipses.

A PaisleyTree is initialized as a single node circle, representing the root node (see Figure 7, left). By clicking on one of the non-leaf nodes, that node will expand and make the current NOI (see Figure 7, left to middle). To make the transition comprehensible, the node and its connecting cone grow out of the position from which it was selected using an animated transition (cf. [29]), while the root node itself is shifted up one space. Thus, clicking on nodes allows for quick navigation through the tree. Choosing a child that is a leaf node results in the new NOI having no further descendants. All displayed nodes (including the root node) are interactive and can be selected as the NOI (See Figure 8).

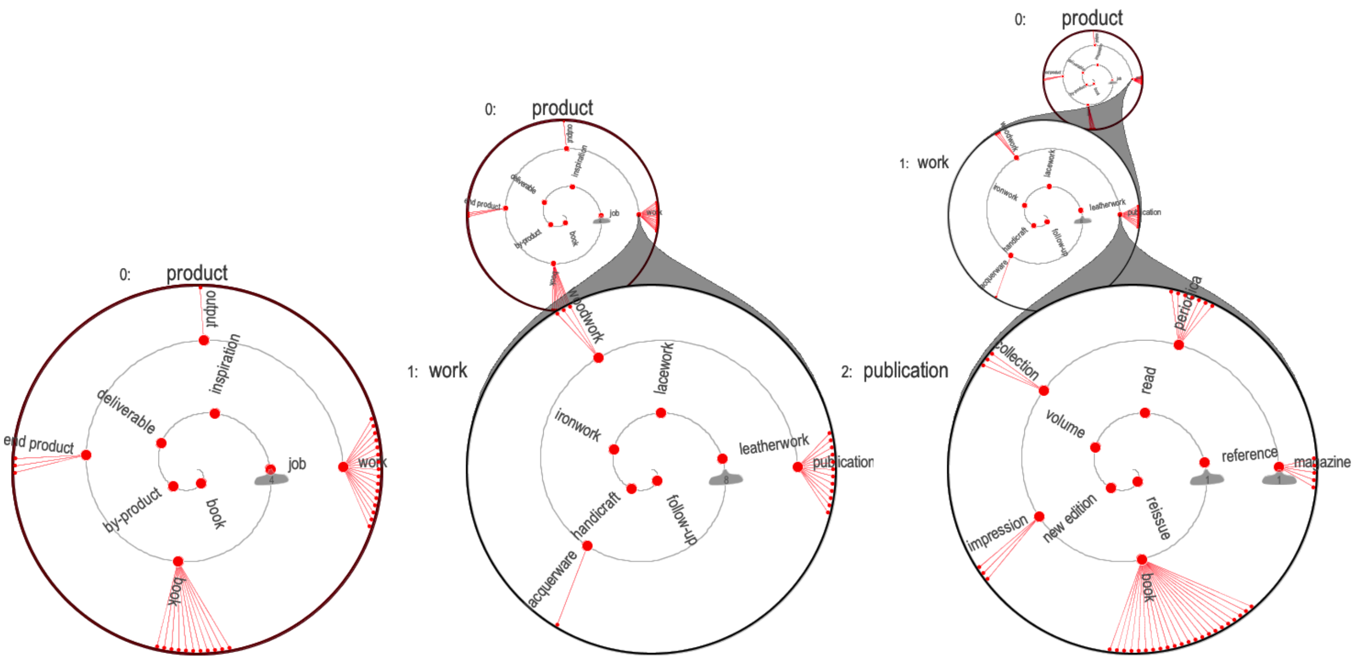

Figure 7. Drilling into a PaisleyTree: Each node circle in a PaisleyTree represents one node, with the bottom-most node representing the current node-of-interest. The node's children are shown along a spiral in the node circle, while grandchildren are shown along the outer perimeter. By consecutively clicking on a sequence of children nodes the most recently selected child node becomes the current node-of-interest and expands into a new node circle, supporting drill down in the tree.

Changing the NOI can also cause the creation or dissolution of an ancestral ellipsis. A limited number of node circles can be displayed within a PaisleyTree at any one time, depending on the on-screen resolution (we commonly use three to seven levels). Every time a node within a node circle becomes the new NOI, all visible node circles shift up one space. Once the limit of visible node circles is reached, the system automatically creates an ancestral ellipsis below the root node and adds the node circle directly below the root to it.

We additionally enforce a limit on the number of children nodes displayed along the spiral within a node circle. When the actual number of children exceeds the limit, the necessary ellipses automatically appear. This prevents overlap and maintains interactivity and legibility. On desktops this fixed maximum can be adjusted, so customization based on screen resolution and preferences is possible.

3.4. Implementation

We implemented the desktop version of PaisleyTrees in Java, relying on Processing as the drawing back-end. We also ported a version of PaisleyTrees to the mobile Android operating system with suitable adaptations for touch input. Rather than requiring a direct hit from a cursor, it returns the closest node within a radius of the touch area.

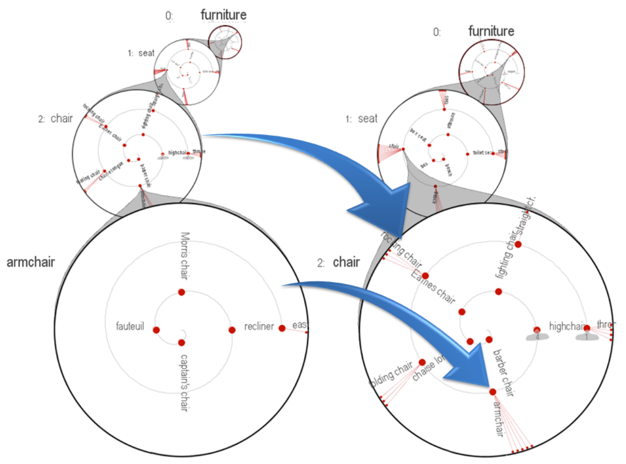

Figure 8. *Clicking on the 'chair' node circle (left) makes it the node of interest (NOI) (right). The current NOI, 'armchair', (left) is reduced to its child position in the 'chair' circle. Ancestral elision is not required here.*

4. Design Rationale

The strict goals of producing a size-invariant visualization technique while keeping visual elements readable required us to take the uncommon approach of combining tree-cuts with a hybrid layout. In this section we provide the background for our design influences and decisions.

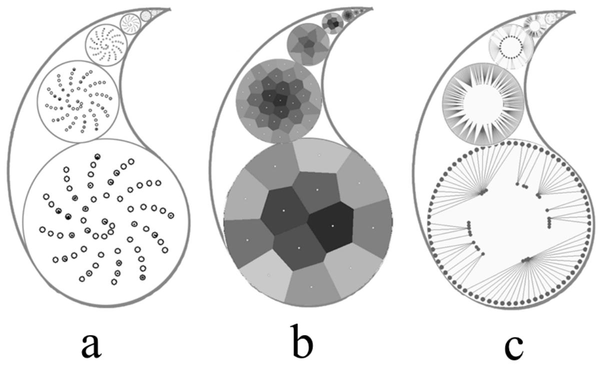

Figure 9. *Different design approaches for layouts of descendant nodes: (a) Phyllotactic, (b) Voronoi and (c) Circle.*

4.1. Hybrid Layout

Combining different layout methods into a hybrid layout incorporates the beneficial aspects of different approaches into one coherent visualization. Existing hybrid approaches (e.g. Elastic Hierarchies [21]) are good examples of this. We arrived at PaisleyTree's hybrid design by considering the advantages and disadvantages of the three main layout approaches and combining them accordingly. For example, nesting methods work well for emphasizing parent-children relationships. We therefore used nesting to represent children and grandchildren inside each node circle. On the other hand, adjacency methods are especially effective at presenting hierarchical structure, so we employed adjacency for the ancestral hierarchy for each selected node. Finally, we use the idea behind node-link layouts to explicitly connect non-leaf children in a node circle with their respective children along the perimeter.

For the NOI, we present its ancestors as a sequence of shrinking circles positioned on the Paisley curve which is part of a spiral (see Section 3.1 for its mathematical derivation). A circle's position on the curve expresses its level in the hierarchy and its distance from the NOI. Therefore, the farther the ancestors are from the NOI, the closer to the apex of the curve they are positioned. The NOI is, by definition, the most interesting node at a given moment, so we mapped the size of ancestor node circles to its distance from the NOI.

Children within node circles are arranged along a spiral to make effective use of the circle's area. As shown in Figure 9 a and b, we considered other arrangements, such as Phyllotactic and Voronoi diagrams, using a recursive nested layout. In these layouts the space assigned to grandchildren is very small. Therefore, we use a node-link layout for children/grandchildren connections. Figure 9 c, shows an early variation of this idea. However, the use of spirals makes the arrangement of nodes more organized and fits better with the traditional form of the Paisley pattern. Spirals exhibit a type of symmetry called *dilational symmetry*, that refers to the symmetry of self-similar elements at different scales. Usually, symmetry creates visual appeal and is known to positively affect information comprehension [30] and can engage and motivate the viewer to devote more time and attention to their information exploration tasks [31]. In addition, the use of spirals provides an effective means for handling elision for descendant nodes. Finally, the ancestral spiral curve and its shrinking circles produce a slight illusion of perspective, as if the circles were shrinking into the distance.

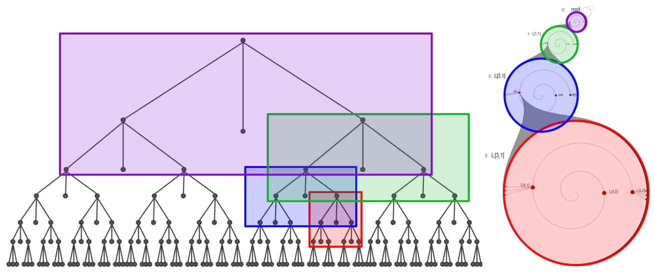

Figure 10. *A tree with 190 nodes and 7 levels in both node-link and PaisleyTree layout. In PaisleyTrees, each node circle shows three levels: a node and all of its children and grandchildren.*

As noted, nodes of the tree are represented redundantly in PaisleyTrees. As visible in Figure 10, nodes can appear repeatedly, either as node circles or as nodes within a node circle. The same node can have up to three concurrent representations in the visualization: as a grandchild, as a child node within the next node circle and as its own node circle. While this concurrent node representation might seem unnecessary, we made the local tree structure explicit with the intention of aiding navigation interaction.

4.2. Use of elision

One of PaisleyTree's goals is to to provide a new alternative for the presentation of trees of arbitrary size. Problems with visualizing large trees arise from two sources: too many children on one level (large breadth) and too many levels (large depth). To keep visual elements at a reasonable size, our strategy was to elide some of the nodes when there are too many in either dimension. Using elision for both the ancestral and descendant dimensions, an arbitrary number of nodes and levels can be implied in the representation - at the cost of increased interaction. Ancestral and descendant ellipses behave differently, in that ancestral ellipses cannot be freely configured. Interaction only allows stepwise upwards or downwards movement, and in the case of too many circles within the elision (see Section 3.3) not every contained circle can be extracted.

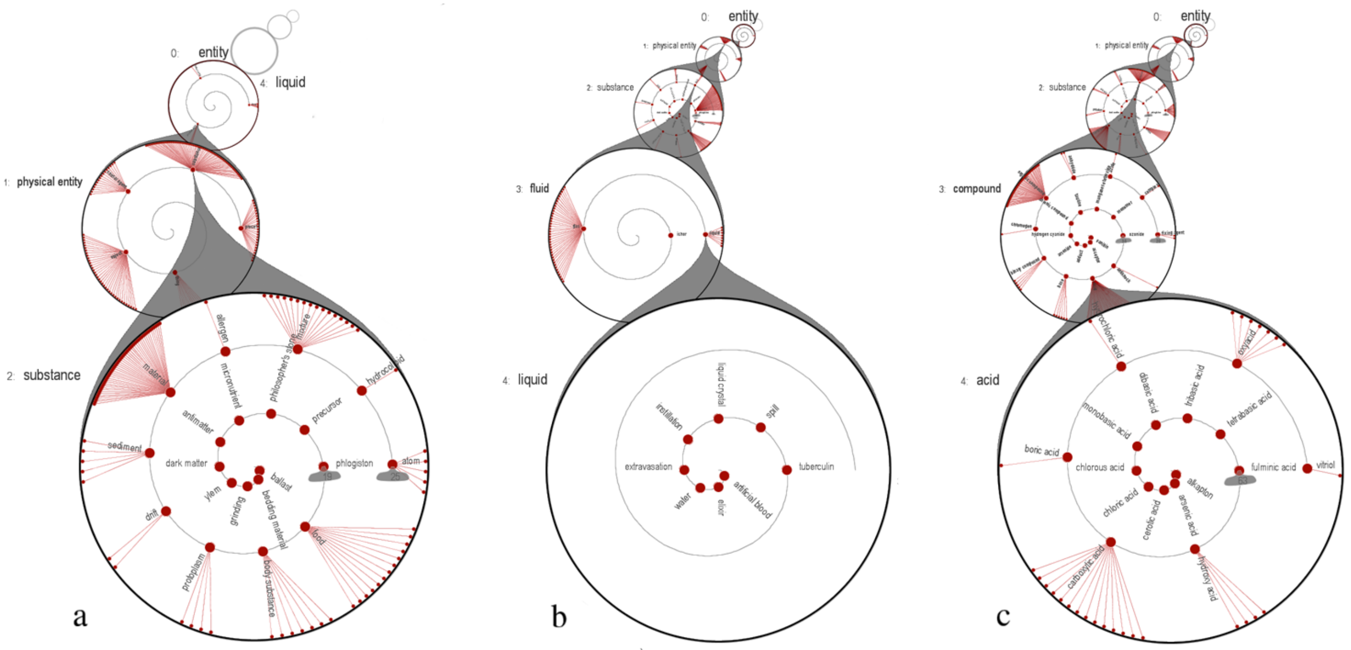

Figure 11. *Select interaction steps in finding the* **hydrochloric acid** *node in the* **entity** *subtree of the WordNet [32] tree structure.*

However, as all node circles within an ellipsis remain reachable, it is still possible to click on one and make it the NOI. The situation is different for descendant ellipses, where the only way to reach a hidden node is through bringing other sibling nodes into the ellipsis. The circular dragging gesture that is required can be repeated indefinitely which allows relatively fast access and also works well on touch-based devices.

5. Examples

In this section, we describe three applications that demonstrate the utility of PaisleyTrees: an exploration of Word Net, a file tree browser for mobile devices, and location selection in mapping and GPS applications.

5.1. Exploration of Word Net

The following provides a brief example of how the PaisleyTree node-focused approach to tree traversals functions when traversing the *entity* (root) subtree of the WordNet [32] tree structure.

Consider Bob, who would like to investigate the *hydrochloric acid (HCl)* node, but is unsure where this node is located. He begins at the root *entity* and traverses down through the nodes, shifting the NOI to *physical entity* and then to *substance* (Figure 11a). Children of *substance* include *chemical irritant*, *fluid*, and *medium*, all of which Bob suspects may lead to *HCl*. He first selects *fluid* which, leads to *liquid*, whose twelve children do not include *HCl* (Figure 11b). Realizing that he has chosen an incorrect *substance* he would like to return to the *substance*, node. To do this, Bob clicks on the *substance* ancestor (two nodes above the current NOI in Figure 11b), returning to this node as the NOI (Figure 11a), allows him to choose a different child. This time Bob selects *compound*, leading him to *acid*, and then to his target, *hydrochloric acid* (Figure 11c).

When searching for nodes that have not been visited previously, a return to the upper nodes of the PaisleyTree is helpful after traversing an incorrect branch. Also, alphabetical sorting of child nodes may ease searching for specific nodes that may or may not be present.

5.2. Location Selection

A task commonly faced on mobile devices such as GPS systems and smartphones is entering a target location. The common approach is to use an on-screen, touch-based keyboard which can be very small, depending on the size of the device. Also, GPS devices often rely on long, paginated lists of potential target locations, adding further interaction overhead. A different approach to target selection is to present targets in a large tree based on location hierarchy: from countries through to states/provinces to cities, streets, houses, etc. The resulting tree for this approach is naturally quite broad, which makes it problematic for common tree visualizations.

To test the capabilities of PaisleyTrees we built such a tree from countries to cities via intermediate states/provinces. The resulting tree has a total of 334,681 city nodes that are either directly below a country node (independent cities/regions) or children of states/provinces. We added two more layers

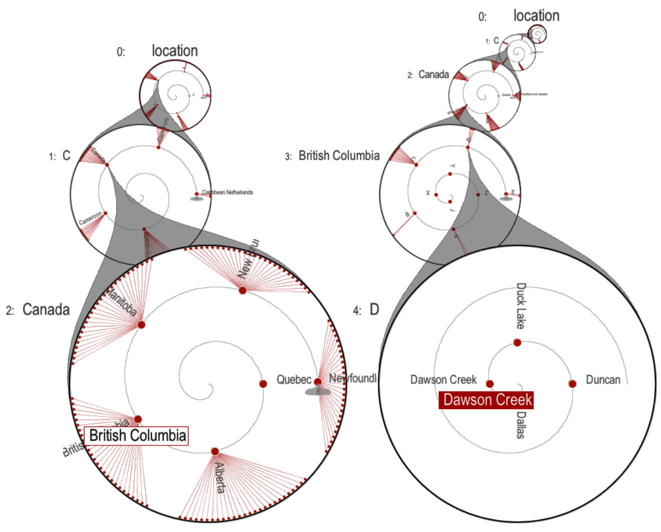

Figure 12. Selecting a location by navigating through a country-state-city tree.

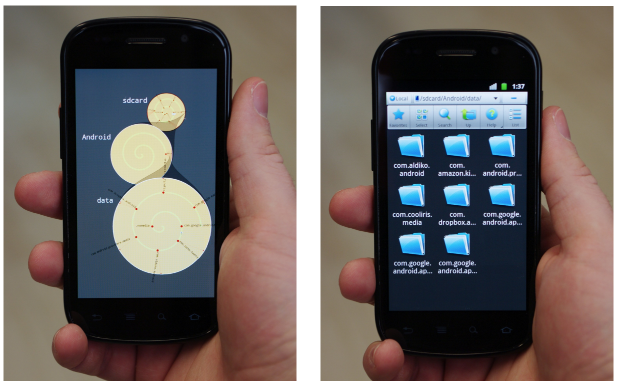

Figure 13. Comparison between a file system as PaisleyTree (left) and a popular mobile file tree browser (right).

containing the 26 letters of the Latin alphabet to make navigation in longer lists easier (regions such as 'England' in Great Britain have more than 1700 cities and towns). One alphabet layer was added above the country level and another one above the city level. Figure 12 shows an illustrative navigation from Canada, through British Columbia, to the town of Dawson Creek. In this scenario, the leaf node for Dawson Creek is the navigation target and would be selected.

5.3. Mobile File Browser

PaisleyTrees have several useful features that are suited for file browsing: fixed screen size prevents the need for zooming and panning; there exists a clear division between leaf-nodes (files) and non-leaf nodes (subdirectories); easy access to ancestor nodes and the root supports easy navigation to higher level directories; and the presentation of children and grandchildren for each node assists in the search for useful subdirectories.

The ability of PaisleyTrees to work entirely within strict size limitations makes PaisleyTree file browsing ideally suited for phones with touch displays. Figure 13 provides a comparison between the popular Android-based file browser ES File Explorer (http://www.estrongs.com/en/products/es-file-explorer.html) with our Paisley mobile file browser. The Paisley file browser is able to show multiple levels of the ancestral directory structure.

Selection and elision panning operations within the NOI are within reach of a thumb at the bottom of the display, which makes it ideally suited for one-handed interaction.

A fully-functional Paisley file browser requires additional interface elements to enable the full range of file system operations, such as sorting and file manipulation operations (i.e., creating, moving, deleting, executing). Additionally, some means of visually distinguishing empty subdirectory leaf nodes from file nodes would be beneficial (e.g., using icons instead of plain circles).

6. Discussion

PaisleyTrees represent an uncommon approach to the task of tree visualization. While they incorporate ideas from hybrid layout systems (e.g., [21]) and tree-cut based visualizations (e.g., [17]), the overall result is decidedly different. In this section we discuss and compare PaisleyTrees to other, more traditional tree visualizations.

In order to emphasize the differences between PaisleyTrees and existing visualization techniques we compared it to two other representative tree layouts. As an example of an adjacency-based layout, we used a radial space-filling layout, similar to the Sunburst technique [15] as implemented in the Lark system [33]. This radial space-filling method shrinks visual elements to cope with increased size, similar to other implicit visualization techniques such as Treemaps. Growth happens outwards for each new level of the tree. A second comparison technique is SpaceTree [17]. SpaceTrees, which also work with tree-cuts, rely on a more conventional node-link-based layout. While they save space by only displaying parts of the tree, once a sufficient number of branches has been expanded, space requirements are comparable to a regular node-link diagram. They share, however, one of the advantages of PaisleyTrees in that they allow rapid navigation of a path through the tree.

Figure 14 shows all three systems, each displaying two different trees. The top row displays the *product* tree, which contains 197 nodes and a total of nine levels, while the bottom row shows the *entity* tree, which

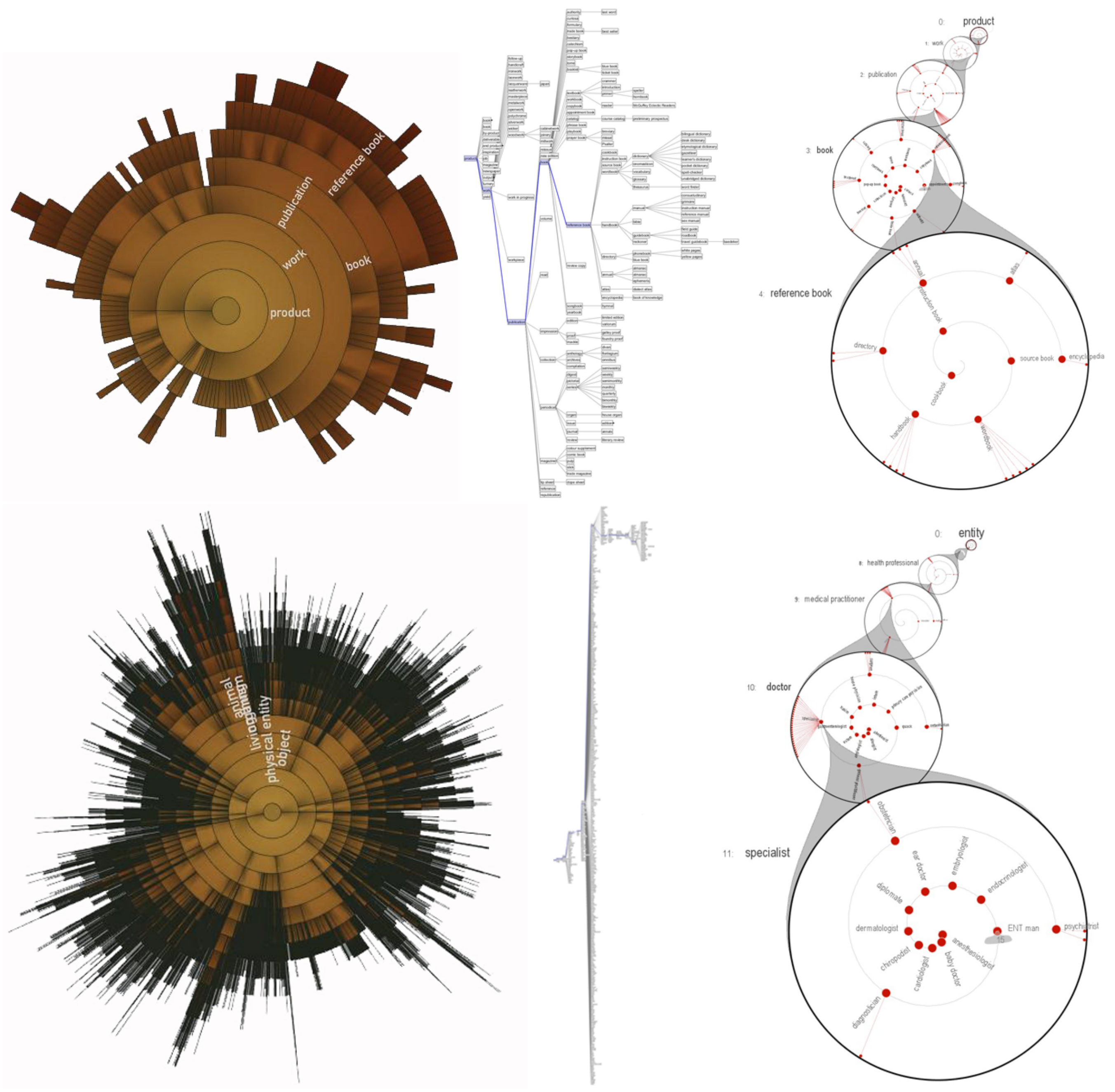

Figure 14. Comparison of three tree visualization techniques for a small tree (top: 197 nodes, navigated to level 4) and a large tree (botton: 75,000 nodes, navigated to level 11). Left: Sunburst/Lark [15, 33]. Middle: SpaceTree [17]. Right: PaisleyTrees.

is considerably larger, holding 75,000 nodes with 18 levels of depth. Visually, comparing the results for these three tree types clearly demonstrates the differences in approach between these three tree visualization techniques. Notice that SpaceTree examples have been created through its default setup. The arrangement can be improved interactively by using zoom/pan and wrap layout tool. The biggest difference between the radial layout and the other two techniques is its goal of displaying the complete tree instead of a tree-cut. This allows the technique to answer structural queries and provide an idea of the overall structure of the data without interaction. The approach, however, breaks down once the tree reaches a certain size: while all 197 nodes of the smaller *product* tree are somewhat visible and shown surrounded by their neighbors, in the visualization of the much larger *entity* tree many nodes become so small as to be virtually unreadable. The Lark implementation only displays labels once a tree element has been clicked, which becomes impossible along the

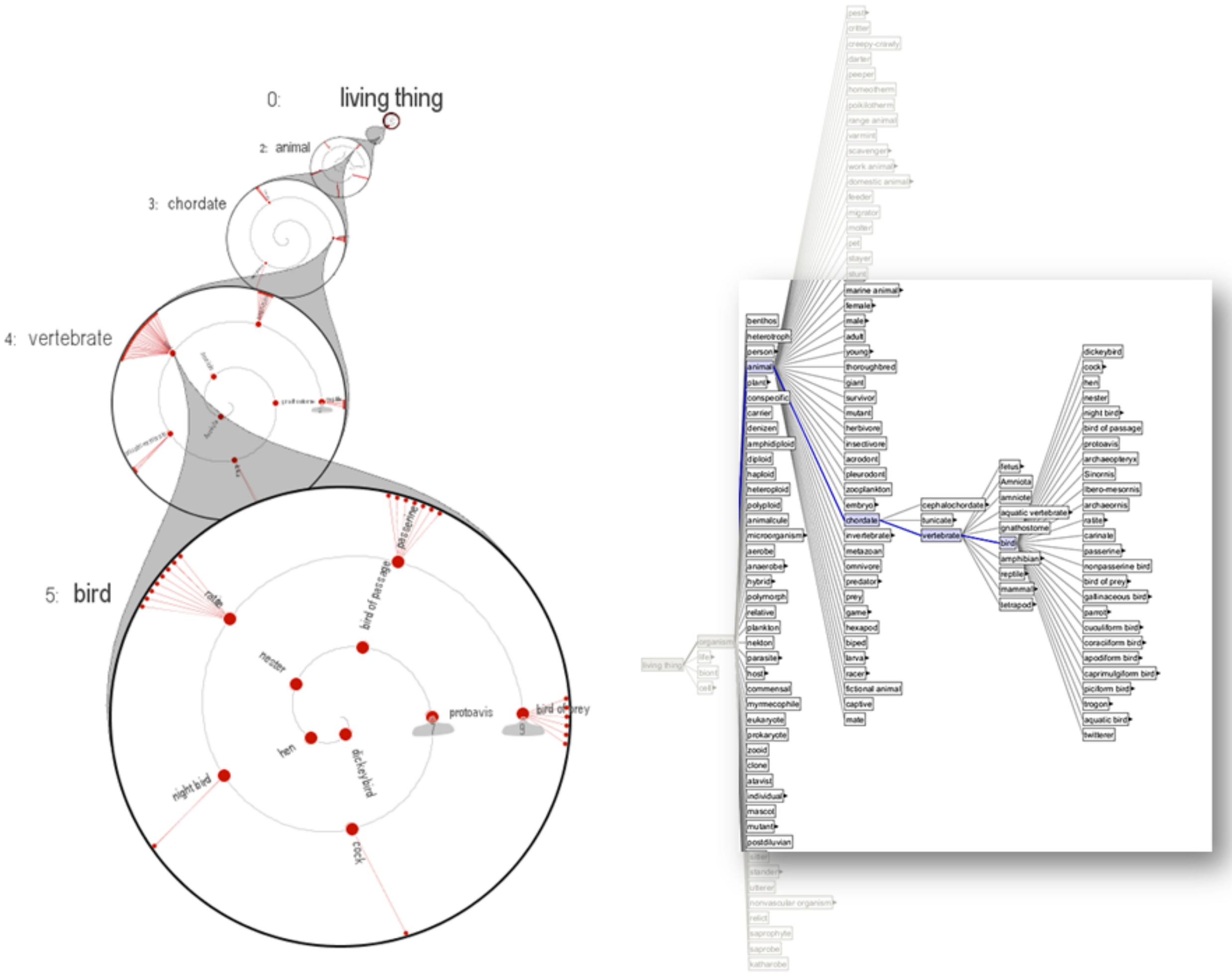

Figure 15. Comparing PasileyTree and SpaceTree: Left: in PaisleyTree the overall size is fixed, so the full structure stays on screen. Right: SpaceTree requires interactive adjustments to view the full structure (in this image the part displayed on screen has been highlighted).

outer rings of large trees where nodes have become very small. The display size is kept relatively compact but visual elements are shrunk in the process, leading to a reading that is mostly a structural impression, rather than providing access to individual nodes. Similar to other techniques that aim to visualize the complete tree, the radial space filling layout is able to convey a picture of a tree's structure, but struggles with larger trees. More importantly, regardless of the size of the tree, there is no mechanism for visually re-focusing the tree based on an NOI. For example, in Figure 14 top-left, the selection of “reference book" as the NOI in Lark only indicates labels of ancestral path, while in PaisleyTree the entire visualization refocuses (and reorganizes) based on the NOI and its proximity.

By taking into account only parts of the tree, SpaceTrees cope much better with larger trees. Due to being based on node-link diagrams, they still grow in size while exploring the tree, though typically gradually. Additionally, on-screen elements remain legible and may be interacted with the whole time. Zooming and panning allows all parts of the tree to be reached. SpaceTree also features several improvements to handle larger trees: the current level is displayed in two columns to use space more efficiently, and structural previews are included, such as small triangles that depict the size and depth/breadth of the subtree.

6.1. Comparison between SpaceTree and PaisleyTree

The main conceptual difference between PaisleyTree and SpaceTree is the focusing strategy of these two visualizations. The zoom/pan method in SpaceTree is a linear approach for presenting the tree-cut containing the NOI, while in PaisleyTree, NOI has pivotal role for defining the tree-cut. In SpaceTree, the interaction for

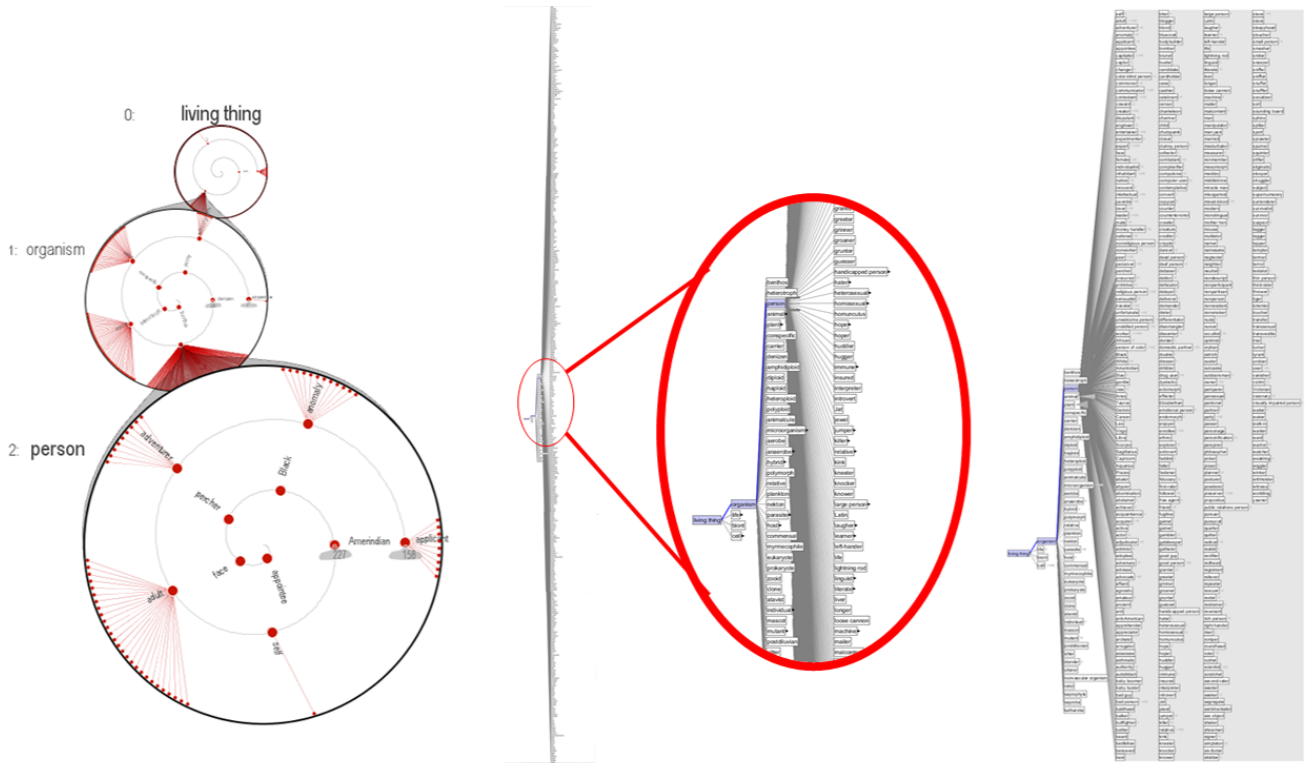

Figure 16. This Figure shows comparison of the descendant tree-cut of PaisleyTree and similar tree-cut of SpaceTree. The node *person* has 163 non-leaf and 232 leaf children.

selecting NOI requires the user to interactively zoom and pan to create a good view of the tree-cut. For larger trees, it is also necessary to interactively organize the vertices, as the default setup does not always provide a suitable arrangement (see Figure 15). In PaisleyTree the only required interaction is to select NOI and the tree-cut is organized automatically based on this NOI.

Keeping overall structure (ancestral tree-cut). In PaisleyTree, we intentionally create two tree-cuts of the NOI: ancestral tree-cut and descendant tree-cut of the NOI. The layout and strategy used for these tree-cuts are different. For the ancestral tree-cut, we present the context and overall structure from NOI to the root. This has been a positive aspect of radial space filling layouts. In essence, the ancestral tree-cut is constructed in proximity to the ancestral path from NOI to the root. A closer zoom in SpaceTree loses the context as shown in Figure 16.

Descendant Tree-cut. For visualizing descendant tree-cut, we use a hybrid method to reduce clutter and make a better use of space (see Figure 16). Using nested layout for the children removes the need for drawing a large set of edges. Furthermore, using circular space and the radial arrangement of the children provides a compact and well organized structure. This arrangement is generated algorithmically. Our current implementation of PaisleyTree can only visualize up to two levels of the descendant subtree. Although other nested layouts such as ShamsehTree [12] can be used for more than two levels, they may not be useful for larger trees.

Aesthetic factor. An exploration of aesthetics has been one of the goals for designing the PaisleyTree layout. In addition to the use of Paisley pattern, circles and spirals are important elements of the PaisleyTree layout. All these elements influence the aesthetics of the visualization [31, 34–36]. Perhaps most importantly, we have used the exploration of a different aesthetic, in this case traditional Paisley patterns, to help us think out of the box and help us create an alternate layout with different presentation and interaction possibilities that expands the options for large tree layout.

Elision. The two types of elisions used in PaisleyTree are unique. While SpaceTree does provide elision for descendant nodes, it does not have elision for ancestral nodes. PaisleyTree's use of ellipses helps in handling an NOI with extreme degree in either number of children or a very long ancestral path. In this way PaisleyTrees can display long ancestral paths as demonstrated in Figure 15. In SpaceTree, the elision is applied to parts of the descendant subtree of NOI as shown in Figure 16 right. The small gray triangles in front of the nodes are ellipses that show the existence of descendant subtrees. Since SpaceTree displays all the children of the NOI, it has difficulty with nodes of high degree (see Figure 16 middle). While PaisleyTree elision for the descendant subtree can handle high degree nodes, more interactions may be needed to extract the hidden nodes (see Figure 16 left).

7. Conclusion

We have presented PaisleyTrees, a novel tree visualization technique intended to offer an alternate solution to the problem of tree visualization techniques that grow in size with the underlying data or shrink their on-screen elements indefinitely. PaisleyTrees combine three different methods to obtain a solution and offers size-invariance: (1) we use a hybrid combination of node-link, adjacency and nested layouts, to efficiently present different parts of the tree in one uniform visualization, (2) we display a tree-cut instead of the whole tree, and (3) we introduce elision on the depth and breadth aspects of the tree to indicate access to the nodes of the tree that would expand the visualization beyond the available screen space. Animated transitions and interaction techniques that also work on mobile devices make PaisleyTrees applicable to both small and large trees. PaisleyTrees are promising in that they directly tackle the problem of size-invariance for tree visualizations. Future work could extend the principles we described here and apply them to other types of data. The idea of hybrid layouts could be applied to other types of data-sets with an existing library of visualization techniques, such as graphs. Similarly, PaisleyTrees focus on a single path through the tree, but a variation that also supports access to different branches and allows direct comparison could be a useful exploration for future.

Acknowledgements

This research was supported in part by NSERC, SMART Technologies, AITF, NSERC Surfnet, and GRAND NCE.

References

[1] Di Battista, G., Eades, P., Tamassia, R. and Tollis, I. (1999) *Graph Drawing: Algorithms for the Visualization of Graphs.* (Prentice-Hall).

[2] Johnson, B. and Shneiderman, B. (1991) Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the 2nd Conference on Visualization, (VIS'91)* (Los Alamitos, CA, USA: IEEE Computer Society Press): 284–291.

[3] Etemad, K. and Carpendale, S. (2010) Symmetry and node focused visualization of large trees. In *Poster: at GRAND NCE Annual Conference, GRAND.*

[4] Schulz, H.J., Hadlak, S. and Schumann, H. (2010) The Design Space of Implicit Hierarchy Visualization: A Survey. *IEEE transactions on visualization and computer graphics* **17**(4): 393–411. doi:10.1109/TVCG.2010.79, URL `http://www.ncbi.nlm.nih.gov/pubmed/20498508`.

[5] Reingold, E. and Tilford, J.S. (1981) Tidier drawings of trees. *Software Engineering, IEEE* **SE-7**(2): 223–228. doi:10.1109/TSE.1981.234519, URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1702828http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1702828`.

[6] Wetherell, C. and Shannon, A. (1979) Tidy drawing of trees. *IEEE Transactions on Software Engineering* **5**(5): 514–520.

[7] Beaudoin, L., Parent, M.A. and Vroomen, L.C. (1996) Cheops: A compact explorer for complex hierarchies. In *Proceedings VIS '96*: 1–14. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=567745`.

[8] Robertson, G., Mackinlay, J. and Card, S. (1991) Cone Trees: Animated 3D Visualizations of Hierarchical Information. In *Proceedings CHI '91*: 189–194. URL `http://dl.acm.org/citation.cfm?id=108883`.

[9] Neumann, P., Carpendale, S. and Agarawala, A. (2006) Phyllotrees: Phyllotactic patterns for tree layout. In *Proceedings of Eurographics / IEEE VGTC Symposium on Visualization, (EUROVIS'06)* (Eurographics): 59–66. doi:http://dx.doi.org/10.2312/VisSym/EuroVis06/059-066.

[10] Hao, L. and Fogarty, J. (2008) Cascaded treemaps: examining the visibility and stability of structure in treemaps. In *Proceedings of Graphics Interface, (GI'08)* (Toronto, Ontario, Canada): 259–266.

[11] Boardman, R. (2000) Bubble trees, the visualization of hierarchical information structures. In *Extended Abstracts on Human Factors in Computing Systems, (CHI'00)* (New York, NY, USA: ACM Press): 315–316. doi:http://doi.acm.org/10.1145/633292.633484.

[12] Etemad, K. and Carpendale, S. (2009) Shamsehtrees: Providing hierarchical context for nodes of interest. In *Proceedings of Bridges 2009: Mathematics, Music, Art, Architecture, Culture* (Tarquin Books): 293–300.

[13] Collins, C., Carpendale, S. and Penn, G. (2009) Docuburst: Visualizing document content using language structure. *Computer Graphics Forum* **28**(3): 1039–1046.

[14] Chevalier, F., Auber, D. and Telea, A. (2007) Structural analysis and visualization of c++ code evolution using syntax trees. In *Proceedings of the 9th International Workshop on Principles of Software Evolution, (IWPSE'07)* (New York, NY, USA: ACM Press): 90–97. doi:http://doi.acm.org/10.1145/1294948.1294971.

[15] Stasko, J., Catrambone, R., Guzdial, M. and Mcdonald, K. (2000) An evaluation of space-filling information visualizations for depicting hierarchical structures. *International Journal of Human-Computer Studies* **53**(5): 663–694. doi:10.1006/ijhc.2000.0420, URL `http://linkinghub.elsevier.com/retrieve/pii/S1071581900904208`.

[16] Andrews, K. and Heidegger, H. (1998) Information slices: Visualising and exploring large hierarchies using cascading, semi-circular discs. In *Proc of IEEE Infovisï£¡ 98 late breaking Hot Topics*: 9–11.

[17] Plaisant, C., Grosjean, J. and Bederson, B.B. (2002) Spacetree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In *Proceedings INFOVIS '02*: 57–64.

[18] Card, S. and Nation, D. (2002) Degree-of-interest trees: A component of an attention-reactive user interface. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (ACM): 231–245. URL http://dl.acm.org/citation.cfm?id=1556300.

[19] Heer, J. and Card, S. (2004) DOITrees revisited: scalable, space-constrained visualization of hierarchical data. In *Proceedings of the working conference on Advanced visual interfaces* (ACM): 421–424. URL http://dl.acm.org/citation.cfm?id=989941.

[20] Li, H. and Abe, N. (1998) Generalizing case frames using a thesaurus and the MDL principle. *Computational linguistics* **24**(2): 217–244. URL http://dl.acm.org/citation.cfm?id=972734.

[21] Zhao, S., McGuffin, M. and Chignell, M. (2005) Elastic hierarchies: Combining treemaps and node-link diagrams. In *Proceedings INFOVIS '05*: 57–64. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1532129.

[22] Linsen, L. and Behrendt, S. (2011) Linked treemap: a 3D treemap-nodelink layout for visualizing hierarchical structures. *Computational Statistics* : 679–697doi:10.1007/s00180-011-0272-2, URL http://www.springerlink.com/index/L1327352V579R838.pdf.

[23] Shin, H., Park, G. and Han, J. (2011) Tablorer - An Interactive Tree Visualization System for Tablet PCs. *EuroVis 2011* **30**(3): 1131–1140. doi:10.1111/j.1467-8659.2011.01962.x, URL http://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2011.01962.x/full.

[24] Karstens, B., Kreuseler, M. and Schumann, H. (2003) Visualization of complex structures on mobile handhelds. In *Proceedings Workshop on Mobile Computing*. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.112.1960&amp;rep=rep1&amp;type=pdf.

[25] Hao, J. and Zhang, K. (2007) A mobile interface for hierarchical information visualization and navigation. In *Proceedings ISCE '07* (IEEE): 1–7. doi:10.1109/ISCE.2007.4382214, URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4382214.

[26] Scruton, R. (2009) *Why Beauty Matters* (BBC2).

[27] Skog, T., Ljungblad, S. and Holmquist, L.E. (2003) Between aesthetics and utility: designing ambient information visualizations. In *Information Visualization, 2003. INFOVIS 2003. IEEE Symposium on* (IEEE): 233–240.

[28] Eric, W. (2009), Fermat's spiral. mathworld–a wolfram web resource., http://mathworld.wolfram.com/FermatsSpiral.html.

[29] Elmqvist, N., Moere, A.V., Jetter, H.C., Cernea, D., Reiterer, H. and Jankun-Kelly, T. (2011) Fluid interaction for information visualization. *Information Visualization* **10**(4): 327–340. doi:10.1177/1473871611413180, URL http://ivi.sagepub.com/lookup/doi/10.1177/1473871611413180.

[30] Attneave, F. (1955) Symmetry, information, and memory for patterns. *American Journal of Psychology* **68**(2): 209–222. doi:http://www.jstor.org/stable/1418892.

[31] Zhang, K. (2007) From abstract painting to information visualization. *Computer Graphics and Applications* **27**(3): 12–16. doi:10.1109/MCG.2007.58.

[32] Fellbaum, C. (1998) *WordNet: An Electronic Lexical Database.* (Cambridge, MA, USA: MIT Press).

[33] Tobiasz, M., Isenberg, P. and Carpendale, S. (2009) Lark: coordinating co-located collaboration with information visualization. *IEEE transactions on visualization and computer graphics* **15**(6): 1065–72. doi:10.1109/TVCG.2009.162, URL http://www.ncbi.nlm.nih.gov/pubmed/19834173.

[34] Etemad, K., Carpendale, S. and Samavati, F. (2014) Spirograph inspired visualization of ecological networks. In *Proceedings of Computational Aesthetics in Graphics, Visualization, and Imaging* (ACM): 81–91.

[35] C. Bennett, C., Ryall, J., Spalteholz, L. and Gooch, A. (2007) The aesthetics of graph visualization. *Computational Aesthetics in Graphics, Visualization, and Imaging* **27**: 1–8.

[36] Coleman, M. and Parker, D. (1996) Aesthetics-based graph layout for human consumption. *Softw. Pract. Exper.* **26**(12): 1415–1438. doi:http://dx.doi.org/10.1002/(SICI)1097-024X(199612)26:12<1415::AID-SPE69>3.3.CO;2-G.