

# A survey of multiple tree visualisation

Martin Graham\* and  
Jessie Kennedy

School of Computing, Napier University,  
10 Colinton Road, Merchiston, Edinburgh,  
Lothian, EH10 5DT, UK.  
E-mails: m.graham@napier.ac.uk;  
kennedy@napier.ac.uk

\*Corresponding author.

**Abstract** This article summarises the current state of research into multiple tree visualisations. It discusses the spectrum of current representation techniques used on single trees, pairs of trees and finally multiple trees, in order to identify which representations are best suited to particular tasks and to find gaps in the representation space, in which opportunities for future multiple tree visualisation research may exist. The application areas from where multiple tree data are derived are enumerated, and the distinct structures that multiple trees make in combination with each other and the effect on subsequent approaches to their visualisation are discussed, along with the basic high-level goals of existing multiple tree visualisations.

*Information Visualization* advance online publication, 5 November 2009;  
doi:10.1057/ivs.2009.29

**Keywords:** multiple trees; layout techniques; survey

---

## Introduction

Tree visualisation has been one of the staples of Information Visualisation (IV) since its inception, inspiring myriad variations in terms of layout styles and interaction techniques. The bulk of this research has been performed on single tree instances, yet as more data are produced that requires users to navigate through multiple tree classifications or to understand the complex interaction of several tree structures, an increasing number of situations call for the analysis and comparison of *multiple* tree structures – tasks to which IV techniques can also be successfully applied.

There have been attempts to categorise tree visualisation literature by criteria such as layout style, interaction technique and data type in previous survey papers. Again, however, most of them concern themselves with only the visualisation of single trees. Noik<sup>1</sup> provides an early survey and classification system of graph presentation techniques, including trees, primarily focused on classifying by focusing and filtering attributes, and is too early for multiple tree visualisations to have registered in the literature. Later, Herman *et al*<sup>2</sup> surveyed the graph visualisation literature from an IV perspective, grouping techniques by interaction and layout style rather than through the more formal algorithmic perspective favoured in traditional graph drawing domains. Their examples cover mostly restricted graphs including trees along with directed acyclic graphs (DAGs) and other graphs derived from general graphs. As with Noik, the survey does not include any examples of multiple tree visualisations due to the survey now being several years old. More recently, PhD and Masters' theses regarding tree visualisation such as Nguyen<sup>3</sup> and Nussbaumer<sup>4</sup> include background chapters summarising the state-of-the-art at the time, but again cover only single tree visualisations.

Graham's thesis<sup>5</sup> includes a background chapter on multiple tree visualisation as it existed in 2001 but, as much of the work on the topic has occurred since then, recent work has not yet been summarised or explored comprehensively. As such, this article aims to act as a survey paper for

---

Received: 22 December 2008  
Revised: 3 July 2009  
Accepted: 23 July 2009

the field, collecting and summarising the current state-of-the-art in multiple tree visualisation. We begin by describing the areas in which multiple tree data are to be found and the different types of structure formed by overlapping trees. Then we give a brief summary of single tree visualisation before moving on to two-tree and multiple tree visualisations in greater detail, discussing the techniques used to display and interact with these structures and their associated advantages and disadvantages. From the review of such work we summarise the basic tasks that multiple tree visualisations are attempting to allow users to perform.

## Application Areas

The primary sources of data, and therefore application areas, for multiple tree visualisations are bioinformatics, faceted classifications, schema/ontology mapping and software development. These are fields with a common thread of re-classification or re-categorisation that produce in some form or other multiple tree data sets that users are interested in analysing for finding patterns, editing relationships or using as mechanisms for querying data.

In bioinformatics, analyses of species relationships produce multiple conflicting phylogenies and taxonomies over the same or overlapping groups of species; each instance being a model of evolutionary or morphological similarity between those species. Constantly, new classifications are produced over previously classified data due to the discovery of new species and the development of new analytic techniques. Currently, reconciling such structures relies on producing consensus trees<sup>6</sup> that accent shared structure but omit unique details of individual tree structures in the process, even though practitioners regard each conflicting classification as legitimate in its own right.

Hierarchically faceted classifications<sup>7</sup> occur when items can be filtered by multiple *hierarchically organised* attributes, with the most common example being internet shopping sites where products are classified by price, type, maker and so on. We must be careful to distinguish between flat facets – those facets that group a set of objects according to one level of categorisation, for example manufacturer (Dell, HP, Viglen and so on) – and hierarchical facets that have multiple levels of categorisation – an ‘operating systems’ facet can have Windows, Mac OS and Linux at the top level, and in turn Windows can have sub-categorisations of Vista, XP, 98, 95 and so on. It is the hierarchical facets that are of interest to us, as a collection of such facets can be understood as a collection of multiple, overlapping trees.

Hierarchical facets are similar to biological taxonomies in that multiple hierarchies are constructed over a set of objects, though there are two distinct differences. Firstly, each faceted classification will include all the objects in a set, even if some objects end up being assigned to

an ‘unknown’ or ‘other’ group within a classification, whereas biological taxonomies may only roughly overlap on the object set rather than match completely. Secondly, the problem for users with biological classifications is trying to reconcile these different views of the data, whereas with faceted classifications the ability to filter through multiple trees, as seen, for example, in Sifer,<sup>8</sup> is an aid to the end user, designed to free users from just the one way of browsing or interrogating data. We can say such classifications are complementary rather than conflicting in purpose.

Research into ontologies and semantic web issues has looked at the problem of *schema* or *ontology mapping*,<sup>9</sup> where the challenge is to find the best match between the various parts of related schemas or ontologies. XML schemas are hierarchical, and while ontologies are more complex they do contain major tree-based structures such as concept and relation hierarchies. As such, they can often have mapping issues that fit into the domain of multiple trees as seen in Aumueller *et al.*<sup>10</sup> Mapping can be automated to various extents by name and structure analysis and also by applying further ontologies that form a semantic bridge between different schemas, but an associated visualisation as in Altova’s MapForce product<sup>11</sup> and Wang *et al.*’s SCSI application<sup>12</sup> supplies a mechanism for a human expert to validate relationships and, more importantly, to resolve instances that are not amenable to automatic methods.

Lastly, software development versioning systems such as CVS produce snapshots of classes and packages that are organised hierarchically, and software developers may wish to compare these snapshots to track package growth and discover where development effort was concentrated at any particular time. As computer scientists like to solve their own problems first, there have been many attempts to visualise software evolution, an early example being Eick *et al.*’s SeeSoft system.<sup>13</sup> These tools in turn are advances on visual *diff* tools that analyse source files or directories to discover differences. For directories and hierarchically structured files such as XML, the problem again becomes one of mapping between multiple trees. Software visualisation undoubtedly covers a much wider area than simply revisions of package/class hierarchies, but of those that do examine this aspect, Gırba *et al.*’s work<sup>14</sup> attempts to explicitly show the evolution of hierarchical structures alongside other attributes in code repositories, and Wu *et al.*<sup>15</sup> display CVS package hierarchies as adjacent entities for user examination.

The semantics of a data set make a difference to the choice of multiple tree visualisations to use, and probably the most basic dichotomy in tree data semantics is in the difference between internal and leaf nodes. In a biological taxonomy, every node is simply a container for the nodes below it and leaf nodes are nothing special in this respect, they are merely the same type of node except they have zero child nodes. In contrast, class hierarchies in software visualisation have the actual classes (compiled code) as leaves and the internal nodes are arbitrary categorisations



of those classes, thus there is a semantic difference in this case between leaf and internal nodes.

Together with basic research into new IV techniques, investigations into handling data from these domains accounts for the vast majority of literature in multiple tree visualisation.

## Multiple Tree Structures

The logical structures formed by merging multiple trees dwell in the messy domain between single tree structures and general graphs; the exact type of structure formed depends on the trees' node overlap characteristics and structural similarity. The starting point is to define what a tree itself is, with one formal definition being that it is an undirected graph structure with one path and one path only between any pair of nodes in the graph. It must thus be acyclic as the presence of cycles would produce multiple possible paths between some nodes in an undirected graph. The more common, colloquial definition is to think of a tree as a rooted structure, starting at a root node that may have zero or more children nodes linked to it. In turn each of these nodes can link to other child nodes, so that each node has one parent node only (except the root, which has no parent) and zero or more children nodes. IV literature tends to freely swap the terms 'tree' and 'hierarchy' even though hierarchies are not always trees, though others are tighter by referring to a 'hierarchical tree'. The essential difference is the presence of multiple inheritance as found in pedigrees<sup>16</sup> or class

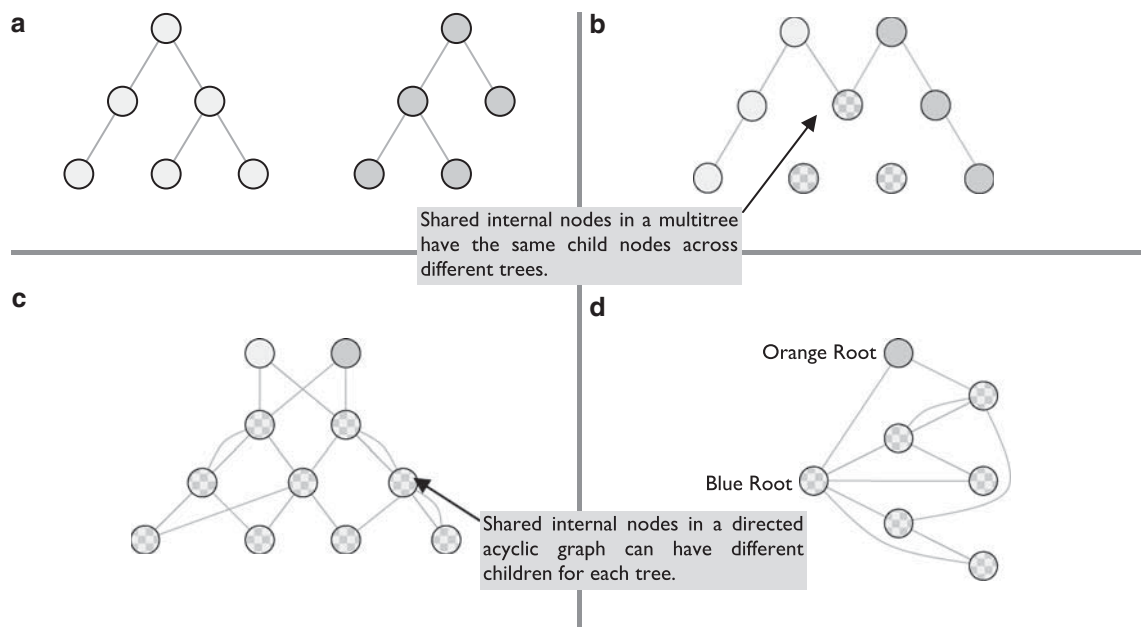
hierarchies<sup>17</sup> – an entity can have one *or more* parents in a hierarchy, but only one in the strict definition of a tree.

There are four basic structures that can be constructed if we consider direct node overlap only, shown in Figure 1, where these structures fit in a subsumption hierarchy of restricted graph structures as outlined in McGuffin and Schraefel.<sup>18</sup>

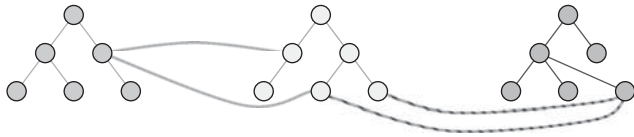
The simplest case is that where no node overlap occurs between a set of trees. The resulting structure is a forest, a collection of disjoint trees as shown in Figure 1(a), though as discussed later other relationships may yet exist between the trees.

Trees that only share leaf nodes form Multitrees as envisaged by Furnas and Zacks<sup>19</sup> and also demonstrated in Wittenburg *et al*<sup>20</sup>; related phylogenies fall into this category, as do faceted classifications, in which the same group of objects are categorised under different classifications.<sup>21</sup> Multitrees also capture situations where trees reuse fragments of other classifications instead of building new structures where a previously defined structure already resides, shown in Figure 1(b). The prime example of this type of structure are family trees as visualised by McGuffin and Balakrishnan;<sup>22</sup> one person's descendants may overlap with another, but the structure of the parts that do overlap will be the same.

Tree collections whose shared interior nodes are subject to restructuring can be divided into two categories depending on whether there is a global parent-node orientation across the trees. If there is a global orientation then the multiple trees form a DAG, or a multi-DAG as seen in Figure 1(c) if multiple edges are kept distinct, such



**Figure 1:** Structures that multiple trees can make through node overlap – (a) Forest, (b) Multitree, (c) DAMG and (d) Polyarchy.



**Figure 2:** Inter-tree links defined between non-overlapping nodes.

as those produced by Graham's multiple taxonomies.<sup>5</sup> For example, a biological taxonomy may reuse internal nodes such as kingdoms and genera from another taxonomy but then construct its own paths between them by defining other internal nodes; however, a global parent-child orientation is preserved, as a genus will always be lower down than a kingdom in any taxonomic tree, never the other way round.

If there is no global orientation shared between the structures, then the combined trees form a polyarchy structure (Figure 1(d)) as identified by Robertson *et al*<sup>23</sup> and used by Conklin *et al*'s drill-down data browser.<sup>24</sup> Here the distinction between leaf nodes and internal nodes across trees breaks down; nodes that form parent-child relationships in one tree may end up forming the inverted relationship in another tree, or be siblings in yet another tree or unrelated altogether; leaf nodes in one tree may well be internal nodes in another and vice versa.

For many scenarios, simple node overlapping does not adequately describe the mapping between the trees. In these cases, the mapping is described through inter-tree edges that relate the trees to each other. These relations can either be one-to-one or one-to-many as shown in Figure 2, and can also have data such as typing or weighting associated with those edges. For example, in biological taxonomy there can be explicit relations defined between nodes in different classification trees, as detailed in Graham and Kennedy,<sup>25</sup> which are deduced by field experts. These links can indicate a spectrum of different relations based on set notation such as, congruence, dissimilarity and so on, and one node may be the source or target of several links involving one or more other trees. What appears to be a collection of disparate trees from the perspective of node sharing, now reveals a complex tree-based graph structure.

Ontology and schema matching visualisations, such as Dadzie and Burger's mouse ontology viewer,<sup>26</sup> display similarly complex relationships. These relationships can be generated by various means such as comparing element names, structural similarities or analysing semantic similarities as seen in Cruz *et al*<sup>27</sup> – in a simple example a single full name element in one schema matches to a combination of first and last name elements in another. Similar relationships can also occur in software versioning, in which renaming or refactoring may give the same class or method different names between different releases, or cause classes to be split or joined together between

different releases. Tu and Godfrey<sup>28</sup> describe matching such files by a joint comparison of software metrics and name string matching between files.

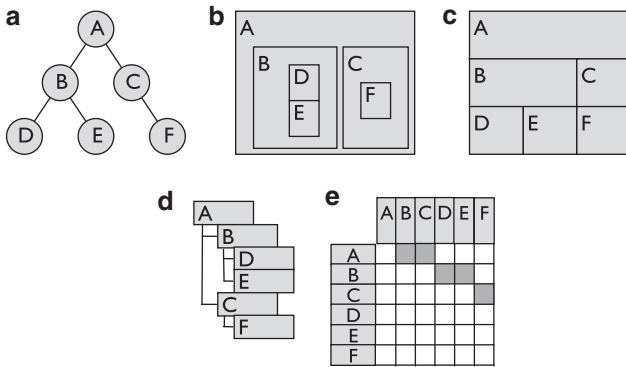
## Representations

The most noticeable variable in multiple tree visualisations is the representation used to show and allow interaction with multiple tree structures. The representations can be based on layouts used for individual trees or on layouts designed for restricted graphs, such as DAGs, or adapted from those used to display more general graph structures. Some of the options often used for single trees, such as node-link representations are applicable to more complex graph structures such as multiple trees, and the widespread use of the small multiple<sup>29</sup> technique means that single tree representations often feature prominently in multiple tree visualisations. Therefore we begin our representation section with a quick discussion of single tree representation styles, which is by no means exhaustive in its coverage of the literature. For a fuller review of single tree visualisations we recommend the theses referenced in the introduction such as Nguyen<sup>3</sup> and Nussbaumer.<sup>4</sup>

### Single trees

Single tree visualisations have a long history before the coining of the term 'Information Visualization', a classic reference being Reingold and Tilford's<sup>30</sup> work, itself only one of many pre-1990 algorithms listed in Beebe's exhaustive bibliography<sup>31</sup> for tree drawing. However, these approaches tended to focus exclusively on layout algorithms; what IV introduced was the notion of being able to interact with the generated tree visualisations.

Single tree layout divides into a number of categories, based on the graphical method used to indicate a parent-child relationship, five of which are shown in Figure 3. The first and most widespread is the node-link layout,<sup>30,32</sup> with parent-child relations represented as lines (links) drawn between the visual objects that represent the nodes in the tree. Secondly, nested or enclosure layouts<sup>33,34</sup> indicate the same relationship by placing child node representations within the boundaries of their parent node. Thirdly, there is the adjacency layout or icicle plot<sup>35</sup> style where child nodes are drawn as abutting their parent node. This method, more than the node-link approach, requires the definition of a parent-child orientation to differentiate parent-child relations from sibling relationships and to indicate the direction of a relationship. Usually this orientation is either top-down as in Graham and Kennedy's taxonomy browser<sup>25</sup> or centre-out as in Stasko *et al*.<sup>36</sup> All three of these layout styles have been extended from their original 2D projection to 3D variants with various degrees of success: node-link,<sup>37</sup> nested<sup>38</sup> and adjacency.<sup>39</sup>



**Figure 3:** Basic types of tree representation – (a) node-link, (b) nested, (c) adjacency, (d) indented list and (e) matrix representations.

A fourth representation style is indentation, in which nodes are listed linearly in order of depth-based traversal and then indented by an amount proportional to their depth in the tree. Often, stylised links are drawn to make parent–child relationships clearer, but this is not always the case. This is the most common form of tree display used in contemporary graphical user interfaces (GUIs), seen in locations such as the folders view of Microsoft Windows Explorer. In empirical evaluation by Cockburn and Mackenzie<sup>40</sup> this layout has shown to be the objectively preferred choice when compared to other styles of tree visualisation, though Kobsa<sup>41</sup> suggested that much of this performance advantage is explained by familiarity because of the ubiquitous presence of Microsoft Windows.

Finally, individual trees can also be displayed via a matrix representation, but this tends to be less common than the previous styles for good reason. Firstly, this is because of the difficulty in following edge paths in matrices, as recognised by Shen and Ma.<sup>42</sup> In Figures 3(a), (c) and (d), it is clear that D is a ‘grandchild’ of A, and while slightly trickier in the case of the nested representation in Figure 3(b) (Lü and Fogarty<sup>43</sup> discuss how variation in nested representations can greatly affect this property), in the matrix representation the A–B and B–D edges need to be discerned independently and then combined, making the relationship much more difficult to deduce. A second issue is that essentially a single tree is not complicated enough in structure to warrant a matrix representation. One of the main reasons cited for using matrices to visualise graph types is that they eliminate edge crossings that occur in other graph representations, but a single tree can always be drawn with no edge-crossings in the other representations and so this reason no longer applies. Further to this point, a tree with  $N$  nodes has  $N - 1$  edges, and thus when displayed as a matrix will only fill the square root of the total  $N^2$  possible entries, making it highly space-inefficient.

All the layout styles have associated advantages and disadvantages and the choice of representation is

depending on the tasks that are to be performed with the structures and the semantics of the data concerned. Generally node-link representations are more understandable to the lay-person and communicate structure readily, but use up screen space rapidly. Nested representations allow more nodes to be displayed at once but structure is more difficult to perceive due to lacking a global child-parent orientation, plus they emphasise leaf nodes at the expense of internal nodes. The adjacency and indented list methods strive for a halfway house between these two styles, utilising a higher proportion of screen space than a node-link display, yet making structure relatively simple to follow. Finally, the matrix reduces the tree essentially to a look-up table. These basic layout styles are the foundation for all tree visualisations that display internal tree structure, and the styles themselves can be combined within a visualisation of a single tree as demonstrated by Zhao *et al.*<sup>44</sup> in which portions of the tree are drawn as either nested or node-link representations dependent on screen space and user interaction. Further, Nguyen and Huang’s EncCon technique.<sup>45</sup> combines the enclosure and node-link approaches across an entire tree; the tree nodes being positioned using an optimised nested layout algorithm and then connected with links.

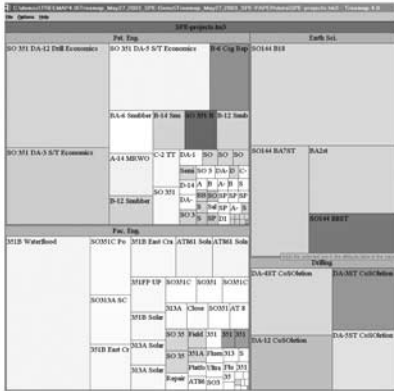
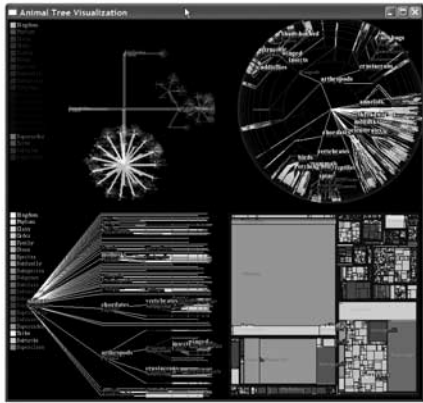
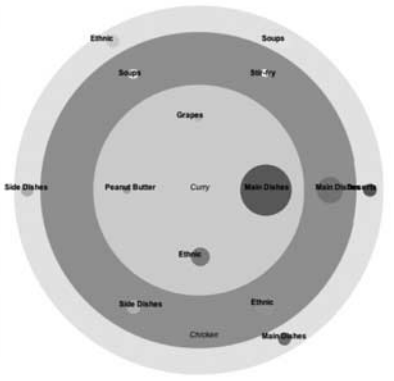
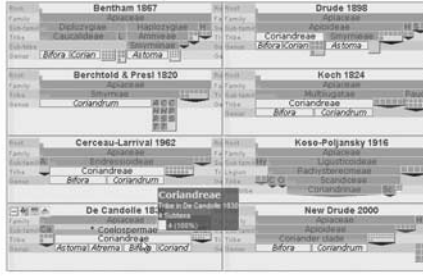
### Multiple tree models × Multiple tree representations

A logical starting point to categorise multiple tree visualisations is to distinguish whether ‘multiplicity’ is based on the number of trees displayed, or the number of trees modelled in the structure, or both. Table 1 shows a brief tabular summary of this categorisation and the four basic cases it produces – with the simplest case of a single tree model represented as a single tree visualisation being covered in the previous section.

The second case covers the scenario of one tree model visualised many times; for instance Wilson and Bergeron’s dynamic hierarchy visualization<sup>46</sup> can display multiple, differing representations of the *same* hierarchy, but does not display multiple structures. A similar caveat applies to Urbanek’s KLIMT system,<sup>47</sup> Schedl *et al.*’s<sup>48</sup> stacked radial tree visualisation and Teoh’s more recent work<sup>49</sup> on multiple views for trees. Kules *et al.*<sup>50</sup> explore the situation of simultaneously using two different, linked representation styles of the same tree – one nested and one node-link representation.

Of more interest to us are the approaches that deal with multiple instances of trees in the data we wish to visualise, and these can be divided into visualisations that are shown as a single tree or show multiple trees. The former case tends to be visualisations built for hierarchical facet exploration, such as MoireTrees<sup>51</sup> and Facet Folders,<sup>52</sup> that try and give a fluid single tree view over a multi-hierarchical structure for ease of navigation. The latter case is that of visualisations that display multiple representations of multiple trees. Here there may not be a universal coverage of leaves by each hierarchy – some may have

**Table 1:** Number of trees compared with number of individual representations

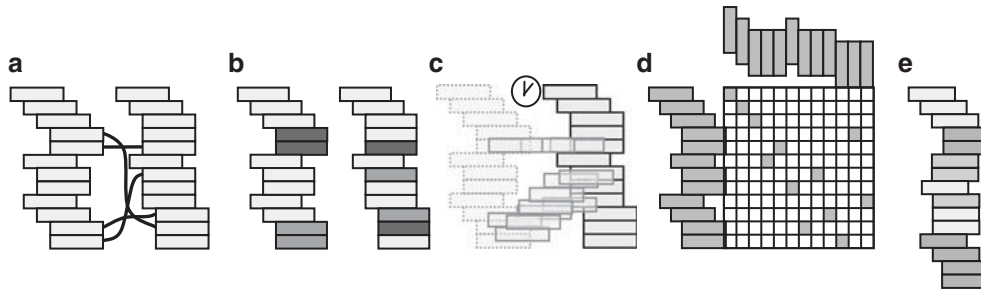
		Number of Tree Representations	
		1 Tree	Multiple Trees
Number of Trees in Structure	1 Tree	<p>Traditional single tree visualisations (e.g. Cone Trees<sup>37</sup>, TreeMaps<sup>33</sup>)</p>  <p>Original TreeMap - (screenshot courtesy of Human-Computer Interaction Laboratory, University of Maryland, College Park)</p>	<p>Multiple/multiform views of a single tree structure (e.g. Wilson &amp; Bergeron<sup>46</sup>, Teoh<sup>49</sup>)</p>  <p>Teoh's Multiple Views of a tree data set</p>
	Multiple Trees	<p>Single hierarchical view of navigation through faceted hierarchies (e.g. MoireTrees<sup>51</sup>)</p>  <p>MoireTrees visualisation</p>	<p>Full multiple tree visualisation (e.g. Graham &amp; Kennedy<sup>57</sup>, Munzner et al<sup>58</sup>, Sifer<sup>8</sup>)</p>  <p>Graham &amp; Kennedy's multiple taxonomy visualisation</p>

more leaves than others and in the case of polyarchy structures the notion of what leaves are may change between hierarchies.

This difference between single and multiple tree representations for multiple tree models is not as clear-cut as a simple dichotomy would suggest, especially in the case of visualisation interfaces for faceted hierarchy exploration. Although for some instances it is fairly clear how many tree representations are being shown, for instance the Mambo music browser<sup>53</sup> only displays one

hierarchical facet at a time, and others such as FacetMap,<sup>54</sup> FacetLens<sup>55</sup> and Yee et al's image browsing system<sup>56</sup> simultaneously display at least parts of a hierarchy per facet type, yet others such as MoireTrees and Facet Folders merge portions of multiple hierarchical facets into one hierarchy representation. The essential difference is that while the same multiple hierarchy models can operate behind all these various interfaces, the latter have deliberately chosen to project this model onto a single hierarchical view (Table 1).





**Figure 4:** Methods of comparing nodes in two trees – (a) edge drawing, (b) colouring, (c) animation, (d) matrix representation and (e) agglomeration – shown using indented list representations as the basis for individual trees.

## Two trees

The first stage in visualising multiple trees is to visualise two trees, approaches to which can be seen in Figure 4 and can be categorised as:

- (a) Linking two spatially separate tree representations.
- (b) Shared colouring between two spatially separate tree representations.
- (c) Animation between trees (temporally separate).
- (d) Matrix comparison of two trees.
- (e) Spatial agglomeration of two tree structures.

### Edge drawing

The first and most widespread technique in the literature is to display both trees as separate structures with lines drawn between them to indicate relationships between nodes they connect<sup>27,59–62</sup> (as in Figure 4(a)). This allows exact and individual relationships to be traced between different trees. The drawback with this approach is if there are many lines displayed at once, then individual edges become impossible to distinguish from the mass of drawn lines, a common problem in the perception of graph drawings as recognised by Purchase,<sup>63</sup> of which this is the specialised case of drawing bipartite graphs,<sup>64</sup> plus the added constraint that both parts of such a graph are grouped into tree structures. In response to this, Robertson *et al*<sup>65</sup> describe a collection of methods to alleviate traceability difficulties, Holten and van Wijk<sup>66</sup> use a bundling technique to visually group related links drawn between two hierarchies and Buchin *et al*<sup>67</sup> explore formal algorithms for reducing the number of edge crossings between two binary trees. Further, many such visualisations such as Craig and Kennedy<sup>61</sup> only show selected subsets of links between trees to reduce the tangled appearance of a fully populated display.

### Colouring

A closely related technique is to show matching nodes between trees through the use of shared colouring as in Figure 4(b), used by Parr *et al*'s DoubleTree system<sup>68</sup> and

TreeJuxtaposer,<sup>58</sup> Zoomology<sup>69</sup> and EVAT<sup>70</sup> applications developed for the InfoVis 2003 contest<sup>71</sup> among others. This technique gives a more general overview of the amount of overlap between trees. In essence, a coloured node is signalling that it has a match somewhere in the other tree, whereas a drawn link signals exactly which node it matches in the opposite tree. Further interaction, such as individual node brushing, is necessary to then locate exact matches of nodes between trees. The vast majority of the numerous commercial or freeware applications for comparing file directories and XML files such as Microsoft's WinDiff<sup>72</sup> use colouring with a linearly indented list representation as it is the simplest effect to implement with standard GUI toolkits. Some merge the approach with edge drawing, such as Kompare<sup>73</sup> and Altova's DiffDog<sup>74</sup> that use both lines and colouring to indicate similarities between two structures.

A further difference between the colouring and edge drawing approaches is that the edge drawing approaches tend to display their trees in representation styles such as a horizontal node-link layout,<sup>75</sup> a horizontal space-filling adjacency layout<sup>61,76</sup> or indented lists<sup>62</sup> that have a non-radial orientation. This enables edges to be drawn, such that, while they may unavoidably cross each other, they do not cross any of the nodes in the trees. The shared colouring approaches do not have to contend with this issue and can therefore use a more varied array of representations, including nested layouts such as Treemaps<sup>20</sup> and hyperbolic trees.<sup>77</sup>

### Animation

Animation such as found in Robertson *et al*<sup>23</sup> is used to display change between representations of different trees, in effect distinguishing the trees temporally rather than spatially. Animation is also used for showing changes in values associated with tree nodes as in Ghoniem and Fekete<sup>78</sup> or the small-scale addition and deletion of nodes seen in Wittenburg and Sigman.<sup>79</sup> In practice, animation is best used either for showing gradual transitions that represent evolving change in the structure of a tree, or in showing switches between hierarchies, in which where only a few nodes stay constant, rather than radical

reorganisation where a user can easily lose track of the overall change. In both cases animation is effective when only a few nodes are either moved or preserved with consideration to the entire node set.

### Matrix

The fourth method for visually comparing two trees is to arrange the trees along the horizontal and vertical axes of a matrix as shown in Figure 4(d), with the matrix now showing relationships or shared leaves between the two trees – this differs from a single tree matrix representation that has the same tree arrayed along both axes. A choice can be made as to whether only leaf nodes are involved in the comparison, in which case an adjacency style or node-link representation along the axes is used, or whether direct internal node comparisons are also to be made in which case using the indented list representation of a tree along the axes gives space for appropriate columns and rows as in Figure 4(d). This choice as previously stated boils down to data semantics – whether an internal node is simply the sum of its leaf nodes or has unique properties in itself.

This method removes the problem of interpreting impenetrable masses of drawn edges, but suffers from the same under-utilisation of screen space as a single tree in the same style. This is especially true if the mapping between tree nodes is strictly one-to-one – in which case the number of unused entries in the matrix is proportional to the square of the number of utilised spaces – thus the effect is exacerbated with larger trees. As such, it tends to be used by techniques wanting to show one-to-many relationships between nodes in related trees. The most common occurrence of this representation is in bioinformatics, in which it is known as a cluster heatmap,<sup>80</sup> a particular use being to plot the distribution of microarray data sets as seen in Eisen's TreeView software.<sup>81</sup> Further, if we stretch our definition of two trees to include two subtrees of the same larger tree, then into this category fall source code and general graph analysers such as van Ham's call matrix visualisation.<sup>82</sup> Here, as with other general clustered graph visualisations, the hierarchies are used as aids to access a general graph structure that resides at the bottom level of the trees, rather than being the focus of queries themselves.

### Agglomeration

This final approach fuses together two tree structures into one structural visual representation. Furnas and Zacks' Multitrees<sup>19</sup> allows two tree structures to be fused together in one node-link representation, in this instance the context is that of a family tree, with one tree showing ancestors and the other descendants. Using their own definition of Multitrees, it follows that the structures between these trees are always shared exactly, so a node in one tree always has the same set of edges in the other.

For more involved structures where parentage of nodes may change between trees, Tu and Shen<sup>83</sup> propose a structure known as a union tree in which nodes that have different parents between the two trees are cloned to appear under both parents simultaneously in a merged structure. This removes cycles from the merged structure, retaining a strict tree structure for subsequent visualisation while preserving the edge sets found in both component trees. This is a common technique when dealing with DAG structures that are to be visualised using tree layouts – spanning trees can perform the same function but by removing edges rather than cloning nodes. Tu and Shen then visualise the union tree with a TreeMap layout and use shading effects within the node representations to indicate changes in node attributes and presence between the two trees.

Hong *et al*'s Zoomology browser<sup>69</sup> features a similar technique to visualise two trees in one merged adjacency representation. The representation is visualised from the perspective of one of the two trees, with areas marked in white indicating nodes 'missing' from the current tree but present in the other, and, for the reverse condition, white borders used to mark nodes occurring only in the current tree. Lee *et al*<sup>84</sup> also merge two similar trees into one structure and display the result as a node-link visualisation. They then use colour and transparency cues to indicate the calculated degree or certainty of structural overlap between the two hierarchies. Isenberg and Carpendale<sup>85</sup> combine elements of multiform view techniques with aspects of agglomeration on a table-top display, letting users interact with and compare a pair of trees through direct visual overlay (that is, no computational analysis on the merged structure is performed), and allowing two different representation styles to be used – selection of which is based on user preference for the task at hand.

### Summary

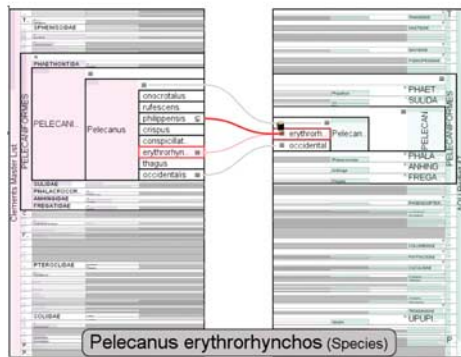
Thus we can see that even for two trees there is a range of possible representation styles that can be utilised to show those trees' inter-relationships. Figure 5 shows a sampling of screenshots for systems that use each of these representation styles, showing the variety of visual forms that the representations of a pair of trees can take. Some lend themselves better to certain structures – the edge drawing and matrix representations can easily show related tree structures that have one-to-many relationships beyond simple node overlapping, while such relationships would remain ambiguous with the colour-coding approach. Similarly for overlapping MultiTree structures the agglomeration approach means simply redrawing structure on top of existing displayed structure.

If set relations rather than individual node matching between trees are of more importance then the colouring approach lends itself well to displaying such data. Of the six published entries for the InfoVis 2003 contest<sup>71</sup> for pairwise comparison of trees, all used a colouring-based linking and brushing approach for at least part of their

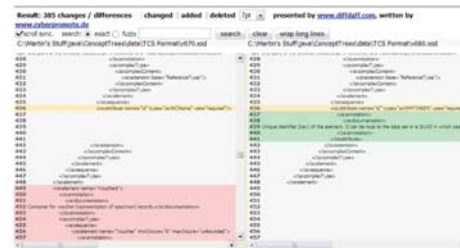




### Edge Drawing



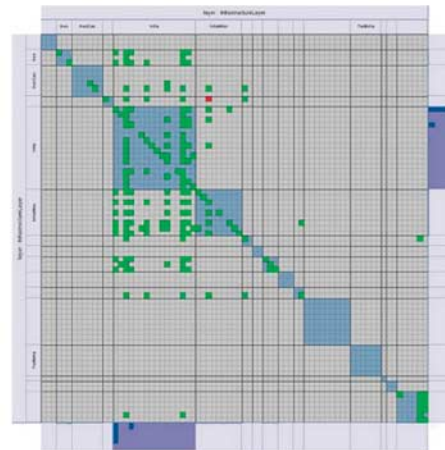
### Colouring



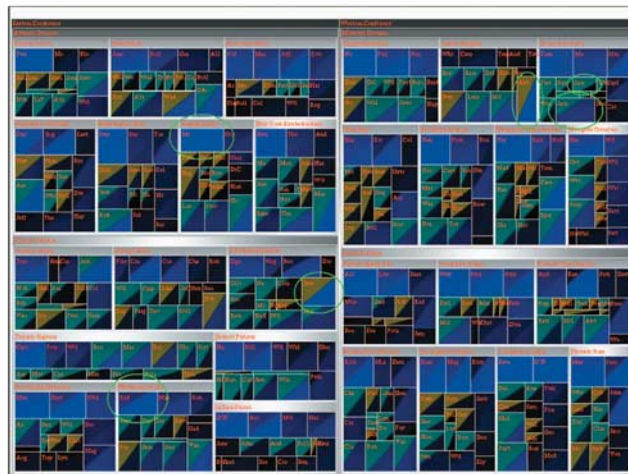
### Animation



### Matrix



### Agglomeration



**Figure 5:** Screenshots of visualisations that show comparisons of two trees. Linking – Craig and Kennedy's<sup>61</sup> Concept Relationship Editor, Colouring – DiffDaff<sup>86</sup> file utility comparing two XML files, Animation – Ghoniem and Fekete's<sup>78</sup> animated treemaps, Matrix – van Ham's code matrix,<sup>82</sup> Agglomeration – Tu and Shen's Union Tree.<sup>83</sup>

solution (entries could include multiple visualisations). Animation between trees was not used in any of the approaches, the technique being reserved for illuminating

focus+context transitions internal to trees. Agglomeration was used in an overview panel presented in Hong *et al*'s Zoomology browser.<sup>69</sup>

## Multiple trees

Multiple tree (> 2) visualisations for the most part extend the two-tree approach to a larger collection of tree structures. Here the pressure builds on available screen space and decisions are needed about whether to show relationships from just one tree to the other trees or to show all relationships between the entire set of trees. Furthermore, other options such as 3D representations for navigating collections of trees may be considered, and when the number of trees becomes too large to display usefully, scatterplot-like representations of tree distances emerge, with trees reduced to individual points

### Edge drawing

This approach, when extended to more than two instances, involves the display of multiple individual representations on-screen, a technique termed as *small multiples* by Tufte.<sup>29</sup> Available screen space is sub-divided into areas in which the individual trees are drawn – an extension of the technique for two trees seen in Figures 4(a) and (b). Then, to draw edges between these multiple tree representations is probably best described as what Parallel Coordinates<sup>87</sup> would resemble if the axes were hierarchical in nature, especially as these visualisations generally do not attempt to draw a many-to-many mapping between all the trees but either a one-to-many mapping or a sequential mapping as seen in Parallel Coordinates. The reasons for not showing a many-to-many mapping in this representation style are that it would firstly produce so many edges intersecting each other and the trees themselves as to be unreadable; secondly, many data sets are specifically visualised in this manner because they have a temporal ordering such that comparison of a particular tree is usually only meaningful to the directly preceding or succeeding trees.

We are not aware of any current general-purpose Parallel Coordinates systems that use hierarchical axes; perhaps the nearest example in appearance is Wernert *et al*'s Tree3D<sup>88</sup> system, based on previous work by Stewart *et al*<sup>89</sup> into visualising multiple phylogenetic trees. This approach lines up multiple phylogenies in a parallel formation and then traces lines between matching nodes in immediately neighbouring phylogenies. Dwyer and Schreiber's<sup>90</sup> later phylogenies visualisation can also be viewed in this style, and includes edge crossing minimisation algorithms to improve readability.

Similarly, Telea and Auber's visualisation of source code evolution<sup>91</sup> uses small multiple representations of code package structures and edges between these representations to indicate the introduction and movement of source code tree changes. Colour is also utilised to pick out particular package grouping or subsets of interest.

Also, Tominski *et al*<sup>92</sup> developed a radial-based layout, VisAxes, for multi-dimensional information. Here, as in Parallel Coordinates, dimensions map one-to-one to a set of axes; a dimension/axis of particular interest is placed

at the centre of the display around which the other axes are arranged. Similarly to Parallel Coordinates, objects are then visualised as poly-lines that plot between the axes set at the appropriate intersections for each axis – though here each poly-line starbursts from the central axis to the others rather than the sequential intersecting that the Parallel Coordinates layout provides. This technique is mentioned because they discuss the possibility of hierarchical axes, and if enough such axes were used, their approach would come under the umbrella of multiple tree visualisations that use edge drawing to show correlations.

### Colouring

It is noticeable that once more than two trees arrive on the scene the dominant approach for showing relationships between individual trees in small multiples is through colour or another highlighting technique. In fact, the majority of the examples referenced in the previous edge drawing section also use colouring in tandem with edge drawing to mark particular subsets. The reasons for this are two-fold.

Firstly, there is the previously described problem of edge crossings and the resulting lack of readability, though alleviated to an extent through edge crossing minimization algorithms. Secondly, edges themselves require space to draw and to visually reroute themselves between trees, but with space at a premium with multiple representations to draw there is pressure to use the more space-efficient tree representations as seen in Figures 3(b)–(d) and to use what space there is for tree structure. Examples that use colouring to show correlations include Munzner *et al*'s TreeJuxtaposer,<sup>58</sup> which can draw multiple linked trees based on dendrograms – a node-link style of layout used for phylogenies. However, internal nodes are not labelled and the allocation for individual nodes can become so compressed that the drawn intra-tree edges may use all the space available for drawing, hence moving towards the adjacency style of representation. Graham and Kennedy<sup>25</sup> use multiple adjacency-layout representations, as do Chi *et al*<sup>93</sup> and Spenke and Beilken,<sup>94</sup> while both Wittenberg *et al*<sup>20</sup> and Kutz<sup>95</sup> use multiple nested layouts to represent their trees. Morse *et al*<sup>96</sup> use multiple indented lists to compare and contrast multiple taxonomic trees. Daida *et al*<sup>97</sup> uses small multiples of individual radial node-link representations to visualise processes in genetic programming, but does not allow direct interaction with the nodes in a tree.

However, even with more efficient single tree visualisations, the 'small multiple' approach does not scale well because of each tree receiving a correspondingly smaller area of screen space as the size of the set grows. There has been research on displaying larger data sets via small multiple representations, but this focuses on larger and larger individual trees<sup>98</sup> rather than a greater number of trees – so far the largest number of trees displayed with this method is approximately 50, for a set of binary trees as seen in Chevenet *et al*'s TreeDyn system,<sup>99</sup> or 14 if we



impose the condition that the tree elements are interactive as in Graham *et al.*<sup>100</sup>

### *Animation*

Animation has further drawbacks when applied to multiple trees rather than just a pair of trees. Here, the number and complexity of trees that can be animated through is not constrained by screen space but by human perceptual abilities; animation can only show at any given moment a change between two trees, tracking a change between multiple trees relies on a user being able to remember the animation's past states. Card *et al.*'s Time-Tree visualisation<sup>101</sup> combines a tree visualisation with temporal data, where changes in a tree structure between different time points are reconciled through animation. Herman *et al.*'s<sup>102</sup> Latour tree drawing system also displays multiple trees through animation, and describes the input data as one initial tree plus a group of 'difference trees' that detail sets of incremental changes to the initial tree structure. This would seem to implicitly recognise that animation is best suited to showing structural evolution rather than complete reorganisations.

Wettel and Lanza<sup>103</sup> use a 'flick-book'-style visualisation by simply switching between a view of one tree to another – however, they preserve the positioning of nodes between views by allocating layout space according to a union of the tree set. Thus a space is allocated in each individual tree view for every node that ever exists throughout the whole set of trees. If a particular tree does not include a node at a particular position then the space is left blank in the associated view.

### *Matrix*

Matrix based visualisation of multiple trees is complicated by the fact that a matrix has only two axes to which an individual tree can be mapped. Multiple trees could be accommodated by extending the multiple scatterplot technique seen in Becker and Cleveland<sup>104</sup> – a visualisation device that allows  $N$ -dimensional data to be rendered as an  $N \times N$  matrix of scatterplots, showing every pairwise combination of variables – to use trees as the basis of comparison rather than dimensions. This would in effect form a matrix of matrices, each of which shows the correlation of two trees against each other, though we are unaware of any current visualisations that do this. In this manner a matrix representation for multiple trees becomes a small multiple display itself, except that each individual representation shows the mapping between two trees rather than just one tree in isolation.

Conversely, the union tree (as defined by Tu and Shen<sup>83</sup>) could be used as a hieraxis for both axes of one matrix, and edges plotted for multiple trees within this single matrix. However, trees that differed significantly in their structures and node overlap would also produce larger and larger union trees and thus correspondingly larger matrices. Further, techniques would then be needed to display and differentiate multi-edges that occupied the

same point in the matrix. Abello and van Ham's Matrix-Zoom system<sup>105</sup> can display matrices containing multi-edges but, rather than distinguish them individually, use colour saturation to indicate the number of edges occupying a particular point in the matrix. The multi-edges in their system are formed by viewing a graph at multiple, hierarchical levels of detail and aggregating edges at each level.

One interesting take on the approach is touched on in Wong *et al.*'s<sup>106</sup> work, in which sketching operations on a matrix representation are used to generate graphs. Their discussion includes an example of how to generate multiple trees through drawing the appropriate matrix representation, though the resulting structure is shown through a traditional node-link graph representation. One limitation is that the same relationship cannot be replicated across trees as their application does not allow the creation of multi-edges.

### *Agglomeration*

Agglomeration of multiple trees means in practice that a node can have multiple parents to display in the same representation, possibly a different parent node per tree. Again the options can include replicating nodes with more than one parent link across the trees to keep the structure as tree-like as possible and amenable to standard tree-drawing techniques. If, though, we wish to represent the multiple tree structure as the truer cycle-containing graph structure, which nested, indented and adjacency layouts cannot display, then agglomerative representations of multiple trees are generally displayed using node-link representations, such as in Florentz and Muecke's GLAD system.<sup>107</sup> Some do try a different tack though, Mank's CristalView<sup>108</sup> uses force-directed placement of overlapping TreeMaps to communicate shared nodes across multiple hierarchies. Burch and Diehl<sup>109</sup> also break the node-link domination to a degree by displaying a TreeMap of a reference taxonomy on top of which are overlaid multiple node-link trees. The nodes in each individual tree are instances of object classes in the reference taxonomy and are thus placed on top of the TreeMap at corresponding positions, with the edges between contorting to connect the appropriate points.

Care must be taken when displaying multiple tree structures using general node-link graph drawing techniques; to begin with multi-edges resulting from consistent edges across different trees need to be distinguished by some method, but most general graph-drawing toolkits do not provide support for this. A general graph layout method also usually results in no global orientation for child-parent links even if one exists in the overall structure. Techniques specifically developed for drawing DAGs can re-impose a global orientation for parent-child links, although the restrictions on node placement involves a trade-off on edge crossings as seen in Graham and Kennedy,<sup>57</sup> D'Ambros and Lanza<sup>110</sup> and Melançon and Herman.<sup>111</sup> The main advantage with agglomeration

displays is that screen space is effectively re-used across tree representations. There is no technical upper limit to how many trees can be displayed through agglomeration, though eventually known perceptual obstacles found in graph drawing caused by edge-crossing and occlusion of nodes and edges will lead to difficulty in interpreting the structure usefully.

### 3D representations

A popular compromise that fuses elements of the previous approaches is to use a 3D representation of multiple trees. These generally take the form of multiple, distinct tree representations drawn in parallel planes to each other. Relationships between the trees are shown again either by drawing edges between trees as in Dadzie and Burger,<sup>26</sup> Dwyer and Schreiber<sup>90</sup> and Wernert *et al*,<sup>88</sup> or by using colouring as in Chi *et al*.<sup>93</sup> The 3D approach means the group of trees can be rotated so that they resemble a small multiple display of multiple trees (one tree per section of screen space), or turned ninety degrees so the trees give the impression of overlaying one another. This feature does have the drawback of not guaranteeing that equivalent nodes in different trees will overlay each other, and can lead to a display with a high degree of occlusion. Reiss' early work on software visualisation<sup>112</sup> used 3D to show a merged structure of three different hierarchies, in which looking at any of the  $xy/xz/yz$  planes head-on would reveal one individual hierarchy.

### Atomic representations

Finally, when the number of trees grows extremely large, the finite screen space cannot show all the trees at any sensible level of detail, so techniques have been developed that visualise the trees as atomic items, from which examples can be viewed in detail. Amenta and Klingner<sup>113</sup> and Hillis *et al*<sup>114</sup> take this approach by visualising a set of phylogenies in a scatterplot, where distances between points relate to the degree of similarity between the associated trees. Meyer<sup>115</sup> proposes a similar scheme, whereby the trees form nodes in a hierarchical graph, connected by edges according to their similarity. Parts of the graph and, from there, individual trees or a consensus tree of a tree group can then be interrogated to display further details and to reveal related phylogenies. Both these applications state that consensus trees lose data from the individual trees that compose them, and so the ability to see the individual phylogenies is crucial.

### Summary

Again, as with the two-tree scenario, the type of overall structure the multiple trees form will have a strong bearing on the visualisation techniques that will be required to effectively visualise the data. An unrelated forest of trees will obviously be easily, and perhaps only, represented as separate visual entities, while visually overlaying trees – an agglomeration layout – might benefit those that share

many multitree-like sub-structures between themselves. Trees that construct their own structure over shared nodes, forming polyarchies or DAGs, are more problematic as the differing tree structures produce significant extra edge-crossings in the agglomeration style views. Finally, trees that are related through non-overlapping node relationships will favour a representation that does not emphasise node-sharing but instead has the ability to show one-to-many relationships between trees. This will favour the edge drawing and matrix representations over the agglomeration based techniques. Figure 6 gives a selection of visualisations for the various multiple tree visualisations, with the omission of the as-yet unimplemented matrix representation.

Table 2 describes a matrix of representations by possible features to highlight the strengths of each representation, including the ability to show node mappings between trees, and the ability to show types of structural changes – specifically the addition, deletion and reorganising of nodes between trees. For instance, finding one-to-many relationships is easier in the edge drawing representation than in the colouring metaphor, while showing new and deleted nodes is easier in the colouring approach than with the edge-drawing representations. Those categories where capabilities for a task in a given representation have been proven are shaded. The 3D category is omitted as much of what 3D visualisations are capable of depends on the individual representations used for the trees. Situations are also marked where there are, to our knowledge, no current representations – such as many in the matrix category.

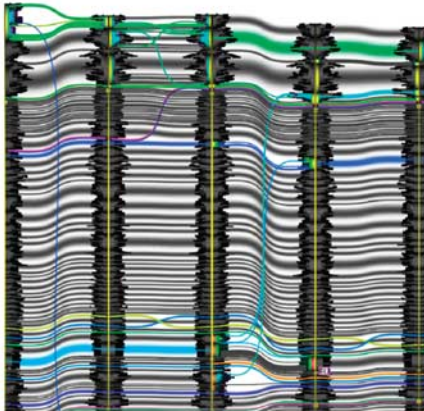
In summary, representations for multiple trees are based on the representations described for two trees, but the drawbacks and advantages of each become more exaggerated as more trees are considered, until atomic representations of trees become the only viable option and more detailed views are only considered upon reducing the tree set size or zooming in on a particular tree instance. Matrix representations appear to be under-exploited even though they are gathering more attention for general graph display due to their removal of edge-crossing difficulties found in general node-link representations. For multiple tree structures that include cycles that hamper the application of certain representations or feature relations that induce edge-crossings in node-link diagrams, perhaps this is one potential avenue of future exploration. To balance against this, matrices have well-known drawbacks in path navigation that are also their main weakness in general graph drawing and will require mechanisms for dealing with multi-edges that occur frequently in multiple tree sets.

### Tasks

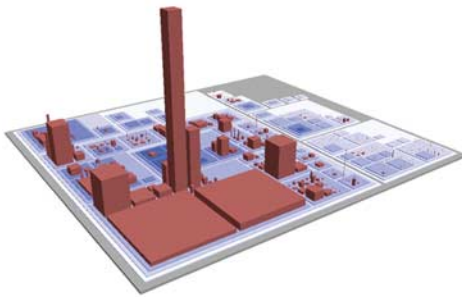
The literature reveals three main high-level tasks that multiple tree visualisations attempt to tackle in order to allow users to use, create and understand



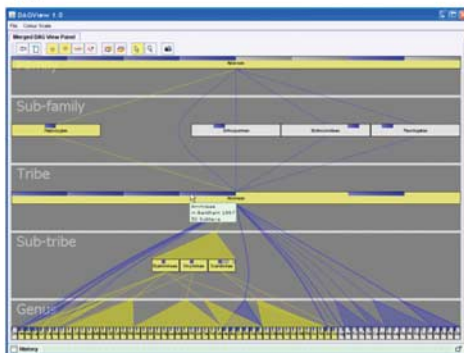
### Edge drawing



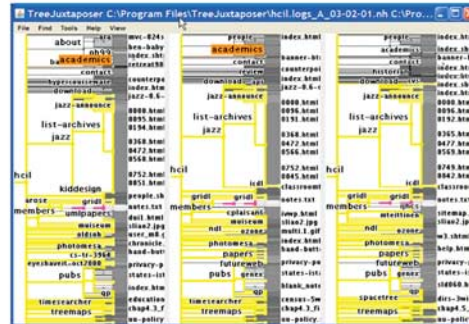
### Animation



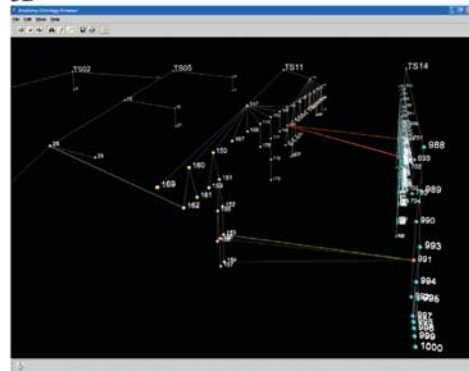
### Agglomeration



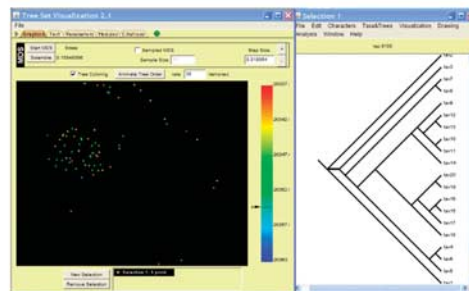
### Colouring



### 3D



### Atomic



**Figure 6:** Screenshots of systems representing the different representation styles for multiple trees. Edge drawing – Telea and Auber's<sup>91</sup> CodeFlow application, Colouring – Munzner *et al*'s TreeJuxtaposer,<sup>58</sup> Animation – Wettel and Lanza's<sup>103</sup> CodeCity representation, 3D – Dadzie and Berger's<sup>26</sup> mouse anatomy ontology viewer, Agglomeration – Graham and Kennedy's<sup>57</sup> DAG viewer and Atomic – Hillis *et al*'s TreeSet visualization<sup>114</sup> (screenshot from the TreeSet module running under the Mesquite software system<sup>116</sup>).

multi-hierarchical data.

1. Filtering of data through multiple hierarchical categories – faceted hierarchy browsing or data-cube querying.
2. Mapping of relationships between multiple hierarchies – such as allowing users to edit machine-produced mappings between ontologies or taxonomic data.

3. Exploration of differences between multiple hierarchies, structurally or in terms of node properties.

Many of the visualisations that fall under the umbrella of data filtering do not wish to compare multiple hierarchies, but instead aim to offer a straightforward way of navigating through the structure formed by multiple hierarchies to reach the data sitting at the leaves. As



**Table 2:** Strengths and weaknesses for representation of multiple trees

Representation	1:1 Node Matching	M:N Node Matching	Extends To N Trees	Structural Changes	Attribute Changes
Small Multiples – Edge Drawing	Yes, edges link between nodes in different trees.	Multiple edges emanating or arriving at a particular node.	Poorly. Could be extended in a hierarchical parallel coordinate style but edge and tree crossings become an issue. Screen space also reduces in proportion to the number of trees.	Yes, for subtrees as edges plot relatively to each other. Addition and deletion more problematic as edges cannot be drawn to or from a non-existent node.	No, edge drawing relates only to structural changes and consistencies between trees
Small Multiples – Coloured	Colouring is a set-based action, so it takes further interaction techniques to discern specific 1:1 relationships in a selection.	Indistinguishable from the 1:1 state.	Yes, selections in 1 tree can be marked by colour in N other trees, though again screen space reduces in proportion to the number of trees.	New and deleted nodes generally coloured as a specific user function. Distribution of blocks of colour can show dispersal of sub-tree across wider structure.	Yes, if colour coding can be assigned to specific attributes and traits, differences between trees can then be seen.
Animation	Can be done by preserving node position between frames or animating between successive positions.	Unknown. Would require animation of node aggregation/splitting.	Pairwise in sequence comparison. Cognitive effort required to remember previous animation states.	Yes, but difficult to track multiple changes, best used when changes are gradual or only a few nodes are preserved between trees.	Dependent on method used to communicate attribute values. Change in e.g. colour space and even size is generally less noticeable than change in position.
Agglomeration	Yes, achieved by spatially overlapping nodes from different trees	Can be done by drawing non-hierarchical edges in addition to hierarchical relationships	Yes, matching parts of tree structure are reinforced in the representation and extra edges used to show new structure or restructuring.	Yes, changes in parents or children between trees will be shown by extra edges in representation. Multi-edge representations critical for showing constancies.	Only if attribute changes can be aggregated and summarised within one node representation – this is usually precluded by space constraints.
Matrix	Yes, 1 point per row or column between two trees on the axes.	Yes, fill in multiple points per row or column.	Possible by building a multiple scatterplot-style matrix or using a union tree of all trees as the axes.	Yes, matrices show edges between trees, a change in structure between any two particular trees would manifest itself as a divergence from a diagonal plot of edges in the matrix.	No, the matrix relates only relationships between the trees; though it is conceivable those relationships could be couched in terms of attribute value changes.
Atomic	No, tree viewed as an atomic object.	No, tree viewed as an atomic object.	Yes, into the hundreds or thousands, trees that occupy ‘interesting’ positions in the space can then be rendered individually.	Yes for global change, if distance metric used in the layout of tree ‘points’ is indicative of a global degree of difference as in Hillis <i>et al</i> <sup>73</sup> .	Possible, if distance metric used to layout tree ‘points’ relates in some way to node attribute values

such they tend not to offer complex or multiple views of the data set, but a current view of where navigation has led to and indications of where immediate navigation forward or backward could lead to. Representations of this task are either a single hierarchy composed of the current and possible further filtering categories or a set of extremely flat hierarchy representations; often only one or two levels of a hierarchy are displayed even if deeper sub-categories exist.

The second high-level task, mapping relationships between multiple hierarchies, is in all cases done on a pair-wise basis. Apart from Wong *et al*'s<sup>106</sup> matrix-based sketcher this task is always carried out by representing the hierarchies in question as two individual representations, mostly in the indented list style (and never as a nested representation), between which relationships are shown by lines or arcs acting as links. This method is preferred in fields such as ontology alignment and taxonomic concept mapping as it can show more complex relationships than simple 1:1 relationships between nodes in different trees – and it is these one-to-many or many-to-many relationships that tend to need expert intervention to specify accurately when the simpler relationships have mostly been resolved algorithmically. Commercial products such as MapForce<sup>11</sup> and research prototypes such as SchemaMapper<sup>65</sup> all adhere to this template of representation and interaction.

The final high-level task, comparing multiple hierarchies to find changes in structure or node properties, is the task that has produced the most varied collection of representations as researchers strive to project the richness and complexity of the inter-relationships between multiple trees, while at the same time attempting to keep

the basic representations intelligible, and can be divided into a number of different tasks – comparing node attributes, finding structural reorganisations or locating node deletion and addition.

Finding differences in node attributes inevitably involves either a small multiple display or animation approach as the differences between successive trees need to be shown in their own area either spatially or temporally. Out of these, the edge drawing approach is usually disregarded as attributes are usually encoded using colour or size as in Treemaps, and edge drawing is almost universally reserved for showing structure reorganisation.

Finding changes in structure depends on the type and detail of change we wish to see. The overlap of common structure can be seen by colouring in small multiples. If we are interested in finding re-classification of existing structure, the most prominent style appears to be some form of agglomeration representation or edge drawing approach, as divergent edges between trees can quickly be seen in the display of the merged structure – though as stated before too many changes can lead to problems with edge-crossings in node-link displays. Addition and deletion of nodes can be seen in general in coloured representations if a function has been implemented to encode such nodes with specific colours, such as found in Tree-Juxtaposer. Edge drawing techniques have difficulty here as if a node is freshly minted or now removed, there is nothing to either draw a connecting edge from or to. (The same problem occurs in parallel coordinates when a null value occurs for an item in a particular value.)

Unlike for single trees, there are limited user studies in comparing multiple tree representations, and those that have occurred have been small in scale. Lee *et al*<sup>85</sup>



compared a node-link agglomerated display of two trees against Microsoft's WinDiff.<sup>72</sup> tool and found preference among software engineers for their CandidTree interface. Graham and Kennedy<sup>117</sup> found that taxonomists preferred a small multiple representation of a tree set linked by colouring to an agglomerated graph representation, and still preferred the small multiple approach to a DAG representation that preserved child-parent orientation. However, these were both studies for specific fields and these preferences may be ingrained to a particular mindset or tasks. Multiple tree research still lacks an equivalent study to Barlow and Neville's,<sup>118</sup> Kobsa's<sup>41</sup> or Andrews and Kasanicka's<sup>119</sup> evaluations across a gamut of single tree visualisation types. Parunak<sup>120</sup> argued that people were best prepared to think of multiple classifications as individual, intersecting entities rather than a merged whole, and this might give a hint as to why the small multiple approaches appear to be the dominant metaphor in multiple tree visualisation.

## Conclusion

This review of current work demonstrates that multiple tree visualisation is still an open research topic. Even for single trees, research is still being published on different, novel ways of displaying and interacting with trees and hierarchies, with techniques designed to accommodate certain user groups and tasks. Visualising multiple hierarchies adds an extra level of complexity, as representations of multiple trees cover a wider breadth of display possibilities than representations for single instances or even pairs of trees. Layouts for general and layered graph drawing enter into consideration as well as interaction techniques such as linking and brushing for discovering correlations between trees.

The complexity of the overall structure varies depending on the inter-relationships between the individual hierarchies, on a spectrum of no overlap whatsoever to DAGs, onto polyarchies, and then through to structures that have extra non-trivial relationships between nodes – this can and does affect the particular choice of layout and techniques used in a multiple tree visualisation.

Consideration of tasks also narrows the possible range of representations; human-assisted mapping between trees is done exclusively on a pair-wise fashion between individual tree representations. Navigation of multiple trees involves displaying as little of the complexity of the structure as possible and keeping the navigation choices down to the next one or two immediately accessible levels in each hierarchy. For more involved tasks, such as discovering differences in structure between trees, increasingly detailed and varied visualisations have been considered. In these circumstances, research so far has shown that developers and users prefer when possible to reduce visual complexity by keeping the individual tree structures visually separate, even if the underlying data model is a fusion of many trees. The layout design space has not been fully

explored by existing visualisations; matrix-style layouts are noticeable by their absence in the literature.

No conclusive user studies have yet been performed comparing the various types of possible multiple tree visualisation. Those small studies that have occurred were based on small user samples, self-assessment as in the InfoVis 2003 competition, or a particular type of data.

The situation is exacerbated by the fact that as stated multiple trees can form different classes of structure. While single tree comparative evaluation can rely on a tree being a tree, multiple tree evaluation will have to accommodate numerous types of structures from multitrees to polyarchies and consider whether systems under comparison are being compared like with like. It would, for instance, not make sense to ask a visualisation designed to show multitrees to handle structures with more complex relationships and then judge its performance against another system based on that capability. As such, any experiments to show which representation or systems are best for particular tasks will have to be doubly careful about choosing a data set.

## Acknowledgements

We thank EPSRC for providing the funding (Grant no. EP/D052629/1) through which this article was produced, and the expert reviewers for providing essential feedback on previous drafts of this article. We also thank those researchers who were kind enough to grant permission to use screenshots of their visualisations in this article.

## References

- Noik, E.G. (1994) A space of presentation emphasis techniques for visualizing graphs. In: W.A. Davis and J. Barry (eds.) *Graphics Interface* Banff, Alberta, Canada: Morgan-Kaufmann Publishers, pp. 225–234.
- Herman, I., Melançon, G. and Marshall, M.S. (2000) Graph visualisation and navigation in information visualisation: A survey. *IEEE Transactions on Visualization and Computer Graphics* 6(1): 24–43.
- Nguyen, Q.V. (2005). Space efficient visualisation of large hierarchies. PhD Thesis, Faculty of Information Technology, University of Technology, Sydney, Australia, pp. 171+xix.
- Nussbaumer, A. (2005). Hierarchy browsers. Master's Thesis, Institute for Information Systems and Computer Media (IICM), Graz University of Technology, Graz, Austria, p. 109.
- Graham, M. (2001). Visualising multiple overlapping classification hierarchies. PhD Thesis, School of Computing, Napier University, Edinburgh, UK, p. 182.
- Steel, M., Dress, A.W.M. and Böcker, S. (2000) Simple but fundamental limitations on supertree and consensus tree methods. *Systematic Biology* 49(2): 363–368.
- Priss, U. (2008) Facet-like structures in computer science. *Axiomathes* 18(2): 243–255.
- Sifer, M. (2006) Filter co-ordinations for exploring multi-dimensional data. *Journal of Visual Languages and Computing* 17(2): 107–125.
- Shvaiko, P. and Euzenat, J. (2008) Ontology matching [Website], <http://www.ontologymatching.org/index.html>, accessed 16 December 2008.



- 10 Aumueller, D., Do, H.-H., Massmann, S. and Rahm, E. (2005) Schema and ontology matching with COMA++. In: J. Widom, F. Özcan and R. Chirkova (eds.) *ACM SIGMOD* Baltimore, MD: ACM Press, pp. 906–908.
- 11 Altova. GmbH MapForce (Version 2008 rel. 2 sp. 2)[Computer Software], <http://www.altova.com/downloadtrialmapforce3.html>, accessed 16 December 2008.
- 12 Wang, G., Rifaieh, R., Goguen, J., Zavesov, V., Rajasekar, M. and Miller, M. (2007). Towards User Centric Schema Mapping Platform. International Workshop on Semantic Data and Service Integration, Vienna, Austria.
- 13 Eick, S.G., Steffen, J.L. and Sumner Jr, E.E. (1992) Seesoft – A tool for visualizing line oriented software statistics. *IEEE Transactions on Software Engineering* 18(11): 957–968.
- 14 Girba, T., Lanza, M. and Ducasse, S. (2005) Characterizing the evolution of class hierarchies. 9th European Conference on Software Maintenance and Reengineering – CSMR, Manchester, UK. Los Alamitos: IEEE Computer Society Press, pp. 2–11.
- 15 Wu, X., Storey, M.-A., Murray, A. and Lintern, R. (2003) Visualization to support version control software: Suggested requirements. In: A.v. Deursen, C. Knight, J.I. Maletic and M.-A. Storey (eds.) *VisSoft*. Amsterdam, Netherlands, pp. 80–86.
- 16 Fuchsberger, C., Falchi, M., Forer, L. and Pramstaller, P.P. (2008) PedVizApi: A Java API for the interactive, visual analysis of extended pedigrees. *Bioinformatics* 24(2): 279–281.
- 17 Storey, M.-A., et al (2001) Jambalaya: Interactive Visualization to Enhance Ontology Authoring and Knowledge Acquisition in Protégé. Workshop on Interactive Tools for Knowledge Capture (K-CAP), Victoria, British Columbia, Canada.
- 18 McGuffin, M. and Schraefel, M.C. (2004) A comparison of hyperstructures: Zstructures, mSpaces, and Polyarchies. In: J. Whitehead and D. De Roure (eds.) *ACM Hypertext*. Santa Cruz, CA: ACM Press, pp. 153–162.
- 19 Furnas, G.W. and Zacks, J. (1994) Multitrees: Enriching and reusing hierarchical structure. In: B. Adelson, S. Dumais and J. Olson (eds.) *ACM CHI*. Boston, MA: ACM Press, pp. 330–336.
- 20 Wittenburg, K., Das, D., Hill, W. and Stead, L. (1995) Group asynchronous browsing on the world wide web. In: C. Irving (ed.) Fourth International World Wide Web Conference 1995. Boston, MA: O'Reilly, pp. 51–62.
- 21 Sifer, M. (2003) Exploring web site log data with a multi-classification interface. In: E. Banissi, F. Khosrowshahi, M. Sarfraz and A. Ursyn (eds.) IEEE Conference on Information Visualisation. Los Alamitos, London, UK: IEEE Computer Society Press, pp. 94–101.
- 22 McGuffin, M. and Balakrishnan, R. (2005) Interactive visualization of genealogical graphs. In: J. Stasko and M.O. Ward (eds.) *IEEE InfoVis*. Minneapolis, MN: IEEE Computer Society Press, pp. 17–24.
- 23 Robertson, G., Cameron, K., Czerwinski, M. and Robbins, D. (2002) Animated visualization of multiple intersecting hierarchies. *Information Visualization* 1(1): 50–65.
- 24 Conklin, N., Prabhakar, S. and North, C. (2002) Multiple foci drill-down through tuple and attribute polyarchies in tabular data. In: P.C. Wong and K. Andrews (eds.) *IEEE InfoVis*. Boston, MA: IEEE Computer Society Press, pp. 131–134.
- 25 Graham, M. and Kennedy, J. (2005) Extending taxonomic visualisation to incorporate synonymy and structural markers. *Information Visualization* 4(3): 206–223.
- 26 Dadzie, A.-S. and Burger, A. (2005) Providing visualisation support for the analysis of anatomy ontology data. *BMC Bioinformatics* 6(74) n/a.
- 27 Cruz, I.F., Sunna, W., Makar, N. and Bathala, S. (2007) A visual tool for ontology alignment to enable geospatial interoperability. *Journal of Visual Languages and Computing* 18(3): 230–254.
- 28 Tu, Q. and Godfrey, M.W. (2002) An Integrated Approach for Studying Architectural Evolution. Paris, France: IEEE Computer Society Press, 10th International Workshop on Program Comprehension (IWPC), pp. 127–136.
- 29 Tufte, E.R. (1983) *The Visual Display of Quantitative Information*. Cheshire, CO: Graphics Press, p. 197.
- 30 Reingold, E.M. and Tilford, J.S. (1981) Tidier drawing of trees. *IEEE Transactions on Software Engineering* 7(2): 223–228.
- 31 Beebe, N.H.F. (2006) A bibliography of tree drawing algorithms. Bibliography, Department of Mathematics, University of Utah, Salt Lake City, pp. 22, <http://www.math.utah.edu/pub/tex/bib/trees.ps.gz>.
- 32 Moen, S. (1990) Drawing dynamic trees. *IEEE Software* 7(4): 21–28.
- 33 Johnson, B. and Shneiderman, B. (1991) Treemaps: A space-filling approach to the visualization of hierarchical information structures. In: G.M. Nielson and L.J. Rosenblum (eds.) *IEEE Visualization*. San Diego, CA: IEEE Computer Society Press, pp. 284–291.
- 34 Wang, W., Wang, H., Dai, G. and Wang, H. (2006) Visualization of large hierarchical data by circle packing. In: R.E. Grinter, T. Rodden, P.M. Aoki, E. Cutrell, R. Jeffries and G.M. Olson (eds.) *ACM CHI Montréal*, Québec, Canada: ACM Press, pp. 517–520.
- 35 Kruskal, J.B. and Landwehr, J.M. (1983) Icicle plots: better displays for hierarchical clustering. *The American Statistician* 37(2): 162–168.
- 36 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.
- 37 Robertson, G.G., Mackinlay, J.D. and Card, S.K. (1991) Cone trees: Animated 3D visualizations of hierarchical information. *ACM CHI: Human Factors in Computing Systems*. New Orleans, LA: ACM Press, pp. 189–194.
- 38 Bladh, T., Carr, D.A. and Scholl, J. (2004) Extending tree-maps to three dimensions: A comparative study. *6th Asia-Pacific Conference on Computer-Human Interaction, Rotorua, New Zealand*. Berlin, Heidelberg: Springer-Verlag, pp. 50–59.
- 39 van Ham, F. and van Wijk, J.J. (2002) Beamtrees: Compact visualization of large hierarchies. In: P.C. Wong and K. Andrews (eds.) *IEEE InfoVis*. Boston, MA: IEEE Computer Society Press, pp. 93–100.
- 40 Cockburn, A. and McKenzie, B. (2000) An evaluation of cone trees. In: S. McDonald, Y. Waern and G. Cockton (eds.) *BCS HCI*. Sunderland, London, UK: Springer-Verlag, pp. 425–436.
- 41 Kobsa, A. (2004) User experiments with tree visualization systems. In: M.O. Ward and T. Munzner (eds.) *IEEE InfoVis*. Austin, TX: IEEE Computer Society Press, pp. 9–16.
- 42 Shen, Z. and Ma, K.-L. (2007) Path visualization for adjacency matrices. In: K. Museth, T. Möller and A. Ynnerman (eds.) *Eurographics/IEEE-VGTC Symposium on Visualization*. Norrköping, Sweden. Aire-la-Ville, Switzerland: Eurographics, pp. 83–90.
- 43 Lü, H. and Fogarty, J. (2008) Cascaded Treemaps: Examining the visibility and stability of structure in treemaps. *Graphics Interface*. Windsor, Ontario, Canada: Canadian Information Processing Society, 322, pp. 259–266.
- 44 Zhao, S., McGuffin, M.J. and Chignell, M.H. (2005) Elastic hierarchies: Combining treemaps and node-link diagrams. *IEEE InfoVis*. Minneapolis, MN: IEEE Computer Society Press, pp. 57–64.
- 45 Nguyen, Q.V. and Huang, M.L. (2005) EncCon: An approach to constructing interactive visualization of large hierarchical data. *Information Visualization* 4(1): 1–21.
- 46 Wilson, R.M. and Bergeron, R.D. (1999) Dynamic hierarchy specification and visualization. *IEEE InfoVis*. San Francisco, CA: IEEE Computer Society Press, pp. 65–72.
- 47 Urbanek, S. (2002) Different ways to see a tree – KLIMT. In: W. Härdle and B. Rönz (eds.) 15th Conference on Computational Statistics, COMPSTAT. Berlin, Germany, Heidelberg: Physica-Verlag, pp. 303–308.
- 48 Schedl, M., Knees, P., Widmer, G., Seyerlehner, K. and Pohle, T. (2007) Browsing the web using stacked three-dimensional sunbursts to visualize term co-occurrences and multimedia content. In: G. Kindlmann and L. Linsen (eds.) *IEEE Visualization*. Sacramento, CA: Poster Compendium, IEEE Computer Society Press, pp. 2–3.
- 49 Teoh, S.T. (2007) A study on multiple views for tree visualization. In: R.F. Erbacher, J.C. Roberts, M.T. Gröhn and K. Börner (eds.) *Visualization and Data Analysis*. Bellingham, San Jose, CA: SPIE Press, 6495, pp. 99–110.



- 50 Kules, B., Shneiderman, B. and Plaisant, C. (2003) Data exploration with paired hierarchical visualizations: Initial designs of pairtrees. National Conf. on Digital Government Research, Boston, MA, 130 – ACM International Conference Proceeding Series, Digital Government Research Center.
- 51 Mohammadi-Aragh, M.J. and Jankun-Kelly, T.J. (2005) MoireTrees: Visualization and interaction for multi-hierarchical data. Eurographics/IEEE VGTC Symposium on Visualization. Leeds, UK: Eurographics, pp. 231–238.
- 52 Weiland, M. and Dachselt, R. (2008) Facet folders: Flexible filter hierarchies with faceted metadata. *ACM CHI*. Florence, Italy: ACM Press, pp. 3735–3740.
- 53 Dachselt, R. and Frisch, M. (2007) Mambo: A facet-based zoomable music browser. In: T. Ojala and M. Ylianttila (eds.) 6th international conference on Mobile and Ubiquitous Multimedia. Oulu, Finland. New York: ACM Press, pp. 110–117.
- 54 Smith, G., Czerwinski, M., Meyers, B., Robbins, D., Robertson, G. and Tan, D.S. (2006) FacetMap: A scalable search and browse visualization. *IEEE Transactions on Visualization and Computer Graphics* 12(5): 797–804.
- 55 Lee, B., Smith, G., Robertson, G.G., Czerwinski, M. and Tan, D.S. (2009) FacetLens: Exposing trends and relationships to support sensemaking within faceted datasets. *ACM CHI*. Boston, MA: ACM Press, pp. 1293–1302.
- 56 Yee, K.-P., Swearingen, K., Li, K. and Hearst, M. (2003) Faceted metadata for image search and browsing. *ACM CHI*. Fort Lauderdale, FL: ACM Press, pp. 401–408.
- 57 Graham, M. and Kennedy, J. (2007) Exploring multiple trees through DAG representations. *IEEE Transactions on Visualization and Computer Graphics* 13(6): 1294–1301.
- 58 Munzner, T., Guimbretière, F., Tasiran, S., Zhang, L. and Zhou, Y. (2003) TreeJuxtaposer: Scalable tree comparison using focus+context with guaranteed visibility. *ACM Transactions on Graphics* 22(3): 453–462.
- 59 Sheth, N., Börner, K., Baumgartner, J., Mane, K. and Wernert, E. (2003) Treemap radial tree and 3D tree visualizations. *IEEE InfoVis Poster Compendium*. Seattle, Washington: IEEE Computer Society Press, pp. 128–129.
- 60 Wan Zainon, W.N. and Calder, P. (2006) Visualising phylogenetic trees. Seventh Australasian User Interface Conference, Hobart, Australia: Australian Computer Society, 50, pp. 145–152.
- 61 Craig, P. and Kennedy, J. (2008) Concept relationship editor: A visual interface to support the assertion of synonymy relationships between taxonomic classifications. In: K. Börner, M.T. Gröhn, J. Park and J.C. Roberts (eds.) *Visualization and Data Analysis*. San Jose, CA: SPIE Press, 6809, p. 12.
- 62 Chiticariu, L., Hernández, M.A., Kolaitis, P.G. and Popa, L. (2007) Semi-automatic schema integration in clio. *VLDB*. Austria, Vienna: ACM Press, pp. 1326–1329.
- 63 Purchase, H. (1997) Which aesthetic has the greatest effect on human understanding? G. Di Battista, (ed.) *Graph Drawing*. Rome, Italy, LNCS 1353. Berlin, Heidelberg, Springer-Verlag, pp. 248–261.
- 64 Eades, P. and Wormald, N.C. (1994) Edge crossings in drawings of bipartite graphs. *Algorithmica* 11(4): 379–403.
- 65 Robertson, G. G., Czerwinski, M. P. and Churchill, J. E. (2005) Visualization of mappings between schemas. In: G.C. van der Veer and C. Gale (eds.) *ACM CHI*. Portland, OR: ACM Press, pp. 431–439.
- 66 Holten, D. and van Wijk, J.J. (2008) Visual comparison of hierarchically organized data. *Computer Graphics Forum* 27(3): 759–766.
- 67 Buchin, K., et al (2008) Drawing (Complete) binary tanglegrams: Hardness, approximation, fixed-parameter tractability. In: I.G. Tollis and M. Patrignani (eds.) *Graph Drawing*. Heraklion, Crete, Greece, LNCS 5417. Berlin, Heidelberg: Springer-Verlag, pp. 324–335.
- 68 Parr, C.S., Lee, B., Campbell, D. and Bederson, B.B. (2004) Visualizations for taxonomic and phylogenetic trees. *Bioinformatics* 20(17): 2997–3004.
- 69 Hong, J. Y., D'Andries, J., Richman, M. and Westfall, M. (2003) Zoomology: Comparing two large hierarchical trees. In: C. Plaisant and J.-D. Fekete (eds.) *IEEE InfoVis Poster Compendium*. Seattle, Washington: IEEE Computer Society Press, pp. 120–121.
- 70 Auber, D., Delest, M., Domenger, J. P., Ferraro, P. and Strandh, R. (2003) EVAT: Environment for visualization and analysis of trees. In: C. Plaisant and J.-D. Fekete (eds.) *IEEE InfoVis Poster Compendium*. Seattle, Washington: IEEE Computer Society Press, pp. 124–125.
- 71 Information Visualization Benchmarks Repository. (2003) InfoVis 2003 Contest – Visualization and PairWise Comparison of Trees [Webpage], <http://www.cs.umd.edu/hcil/InfovisRepository/contest-2003/>, accessed 20 July 2009.
- 72 Microsoft Corp. (2008) WinDiff (Version 5.1) [Computer Software], <http://support.microsoft.com/kb/159214>, accessed 15 December 2008.
- 73 Snyder, J., Dobbe, W., Keel, J., Bruggeman, O. and Firebaugh, J. (2008) Kompere (Version 3.5) [Computer Software], <http://www.kde.org/download/>, accessed 15 December 2008.
- 74 Altova GmbH (2008) DiffDog (Version 2008 rel. 2 sp. 2) [Computer Software], [http://www.altova.com/download/diffdog/diff\\_merge\\_tool.html](http://www.altova.com/download/diffdog/diff_merge_tool.html), accessed 15 December 2008.
- 75 David, J., Guillet, F., Gras, R. and Briand, H. (2006) An interactive, asymmetric and extensional method for matching conceptual hierarchies. EMOI-INTEROP – Open Interop Workshop on Enterprise Modelling and Ontologies for Interoperability, Luxembourg, 200, CEUR-WS.org.
- 76 Bossung, S., Stoeckle, H., Grundy, J., Amor, R. and Hosking, J. (2004) Automated data mapping specification via schema heuristics and user interaction. In: V. Wiels and K. Stirewalt (eds.) 19th IEEE International Conference on Automated Software Engineering (ASE). Linz, Austria. Los Alamitos: IEEE Computer Society Press, pp. 208–217.
- 77 Raghavan, A. (2005) Schema mapper: A visualization tool for incremental semi-automatic mapping-based integration of heterogeneous collections into archaeological digital libraries: The ETANA-DL case study. Master's Thesis, Dept of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, USA, pp. 66+vii.
- 78 Ghoniem, M. and Fekete, J.-D. (2008) Animating Treemaps. 18th HCI Symposium – Workshop on Treemap Implementations and Applications 2001, University of Maryland, College Park, Maryland, USA.
- 79 Wittenburg, K. and Sigman, E. (1997) Visual focusing and transition techniques in a treeviewer for web information access. In: G. Tortora (ed.) *Visual Languages*. Capri Italy: IEEE Computer Society Press, pp. 20–27.
- 80 Wilkinson, L. and Friendly, M. (2009) The history of the cluster heat map. *The American Statistician* 63(2): 179–184.
- 81 Eisen, M. (2002) TreeView (Version 1.60) [Computer Software], <http://rana.lbl.gov/downloads/TreeView/>.
- 82 van Ham, F. (2003) Using multilevel call matrices in large software projects. *IEEE InfoVis*. Seattle, Washington: IEEE Computer Society Press, pp. 227–232.
- 83 Tu, Y. and Shen, H.-W. (2007) Visualizing changes of hierarchical data using treemaps. *IEEE Transactions on Visualization and Computer Graphics* 13(6): 1286–1293.
- 84 Lee, B., Robertson, G.G., Czerwinski, M. and Parr, C.S. (2007) CandidTree: Visualizing structural uncertainty in similar hierarchies. *Information Visualization* 6(3): 233–246.
- 85 Isenberg, P. and Carpendale, S. (2007) Interactive tree comparison for co-located collaborative information visualization. *IEEE Transactions on Visualization and Computer Graphics* 13(6): 1232–1239.
- 86 Cyberpromote (2009) DiffDaff (Version 1.0) [Computer Software], <http://www.diffdaff.com/>, accessed 3 April 2009.
- 87 Inselberg, A. and Dimsdale, B. (1990) Parallel coordinates: A tool for visualizing multidimensional geometry. In: A.E. Kaufman (ed.) *IEEE Visualization*. San Francisco, CA. Los Alamitos: IEEE Computer Society Press, pp. 361–378.
- 88 Wernert, E.A., Berry, D.K., Huffman, J.N. and Stewart, C.A. (2003) Tree3D – A system for temporal and comparative analysis of phylogenetic trees. In: C. Plaisant and J.-D. Fekete (eds.) *IEEE InfoVis Poster Compendium*. Seattle, Washington: IEEE Computer Society Press, pp. 114–115.

- 89 Stewart, C.A., Hart, D., Berry, D.K., Olsen, G.J., Wernert, E.A. and Fischer, W. (2001) Parallel implementation and performance of fastDNAm1 – A program for maximum likelihood phylogenetic inference. ACM/IEEE Conference on Supercomputing, Denver, CO. New York: ACM/IEEE Computer Society, pp. 20–20.
- 90 Dwyer, T. and Schreiber, F. (2004) Optimal leaf ordering for two and a half dimensional phylogenetic tree visualisation. N. Churcher and C. Churcher (eds.) Australian Symposium on Information Visualisation, Dunedin, New Zealand, 35, Sydney: Australian Computer Society, pp. 109–115.
- 91 Telea, A. and Auber, D. (2008) Code flows: Visualizing structural evolution of source code. *Computer Graphics Forum* 27(3): 831–838.
- 92 Tominski, C., Abello, J. and Schumann, H. (2004) Axes-based visualizations with radial layouts. In: H. Haddad, A. Omicini, R.L. Wainwright and L.M. Liebrock (eds.) 19th ACM Symposium on Applied Computing, Nicosia, Cyprus. New York: ACM Press, pp. 1242–1247.
- 93 Chi, E. H., Pitkow, J., Mackinlay, J., Pirolli, P., Gossweiler, R. and Card, S. K. (1998) Visualizing the evolution of web ecologies. In: C.M. Karat, A. Lund, J. Coutaz and J. Karat (eds.) *ACM CHI*. Los Angeles, CA. New York: ACM Press, pp. 400–407.
- 94 Spenke, M. and Beilken, C. (2003) Visualisation of trees as highly compressed tables with InfoZoom. In: C. Plaisant and J.-D. Fekete (eds.) *IEEE InfoVis*. Seattle, Washington: IEEE Computer Society Press, pp. 122–123.
- 95 Kutz, D. O. (2004) Examining the evolution and distribution of patent classifications. IEEE Conference on Information Visualisation. London, UK. Los Alamitos: IEEE Computer Society Press, pp. 983–988.
- 96 Morse, D. R., Ytow, N. and Roberts, D. M. (2003) Comparison of multiple taxonomic hierarchies using TaxoNote. In: C. Plaisant and J.-D. Fekete (eds.) *IEEE InfoVis Poster Compendium*. Seattle, Washington: IEEE Computer Society Press, pp. 126–127.
- 97 Daida, J.M., Hilss, A.M., Ward, D.J. and Long, S.L. (2005) Visualizing tree structures in genetic programming. *Genetic Programming and Evolvable Machines* 6(1): 79–110.
- 98 Slack, J., Hildebrand, K. and Munzner, T. (2006) PRISAD: A partitioned rendering infrastructure for scalable accordion drawing. *Information Visualization* 5(2): 137–151.
- 99 Chevenet, F., Brun, C., Bañuls, A.-L., Jacq, B. and Christen, R. (2006) TreeDyn: Towards dynamic graphics and annotations for analyses of trees. *BMC Bioinformatics* 7: 439.
- 100 Graham, M., Kennedy, J. and Downey, L. (2006) Visual comparison and exploration of natural history collections. *ACM AVI*. Venice, Italy: ACM Press, pp. 310–313.
- 101 Card, S.K., Suh, B., Pendleton, B.A., Heer, J. and Bodnar, J.W. (2006) TimeTree: Exploring time changing hierarchies. In: P.C. Wong and D. Keim (eds.) *IEEE VAST*. Baltimore, MO. Los Alamitos: IEEE Computer Society Press, pp. 3–10.
- 102 Herman, I., Melançon, G., de Ruiter, M.M. and Delest, M. (1999) Latour – A tree visualisation system. In: J. Kratochvil, (ed.) *Graph Drawing*, Stirin Castle, Prague, Czech Republic, LNCS 1731. Berlin, Heidelberg, Springer-Verlag, pp. 392–399.
- 103 Wettel, R. and Lanza, M. (2008) Visual exploration of large-scale system evolution. In: A. Zaidman, M. Di Penta and A. Hassan (eds.) 15th Working Conference on Reverse Engineering (WCRE). Antwerp, Belgium. Los Alamitos: IEEE Computer Society Press, pp. 219–228.
- 104 Becker, R.A. and Cleveland, W.S. (1987) Brushing scatterplots. *Technometrics* 29(2): 127–142.
- 105 Abello, J. and van Ham, F. (2004) Matrix Zoom: A visual interface to semi-external graphs. *IEEE InfoVis*. Austin, TX: IEEE Computer Society Press, pp. 183–190.
- 106 Wong, P.C., Foote, H., Mackey, P., Perrine, K. and Chin Jr, G. (2006) Generating graphs for visual analytics through interactive sketching. *IEEE Transactions on Visualization and Computer Graphics* 12(6): 1386–1398.
- 107 Florentz, B. and Muecke, T. (2006) Unification and evaluation of graph drawing algorithms for different application domains. In: E. Banissi (ed.) International Conference on Information Visualization. London, UK. Los Alamitos: IEEE Computer Society Press, pp. 475–482.
- 108 Mank, F.W.M. (2005) CristalView – The visualization of a Cristal. Master's Thesis, Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, Netherlands, 2005; pp. 115+v.
- 109 Burch, M. and Diehl, S. (2006) Trees in a treemap: Visualizing multiple hierarchies. *Visualization and Data Analysis*. San Jose, CA: SPIE Press, 6060, pp. 224–235.
- 110 D'Ambros, M. and Lanza, M. (2006) Software bugs and evolution: A visual approach to uncover their relationship. *10th European Conference on Software Maintenance and Reengineering – CSMR 2006. Bari, Italy*. Los Alamitos: IEEE Computer Society Press, pp. 227–236.
- 111 Melançon, G. and Herman, I. (2000) DAG drawing from an information visualization perspective. In: R. van Liere and W. de Leeuw (eds.) *Eurographics/IEEE TCVG Symposium on Visualization (VisSym)*. Amsterdam, The Netherlands. Wien: Springer-Verlag, pp. 3–12.
- 112 Reiss, S.P. (1995) An Engine for the 3D visualization of program information. *Journal of Visual Languages and Computing* 6(3): 299–323.
- 113 Amenta, N. and Klingner, J. (2002) Case study: Visualizing sets of evolutionary trees. *IEEE InfoVis*. Boston, MA: IEEE Computer Society Press, pp. 71–74.
- 114 Hillis, D.M. and Heath, T.A. (2005) St John K Analysis and Visualization of Tree Space. *Systematic Biology* 54(3): 471–482.
- 115 Meyer, J. (2006) A framework for large-scale interactive visualization of phylogenetic trees. In: J.J. Villanueva (ed.) IASTED International Conference on Visualization, Imaging and Image Analysis. Palma de Mallorca, Spain. Calgary: ACTA Press, pp. 122–129.
- 116 Maddison, W.P. and Maddison, D.R. (2009) Mesquite: A modular system for evolutionary analysis (Version 1.06) [Computer Software], <http://mesquiteproject.org>, accessed 2 April.
- 117 Graham, M., Kennedy, J.B. and Hand, C.A. (2000) Comparison of set-based and graph-based visualisations of overlapping classification hierarchies. *ACM AVI*. Palermo, Italy: ACM Press, pp. 41–50.
- 118 Barlow, T. and Neville, P. (2001) A comparison of 2-D visualisations of hierarchies. *IEEE InfoVis*. San Diego, CA: IEEE Computer Society Press, pp. 131–138.
- 119 Andrews, K. and Kasanicka, J. (2007) A comparative study of four hierarchy browsers using the hierarchical visualisation testing environment (HVTE). In: E. Banissi et al. (eds.) IEEE IV Conference. Zurich, Switzerland. Los Alamitos: IEEE Computer Society Press, pp. 81–86.
- 120 Parunak, H. V. D. (1991) Don't link me in: Set based hypermedia for taxonomic reasoning. In: J. Walker (ed.) *ACM Hypertext*. San Antonio, TX. New York: ACM Press, pp. 233–242.