Exploring Multiple Trees through DAG Representations

Martin Graham and Jessie Kennedy

Abstract—We present a Directed Acyclic Graph visualisation designed to allow interaction with a set of multiple classification trees, specifically to find overlaps and differences between groups of trees and individual trees. The work is motivated by the need to find a representation for multiple trees that has the space-saving property of a general graph representation and the intuitive parent-child direction cues present in individual representation of trees. Using example taxonomic data sets, we describe augmentations to the common barycenter DAG layout method that reveal shared sets of child nodes between common parents in a clearer manner. Other interactions such as displaying the multiple ancestor paths of a node when it occurs in several trees, and revealing intersecting sibling sets within the context of a single DAG representation are also discussed.

Index Terms—Multiple trees, Directed Acyclic Graph.

1 Introduction

Our previous work in Graham and Kennedy [1] compared the utility of two differing styles of representing multiple, overlapping classifications – namely an agglomerated graph representation of all the classifications and a small-multiple representation of individual classifications with coloured overlaps. Previous discussions and testing with users and constructors of taxonomic classifications revealed that they preferred the small-multiple style of representation, stating that the graph representation showed too much information at once and that the hierarchical information associated with individual classifications was lost when drawn using the force-based layout algorithms that we utilised for graph drawing.

Therefore, to alleviate the difficulties caused by the latter point, we have developed a visualisation based on a DAG (Directed Acyclic Graph) layout, formed by merging all the taxonomies of interest into one agglomerated structure as with the general graph visualisation, but laid out such that nodes at the same rank or layer are drawn so the global parent-child direction is preserved.

In the following sections, we describe related work in tree and multiple tree visualisations, followed by a discussion of how our visualisation differs from existing systems. We then describe the methods used to construct the DAG and the interactions that such a representation makes possible, and user feedback from expert users, namely a group of five taxonomists.

2 Related work – single and multiple tree visualisations

Tree visualisations have a long history before the coining of the term 'Information Visualization' (IV), the classic reference being Reingold and Tilford's [2] work, itself only one of many pre-1990 algorithms for laying out various types of tree structure as documented in Beebe's bibliography [3]. However, these approaches tended to focus exclusively on layout algorithms; what Information Visualisation introduced was the notion of being able to interact with the generated tree visualisations. Here we describe some of the basic approaches to visualising single and then multiple tree visualisations.

2.1 Single Trees

Traditional single tree layouts divide into three basic categories, based on the method used to indicate a parent-child relationship. The first and most well-known is the node-link layout as shown in Figure 1a), developed by Reingold and Tilford [2] and also used by Plaisant et al. [4] with parent-child relations represented by lines (links) drawn between nodes that represent the objects in the tree. Secondly, nested layouts, such as Johnson & Shneiderman's TreeMaps [5] and Wang et al. [6], convey parent-child relationships by placing child nodes within the boundaries of their parent node, as demonstrated in Figure 2b). Finally, there is the adjacency layout style shown in Figure 1c), where child nodes are drawn next to 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 the above figure and Sifer's work [7] or centre-out as in Stasko and Zhang's radial space-filling tree [8]. All of these approaches have been extended from their 2D projections to 3D variants, with various degrees of success: e.g. Robertson et al.'s Cone Tree node-link visualisation [9], Bladh et al.'s nested 3D treemaps [10], and van Ham and van Wijk's Beamtree [11] for adjacency methods. These three basic layout styles are the foundation for all tree visualisations that display internal tree structure. We do not consider adjacency matrix representations of trees, as these are more commonly thought of as mathematical representations than a visualisation style.

All three layout styles have associated advantages and disadvantages and the choice of representation is dependent on the tasks that are to be performed with the structures. 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. The adjacency 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.

The separate styles can be combined within a visualisation of a single tree as demonstrated by Zhao et al. [12], where portions of a tree are drawn as either nested or node-link representations dependent on screen space and user interaction. Another hybrid representation is that used in Microsoft Windows Explorer, which contains stylised links between nodes but mostly relies on indentation and adjacency to communicate parent-child and sibling relationships, achieving a very compact yet legible representation of hierarchical structures. In empirical evaluation by Kobsa [13] this layout was shown to be the objectively preferred choice when compared to other tree visualisations, though it was recognised that some of this performance advantage may be familiarity due to the ubiquitous presence of Microsoft Windows.

2.2 Multiple Trees

The data sets we have studied over the past few years are formed from multiple trees, specifically multiple overlapping taxonomies,

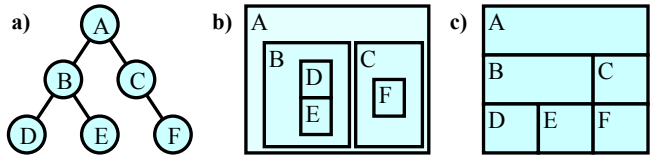


Figure 1. Three basic kinds of tree drawing - a) node-link, b) nested, & c) adjacency

the overlapping condition marking these structures as distinct from simply a collection of unrelated hierarchies. The obvious difficulties are resolving screen space allocation and conveying the overlap between individual trees, for which three approaches exist: division, animation and agglomeration.

The division approach, termed *small multiples* by Tufte [14], sub-divides available screen space into areas in which the individual trees are drawn. Unsurprisingly, this approach tends to favour the more space-efficient individual tree layout representations. Munzner's [15] approach is strictly speaking a node-link layout, but internal nodes are not labelled and the allocation for individual nodes can become so compressed that the drawn links may use all the space available for drawing, hence moving towards the adjacency style of representation. Our previous work [1, 16] used multiple adjacency-layout representations, as did Chi et al. [17], whilst Wittenberg et al. [18] used multiple nested layouts to represent their trees. Morse et al. [19] displayed multiple windows explorer style windows to compare and contrast several taxonomic trees. Interaction is generally achieved through a linking metaphor – objects selected in one representation are marked where they occur in the other tree representations. The division approach works well for a handful of trees but does not scale well, due to each tree receiving a correspondingly smaller area of screen space as the size of the set grows - the largest number of trees so far displayed with this method is 14, demonstrated with a set of museum collections in Graham et al. [16].

The second approach, animation, is used to display changes of structures or viewpoint between representations of different trees, in effect distinguishing the trees temporally rather than spatially. In practice, animation is best used for showing gradual transitions, thus it is suited to showing successive trees that represent evolving change rather than radical reorganisation where a user can easily lose track of the situation, so animation in multiple trees is mainly reserved for showing changes in values associated with tree nodes as in work by Ghoniem and Fekete [20] or small-scale addition and deletion of nodes as demonstrated by Wittenburg and Sigman [21]. The number and complexity of trees which 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,

The final option, agglomeration, is the visual aggregating of multiple tree structures so that correlating nodes in tree structures overlay each other, giving the impression of a directed graph and often a truer representation of the underlying data model. Edges from the different trees can be distinguished by visual properties such as colour, pattern or saturation. Agglomeration of multiple trees means in practice that a node can have multiple parents to display in the same representation, possibly one per tree, which the nested and adjacency approaches find difficult to do, although Hong et al.'s Zoomology system [22] features a view of two merged trees displayed adjacency style, with differences between the trees marked with a specific colour. Thus, agglomerative representations are generally displayed using node-link representations as in Graham et al [1]. Furnas and Zack's Multitree visualisation [23] was one of the first approaches that used this method, with multiple trees defined over the same structure, and family tree style layouts developed from two trees laid out in opposite directions. As screen space is effectively re-used there is no technical upper limit to how many trees can be displayed using this method, though perceptual difficulties in interpreting the merged structure typical of general graph drawing techniques such as edge crossings and nodes and edge occlusion occur. Also, care must be taken when displaying using general graph drawing mechanisms as often it results in no global orientation for child-parent links even if one exists in the overall structure.

A popular compromise between the three former approaches is to use a 3D representation of multiple trees. These 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 links between trees as in Dwyer and Schreiber [24] or by using colouring as in Chi et al. [17]. The 3D approach means the group of trees can be rotated so that they resemble a division style approach to displaying multiple trees (one tree per section of screen space), or turned through ninety degrees so the structures give the impression of overlaying one another. This last feature has the drawback though of not guaranteeing equivalent nodes in different trees will overlay each other and can lead to a display with a high degree of occlusion.

Finally, when the number of trees grows extremely large, the finite screen space cannot show all the trees in detail, so some approaches visualise the trees as atomic items, from which examples can be viewed in detail. Hillis et al. [25] 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.

3 Discussion

Historically, multiple tree visualisations in the IV literature have favoured the small multiples approach of drawing trees separately and using brushing and linking techniques to coordinate selections between the tree representations. Of the six published entries for the InfoVis 2003 contest [26] to visualise multiple trees, five used this approach as the foundation of their visualisation with the exception of Wernert et al. [27] who used a 3D variant. Hong et al.'s Zoomology browser [22] also used an agglomeration technique for an overview comparison of two trees, while animation between trees was not used in any of the approaches, the technique being reserved for illuminating focus+context transitions internal to trees.

Obviously 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, and multiple trees can form a number of different structures dependent on their overlap and relative orientation, examples of which are elaborated in McGuffin and Schraefel's work [28]. An unrelated forest of trees will obviously be easily, and perhaps only, represented as separate visual entities, whilst 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 are more problematic as the differing tree structures produce significant extra edge-crossings in the agglomeration style views. Techniques specifically developed for drawing DAGs can however impose a global orientation for parent-child links on the structure if the trees all have the same parent-child orientation – that is a node closer to the root than another node in one tree will always maintain that characteristic in another tree - though the restrictions on node placement involves a trade-off on edge crossings as seen in Melançon et al. [29]. Linnean taxonomic classifications have this property, through being organised using an immutable set of ordered ranks, though other hierarchical structures such as phylogenies are not guaranteed to have this property, and Robertson et al.'s [30] polyarchy structures in effect may construct trees freely from a pool of nodes regardless of those nodes' positioning in other trees. Multiple taxonomic trees in particular though form a type of DAG we have previously termed a Directed Acyclic MultiGraph (DAMG), with the multigraph qualifier resulting from the fact that the same relation between two nodes can occur in multiple trees, and also has

the feature that as it is composed of overlaid tree structures, it increases in size exponentially like a tree as we traverse down the layers.

Previously, in Graham [31], we had tried to fix the lack of a global hierarchical layout in our earlier agglomerated graph visualisation by experimenting with a rank-restricted layout that kept nodes of the same rank within distinct semi-circular concentric arcs. However this did not work well with the force-directed metaphor we were using, as we had essentially restricted the nodes to one-dimension of freedom within their rank rather than the two-dimensional freedom they enjoyed in the general graph layout. Thus many nodes were prone to becoming trapped in local minima, so rather than clarifying the layout the edges crossings became more prevalent, and also became more densely packed as the nodes they emanated from were restricted to fixed areas.

So, when revisiting the idea of displaying the multiple taxonomic trees as a unified DAG we decided to use one of the common heuristic DAG layout methods developed by Eades and Wormald [32], using Barth et al.'s [33] quick method for calculating the number of edge crossings. These layout heuristics, based on Sugiyama et al.'s early work [34], start with an initial assignment of nodes to distinct layers (the assignment of nodes to layers is inherent in our data as each node/taxon is set at a particular rank), and an initial ordering of nodes is made within each layer. Then, starting with the second topmost layer, each node in this layer finds its immediately connected neighbours in the above layer, and calculates the median or mean coordinate of those neighbouring nodes, the values of which are then used to order the nodes within the current layer. This process proceeds down the layers and at the bottom layer the process is reversed and performed back up the set of layers. Alternating downward and upward passes are iterated until the layout reaches a termination condition such as the number of edge crossings equalling zero or stabilising over the last two or more passes, or that the number of iterations carried out has reached a given maximum.

The procedure is simple and not as effective at reducing edge-crossings as other more sophisticated options, for instance neighbouring nodes can be trapped in unsuitable configurations if their mean or median values are the same, as seen in Figure 3. Further, Marti and Laguna [35] performed empirical experiments that have shown simple ordering heuristics such as ordering on the median and barycentre (mean) averages compare poorly on relatively sparse graphs to other layout approaches such as GRASP (Greedy Randomized Adaptive Search Procedures) [36] and Tabu [37], yet the running time of these simple heuristic methods is vastly superior which is a prime concern in an interactive environment.

The two popular methods of ordering heuristic, the barycentre or median differ in a number of subtle aspects. The median is regarded as being less sensitive to extreme distributions of nodes in the layers, whereas the mean can be skewed by one or two outliers. The barycentre method is considered preferable for graphs with a few nodes of large degree whilst the median heuristic deals more successfully with nodes of smaller degree. In cases where average values for two nodes in a layer are equal, the node with the smallest or odd-degree can be considered as having precedence. In the median case if the number of node values to be considered is even, options include taking one of the median values, halving the two median values or biasing the value towards the side where nodes are more densely distributed as in Gansner et al. [38].

4 DESIGN

To visually simplify the layouts we made some augmentations to the standard DAG heuristic-based layout. Firstly, the node positioning algorithm traverses the layers from top to bottom. If after this first pass there are no crossings detected the layout process halts, and either the graph is an extremely simple one or, more likely, it is a tree. The initial ordering of the nodes within each layer is preserved

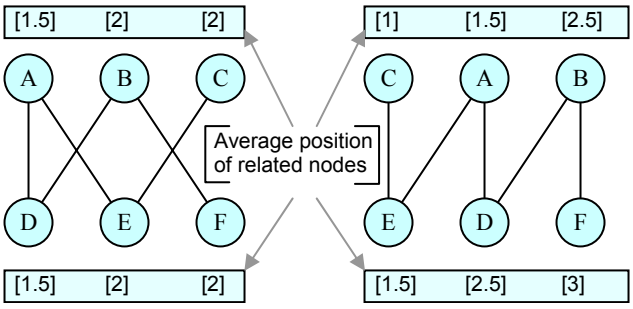


Figure 2. In the left-hand diagram, the node pairs B & C and E & F have the same median and mean values according to the positioning of their immediately connected neighbours. The layout of the nodes will therefore not automatically improve to the layout seen in the right hand side version.

within their sibling sets, such as alphabetical or subtree size ordering. Otherwise the layout process iterates from bottom to top and then back down again until one of the terminating conditions is met. In these further sweeps, the concepts of *major* and *minor* averages are introduced. Major averages are the average positions of the nodes when calculated using the nodes in the next adjacent layer, as per the traditional algorithm. Minor averages are calculated using the node positions in the previous layer, and are used to tiebreak if the major average values are the same for any nodes. Melançon et al. [29] used the concept of calculating averages for nodes in a layer from all their neighbouring node positions after performing the original upwards or downwards-only calculation to reduce the number of iterations needed to stabilise a DAG layout, but did not differentiate between the two types. This method can stop some simple superfluous edge crossings at the cost of some extra calculations. It is not used in the first sweep as we prefer to use the pre-orderings of the nodes within the layers as a tiebreaker in case the structure being displayed is a tree, and in that case we view preserving node orderings such as alphabetically by name or sub-tree sizes as more important for orientation and navigation purposes.

Secondly, standard DAG layout involves inserting dummy nodes along edges whose source and destination nodes do not lie on neighbouring layers, the result being to break up such an edge into a series of segments between adjacent layers. In the context of taxonomic trees, such edges can occur in individual trees when a sub-rank is used inconsistently, so for example a parent-child relationship from a *tribe* to a *genus* will extend across the *sub-tribe* rank if that is used elsewhere in the taxonomy. In the context of a merged set of taxonomic trees, such edges occur when different ranks are used by taxonomists when constructing the taxonomies. The most apparent case is when one taxonomy in the set uses a rank uniquely, such as *legion* or *grex*, and thus all the other trees' edges that span this rank must be routed through this layer by use of dummy nodes. This leads to a visual surfeit of line segments in the final DAG rendering, and for any one parent node with multiple children this results in many lines being drawn close to each other at minimal angular resolution. To counteract this effect we analyse the children of each parent node. Those that are discovered to share the same set of parent taxa across the current set of taxonomies have their individual paths to each parent replaced by one path from each parent to the final dummy node, from where the final segments of individual paths to the child nodes are drawn as demonstrated in Figure 3. This reduces the number of dummy nodes generated when computing the DAG and also reduces the visual complexity of the final rendering, with layer-spanning edges being realised by one intermediate layer-crossing path that encompasses the dummy nodes followed by a number of single-segment paths to their destination nodes instead of the same number of many poly-line paths.

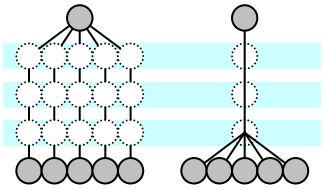


Figure 3. Merging dummy nodes for child nodes with the same parent nodes reduces the number of paths drawn in the final representation.

This could be viewed as counter-productive as it would tend to tie these nodes together when it could be imagined in most circumstances that these nodes would require the freedom to move away from each other. However, these node sets can still pull apart from each other if for instance child node placement is radically different, but the fact that these nodes share the same parents in the 'higher' layers would indicate that these nodes are closely related in the graph structure and a layout that encourages this to be accentuated is not as detrimental as at first glance. Indeed, we use this notion of parent 'sets' to pre-order the nodes within each layer – it is acknowledged that the heuristic placement methods for DAGs are very susceptible to the initial node orderings and having such related nodes gathered together in this manner makes sense.

The edge-grouping effect seen in the right-hand side of Figure 3 was pleasing enough that it encouraged us to reproduce the same effect for simple edges between adjacent layers. This was achieved by inserting entire layers of dummy nodes between adjacent layers. This obviously raised the number of dummy nodes in the calculation of the DAG layout, but was still less than those occurred by the traditional method as dummy nodes occurred between two layers at the rate of one per unique set of parents rather than one per every child node.

This bunching of child node edges by the child's shared parent sets also led to interesting results when viewing nodes whose range spans many trees. Under normal barycentre DAG layout the children of such nodes would be laid out with no consideration for the trees they each belonged to, only for the parent nodes they were classified under in those trees. Obviously if the parent node is the same across multiple trees the discrimination in laying out the child nodes is lost. However, using the parent set approach the children are collected together according to which tree or set of trees they occurred in so patterns such as which child nodes belong to the same tree can be observed.

5 LAYOUT

Figure 4 demonstrates a typical layout of multiple trees as a DAG using a set of eight different hierarchical classifications of the *Apiaceae* taxonomic family, the same data set we previously used in Graham et al. [1]. This data set is not particularly large in the context of some other taxonomic trees, but the various classifications contain a lot of differences, as opposed to larger taxonomies which tend to simply feature annual additions or deletions and little in the way of structural rearrangements.

In Figure 4, each layer in the DAG is assigned to a horizontal band in the display, with nodes represented as labelled boxes. Along the top of each node representation lie a set of small rectangles that are coloured according to which trees the node occurs in and whether those trees are currently active or hidden. For instance in the centre of Figure 4 *Ammieae* has been selected – the node representation has five blue, gradient-shaded rectangles along its top edge representing its occurrence and subsequent selection in five out of the eight

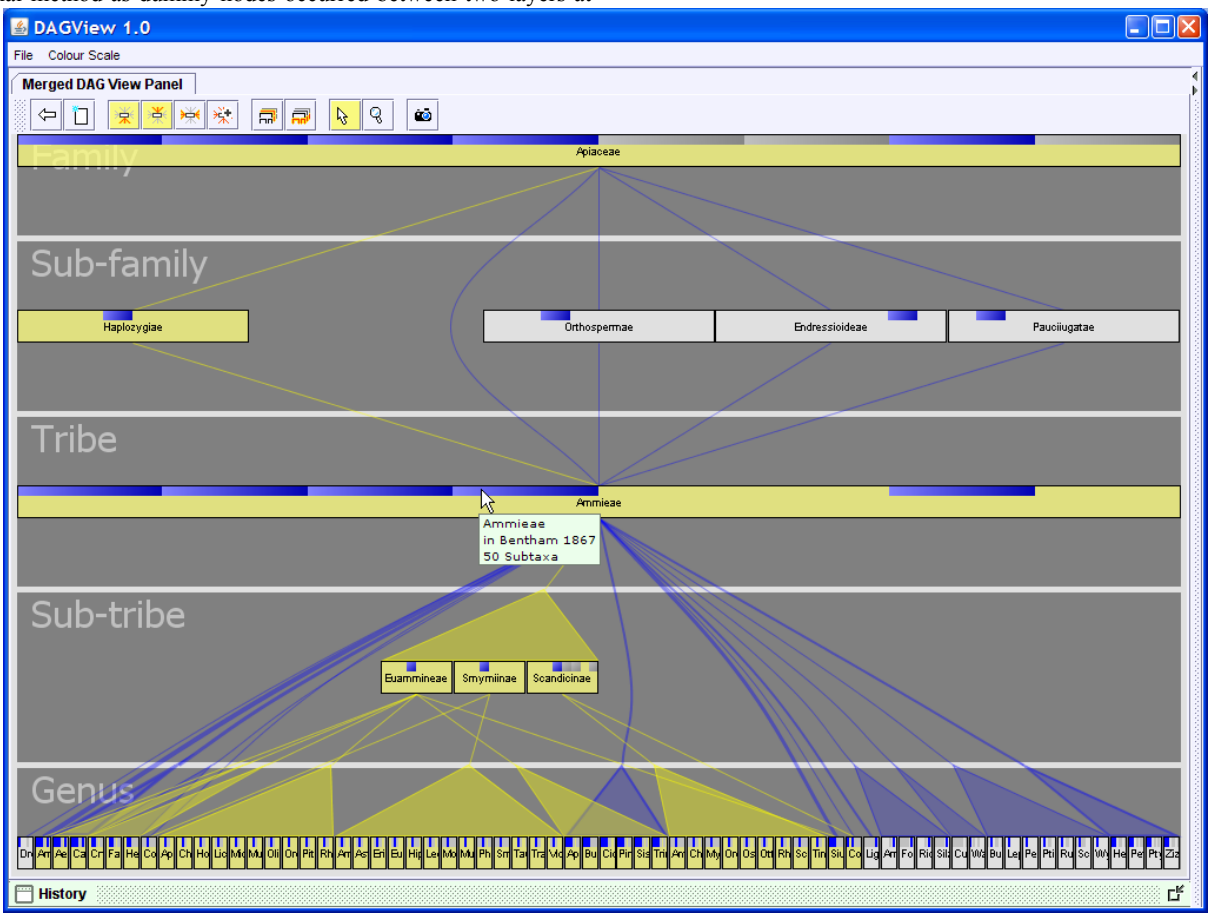


Figure 4. Screenshot of the multiple classification DAG visualisation. This features an example from a data set of eight merged taxonomic classifications, in this instance centered on *Ammieae*.

classifications. There are gaps where representations for three other classifications could appear; their absence indicating that this node does not occur in those classifications. Similarly, the *Apiaceae* node representation at the top of the figure, which is the family to which all the nodes in the data sets belong, has eight boxes along its top edge, revealing it is present in all eight classifications. Five are these are coloured blue, to indicate the trees that are currently involved in the relationships shown on screen - these match to the five taxonomies that *Ammieae* occurs in - and three are coloured grey to indicate *Apiaceae* occurs in these taxonomies but no relationships are currently represented. This is useful to give an indication of where further undisplayed relationships for a node are present.

The mouse pointer is pictured brushing the *Ammieae* node, specifically on the decal representing the Bentham classification. The descendants, ancestors of *Ammieae* in Bentham along with the associated edges are thus coloured yellow.

Edge paths are drawn as curves. We use a simple algorithm to smooth out the complex polyline paths that can result when an edge between two nodes pass through many intervening layers via a series of dummy nodes. We decided that an optimal solution was not to route the edges through the dummy nodes involved in the entirely dummy layers unless it was the dummy layer immediately above the child nodes belonging to the edge(s) in question, otherwise the nodes in these layers are discarded. We then aim from the source node towards the destination node, calculating if the line can pass through each intersecting layer in turn by moving the dummy node left or right without encountering any 'real' nodes. If this can't be done for a particular layer we move the dummy node in that layer as far as we can and then aim afresh for the destination node from that point. The source, destination and dummy nodes along each edge are then joined smoothly with a series of Catmull-Rom curves [39]. This occasionally leads to some extraneous edge crossings or edge back-tracking, but mostly alleviates the complex paths introduced by the use of the extra dummy layers.

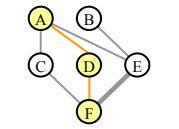
Multiedges, which occur when a direct relationship between two nodes occurs in multiple trees, are drawn with a line thickness proportional to the number of individual relations they represent, a visual cue common to other aggregated tree and graph representations such as Wattenberg's multivariate graphs [40]. In Figure 4, several thicker blue edges can be seen curving out from beneath the centre of *Ammieae* towards nodes and groups of nodes in the bottom layer, indicating relationships that are sustained over several classifications.

Finally, rather than drawing individual lines to nodes that share the same set of parents in the display, a translucent triangle is drawn that uses a shared dummy node as it's apex and the leftmost and rightmost occurrences of child nodes in a set as its other corners. The lowest layer of Figure 4 shows several such groups, indicating node groups that have common parents across the selected trees. The edges of these triangles are allocated a thickness using the same manner as for multiedges; the figure shows that a triangle enclosing the furthest right-hand nodes has a thicker edge than neighbouring groups, indicating that group of nodes are direct descendants of *Ammieae* in more than one taxonomy. In the middle of the bottom layer, there's an overlap between a blue and a yellow triangle where *Euammineae* in Bentham's classification has claimed ownership of a group of nodes that belonged directly to *Ammieae* in other classifications. In Figure 4 in particular most of the different groups result from Bentham's classification having sub-tribes as intermediate nodes and from nodes occurring inconsistently in the five instances of *Ammieae*. On an intermediate layer in the current display, triangles can sometimes include nodes that aren't members of that particular set, but brushing over the appropriate parent taxa will confirm whether it is a continuous range.

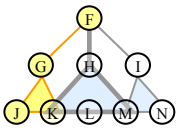
The visualisation is implemented in Java 1.6 using the Swing libraries.

6 INTERACTION

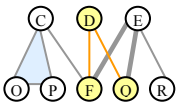
We decided to implement three basic operations that taxonomists had previously said would be useful to visualise in the context of multiple taxonomies, namely finding the parent taxa of a named taxon, finding the children of a named taxon and finding the siblings of a taxon across multiple trees. Figure 5 gives a diagrammatic example of each of these tasks, centred around the selection of one node.



Example of 4 ancestry paths from node F – FCA, FDA, FEA and FEB. The FE relationship occurs in two places so is drawn as a thicker edge, and the FDA path is highlighted.



Example of the children of node F over 4 trees – FG(JK), FH(KLM), FH(KLM) and FI(MN). The H(KLM) group is constant for two trees and is thus marked with a thicker outline. The children FG(JK) are highlighted.



Siblings of node F over 4 classifications – C(OPF), D(FQ), E(FQ) and E(FQR). Thicker lines show where nodes belong to the same parent for two or more classificiations. Node Q shares 3 parents with Node F.

Figure 5. Diagrams showing examples of how three basic tasks on a given node are performed.

The simplest operation possible is to trace the multiple ancestries of a node in the many trees. Here the number of ancestors is a multiple of the number of trees and the number of ranks above the node in question in those trees, as each child node has strictly one parent node per tree - as opposed to family trees where ancestors is an exponential function of the number of past generations that are considered. In the example of *Daucus* in Figure 6 it can be deduced that the thicker paths from *Daucus* going to *Caucalideae* and *Daucineae* represent the fact that these nodes are the parent of *Daucus* in more than one of the taxonomies under investigation. In this particular example, by tracing these ancestries further up the DAG it can be seen that any agreement between the taxonomies beyond the fact they both reside in the *Apiaceae* family ends there, with the ancestries via *Caucalideae* and *Daucineae* subsequently routing via differently named sub-families or straight to the family

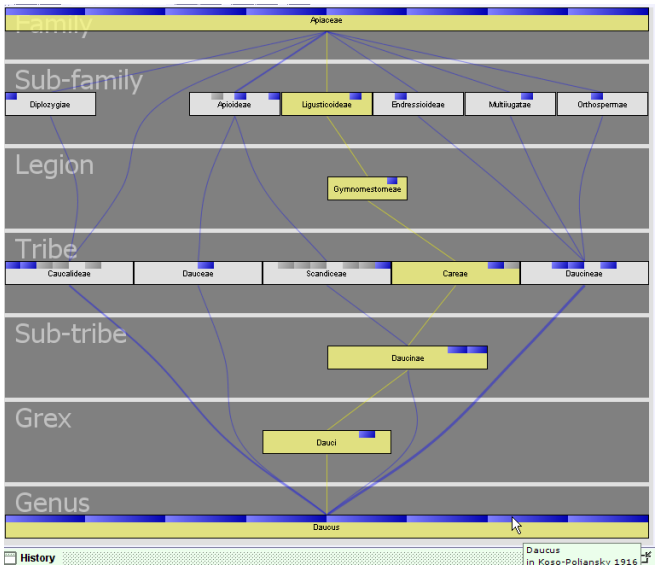


Figure 6. Screenshot of *Daucus'* ancestries with one tree path shown in yellow. Thicker lines represent multiedges where the same relationship occurs in multiple classifications

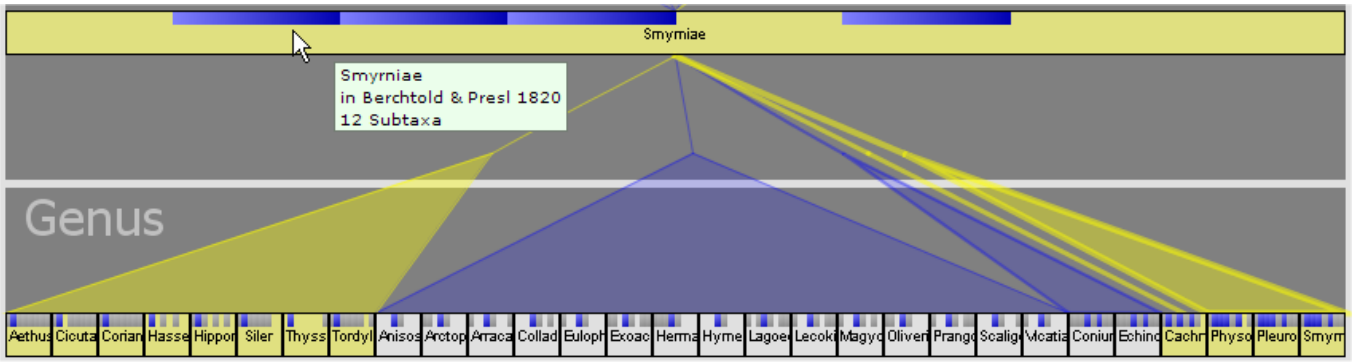


Figure 7. The child taxa of *Smyrniae* across four classifications are divided into 5 groups dependent on which combination of the four taxonomies their relationship to *Smyrniae* occurs in.

Apiaceae that encompasses these example data sets. The example is currently brushing the Koso-Poljansky classification and thus the path from Daucus up to Apiaceae for that classification is highlighted in yellow. For multiple ancestries, the overall effect of the layout is that of a small-scale Sankey diagram, larger versions of which are detailed in Riehmann et al. [41], with the sum of edge widths heading into any higher node equal to the number that leave it to travel further up the DAG towards the root.

The reciprocal operation to discovering the ancestry of a node, which is finding all the children of a given node across multiple trees, reveals the different structures that underlie that node where it occurs across multiple taxonomies. This is shown for *Smyrniae* in Figure 7, where its child nodes have been broken into five groups, with membership of a group dependent on which combination of the four 'versions' of *Smyrnieae* each child node belongs to. In the figure, the Berchtold & Presl taxonomy has been brushed highlighting the children of *Smyrniae* in that classification with a yellow hue. The blue-coloured membership decals along the top edge of the child nodes give clues as to why they have been grouped together. The left-hand highlighted group, from "Aethus…" to "Tordyl…" have one blue decal in the same position, indicating they are members of *Smyrniae* only for this taxonomy. The furthest right group of three highlighted child nodes, from "Physo…" to "Smyrn…" have the same blue colouring pattern as each other, indicating they are members of Smyrniae in this classification and also share the same membership pattern in several other classifications of *Smyrniae*. Finally "Cachr…", immediately to the left of the three previous nodes, has a different colouring pattern, indicating that while it is present in Berchtold & Presl's classification of *Smyrniae* and it is present in other's definitions of the same node, no other child node shares exactly that pattern of inclusion. Brushing the different classifications in the Smyrniae node

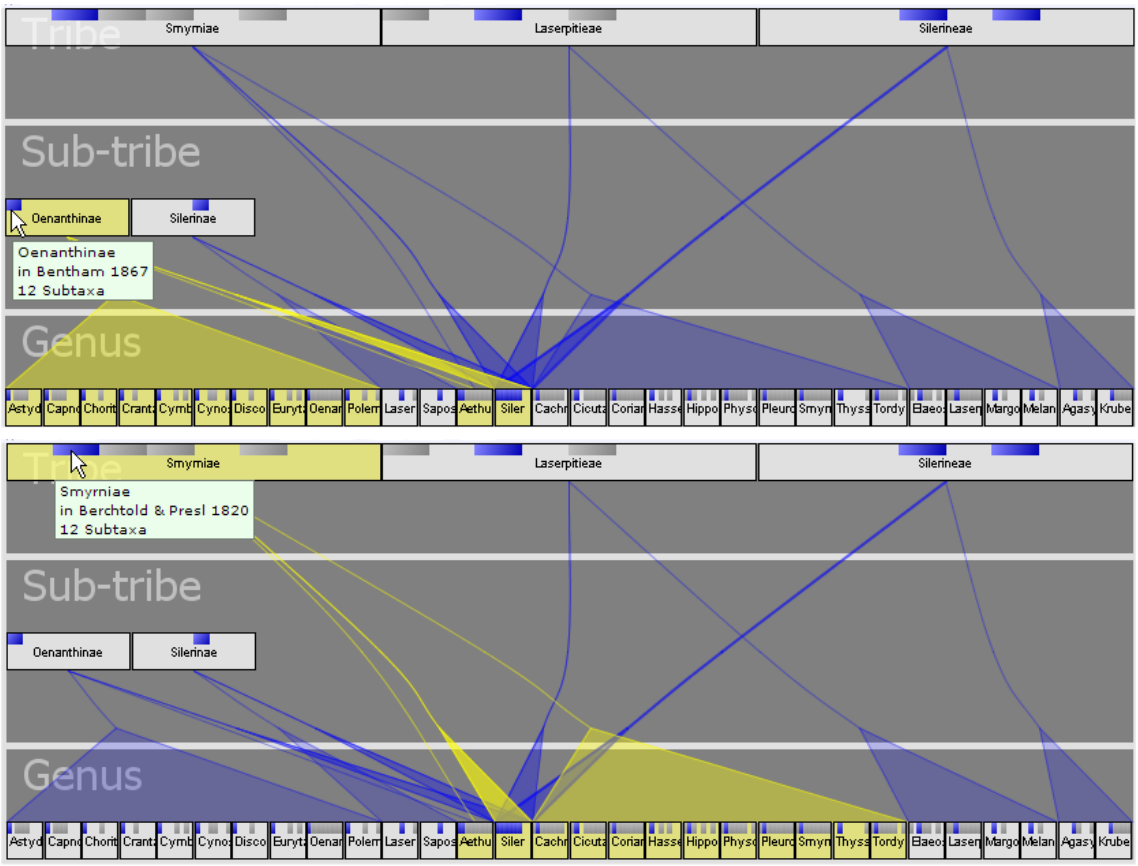


Figure 8. *Siler's* siblings across six classifications are lined up and grouped along the genus rank according to the parents they share with Siler. Brushing the parent nodes reveals Siler to have only one sibling in common across any of the taxonomies it occurs in, *Aethusa*, which in the layout lies immediately to Siler's left.

will reveal exactly what those patterns of inclusion are.

Again, multiedges are represented using lines of proportional thickness to the number of edges they represent, so the regularity with which child nodes or sets of child nodes appear under the same parent nodes can be viewed. In contrast, a node that has radically different contents in different trees will have a plethora of single-thickness links and more nodes underneath it. For large graphs the number of children can be overwhelming

Both the children and parent graphing operations can be carried out on a single tree basis by pressing the mouse when over one of the tree decals drawn in the top half of a node. The mouse tooltip will signal when it is over one of these areas and a subsequent mouse press will draw the ancestors and/or children of the node for that tree only. This allows queries to be made internally of a single tree rather than the whole tree set.

The final 'one-click' operation is the ability to find all the siblings of a node across a set of taxonomies, that is: all the other nodes it has shared at least one immediate parent node with, and thus been a sibling of those nodes at some time. Upon choosing this option and selecting a node, the sibling set is drawn across the bottom of the screen, and above it are placed the node's parents with the appropriate links drawn between the parent and sibling layers.

As an example, Figure 8 shows two comparative screenshots of the result of selecting to view *Siler*'s siblings. The siblings, and *Siler* itself, are laid out across the bottom of the screen, with its different parents in the set of active taxonomies positioned in the layers above. In the top screenshot, one particular parent node, *Oenanthinae,* has been brushed and thus its child nodes and relationships have been coloured yellow, revealing the nodes *Siler* was grouped with in Bentham's 1867 classification – comprising of one group at the far left of the row and *Aethusa* positioned immediately to the left of *Siler*. Similarly, the lower screenshot shows *Smyrniae* being brushed, *Siler's* parent in Berchtold & Presl's taxonomy. Here the children are revealed to be a group immediately to the right of *Siler* and also *Aethusa* again. Thus, the only sibling *Siler* has in common between these two taxonomies is *Aethusa*. Again this operation can be restricted to a single tree, which is simply equivalent to selecting to reveal its parent node's children in that tree.

The interface allows any combination of these three operations to be performed on a node. Finding the ancestries and the children of a node is useful for middle ranking nodes - leaf nodes obviously have no children to display. Sibling information is often best displayed by itself as it often forms a separate question to that of discovering a node's ancestors or descendants. Furthermore, an 'accumulate' mode can be selected which incorporates the results of new selections into the structure currently visible on-screen. This is necessary in large graphs where drawing the entire set of trees would both be unwieldy in terms of time to calculate the layout, and in interaction, as individual node representations would shrink to sub-pixel sizes. Rather, we calculate a maximum number of nodes to be drawn at any one level, beyond which a cut-off applies to their visibility, and the accumulate option enables drill-down operations into tree sets that form deep, large DAGs.

Both the individual trees and ranks that form the DAG can be hidden and then re-displayed at command. Hiding ranks is often useful when only one or two trees in a set use a particular rank, as this then allows comparison of taxonomies based only on common ranks. It also helps to clarify the DAG representation by removing sparsely populated ranks from the display. Similarly, trees can be added or removed, which is useful for when either comparisons of structures of tree subsets is needed, or when viewing the structure of an individual tree is required. In the latter situation, a graph can be drawn composed of just one tree, and then by making more trees visible in the overall graph, nodes that are subsequently selected in this single tree can have their relations to other trees visualised.

Upon a new selection a standard IV animation technique is used to fade out old nodes, move nodes present in both graph views between their old and new positions, and then fade in newly introduced nodes. Animation is extremely useful when we change the section of the graph we are viewing, as the heuristic algorithms for laying out DAGs sometimes produce radical changes in node positions even if only relatively small changes are made. Pre-ordering nodes in their layers by the same metric each time keeps the layout as stable as can be expected with these heuristics, but the animation makes any changes clear to the user, communicating not only changes in viewpoint but in the local structure they are viewing.

Brushing a node by moving the mouse cursor over it highlights the node plus its associated edges and paths both up and down the DAG. By moving over the top of the node, where the tree visibility information is displayed, edges can be highlighted on a per-tree basis by moving over the corresponding tree decal.

Initial selections or selections of nodes not present in the current display can be made through a list of nodes, itself divided into columns by tree, on the right-hand side of the display. This can be hidden using a split-pane control to maximise the display area for the DAG visualisation.

7 User feedback

Informal usability testing has been carried out with five taxonomists, following Nielsen's [42] discount usability methodology of getting real users to attempt representative tasks with the application. This revealed various issues with the visualisation and the tasks they wanted to accomplish, examples including the ability to sort the classifications chronologically as well as alphabetically to quickly find first use of a name within a set of classifications, and a simple 'back button' method to return to a previous layout was strongly urged. Various minor problems with the general interface were also picked up, such as the tree inclusion decals in the node being hard to separate when they merged together in a long row, this for instance was rectified with a gradient shading as seen in the previous figures.

8 Conclusion

We have presented a DAG-based visualisation of overlapping multiple classifications. This visualisation differs from other IV systems designed to explore multiple classifications in that it merges the trees into a unified structure whilst preserving a global parent-child orientation of the nodes, which is essential for navigation and interpretation of taxonomic trees with their strict assignment of nodes to ordinal ranks.

This method of overlaying classifications allows nodes to be seen in the context of multiple trees, without the shrinking space problems of the small multiple design.

Acknowledgements

We would like to thank the taxonomists at the Royal Botanical Garden Edinburgh (RBGE) for their time and invaluable feedback. This work was supported by a Engineering and Physical Sciences Research Council (EPSRC) grant.

References

[1] M. Graham, J. B. Kennedy, and C. Hand, "A Comparison of Set-Based and Graph-Based Visualisations of Overlapping Classification Hierarchies," In, V. D. Gesù, S. Levialdi, and L. Tarantino, editors, *Proc. ACM AVI* (Palermo, Italy, May 23-26, 2000), pages 41-50. ACM Press.

[2] E. M. Reingold and J. S. Tilford, "Tidier drawing of trees," *IEEE Transactions on Software Engineering*, vol. 7, no. 2, pp. 223-228, March 1981.

[3] N. H. F. Beebe, "A Bibliography of Tree Drawing Algorithms." Salt Lake City: Department of Mathematics, University of Utah, 2006, pp. 22.

[4] C. Plaisant, J. Grosjean, and B. B. Bederson, "SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical

Evaluation," In *Proc. IEEE InfoVis* (Boston, Massachusetts, USA, October 28-29, 2002), pages 57-64. IEEE Computer Society.

[5] B. Johnson and B. Shneiderman, "Treemaps: A Space-Filling approach to the visualization of hierarchical information structures," In *Proc. IEEE Visualization* (San Diego, California, USA, Oct 22-25, 1991), pages 284-291. IEEE Computer Society Press.

[6] W. Wang, H. Wang, G. Dai, and H. Wang, "Visualization of Large Hierarchical Data by Circle Packing," In *Proc. ACM CHI* (Montréal, Québec, Canada, April 22-28, 2006), pages 517-520. ACM Press.

[7] M. Sifer, "Filter co-ordinations for exploring multi-dimensional data," *Journal of Visual Languages and Computing*, vol. 17, no. 2, pp. 107-125, 2006.

[8] J. Stasko and E. Zhang, "Focus+Context Display and Navigation Techniques for Enhancing Radial, Space-Filling Hierarchy Visualizations," In *Proc. IEEE InfoVis* (Salt Lake City, Utah, USA, October 9-10, 2000), pages 57-65. IEEE Computer Society Press.

[9] G. G. Robertson, J. D. Mackinlay, and S. K. Card, "Cone Trees: Animated 3D Visualizations of Hierarchical Information," In *Proc. ACM CHI: Human Factors in Computing Systems* (New Orleans, Louisiana, USA, April 27 - May 2, 1991), pages 189-194. ACM Press.

[10] T. Bladh, D. A. Carr, and J. Scholl, "Extending Tree-Maps to Three Dimensions: A Comparative Study," In, M. Masoodian, S. Jones, and B. Rogers, editors, *Proc. 6th Asia-Pacific Conference on Computer-Human Interaction* (Rotorua, New Zealand, June 29 - July 2, 2004), pages 50-59. Springer-Verlag.

[11] F. van Ham and J. J. van Wijk, "Beamtrees: Compact Visualization of Large Hierarchies," In *Proc. IEEE InfoVis* (Boston, Massachussets, USA, October 28-29, 2002), pages 93-100. IEEE Computer Society Press.

[12] S. Zhao, M. J. McGuffin, and M. H. Chignell, "Elastic Hierarchies: Combining Treemaps and Node-Link Diagrams," In *Proc. IEEE InfoVis* (Minneapolis, Minnesota, USA, October 23-25, 2005), pages 57-64. IEEE Computer Society Press.

[13] A. Kobsa, "User Experiments with Tree Visualization Systems," In *Proc. IEEE InfoVis* (Austin, Texas, USA, September 10-12, 2004), pages 9-16. IEEE Computer Society Press.

[14] E. R. Tufte, *The Visual Display of Quantitative Information*. Cheshire, Connecticut: Graphics Press, 1983.

[15] T. Munzner, F. Guimbretière, S. Tasiran, L. Zhang, and Y. Zhou, "TreeJuxtaposer: Scalable Tree Comparison using Focus+Context with Guaranteed Visibility," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 453-462, July 2003.

[16] M. Graham, J. Kennedy, and L. Downey, "Visual Comparison and Exploration of Natural History Collections," In, A. Celentano and P. Mussio, editors, *Proc. Advanced Visual Interfaces (AVI)* (Venice, Italy, May 23-26, 2006), pages 310-313. ACM Press.

[17] E. H. Chi, J. Pitkow, J. Mackinlay, P. Pirolli, R. Gossweiler, and S. K. Card, "Visualizing the Evolution of Web Ecologies," In *Proc. ACM CHI* (Los Angeles, California, USA, April 18-23, 1998), pages 400-407. ACM Press.

[18] K. Wittenburg, D. Das, W. Hill, and L. Stead, "Group Asynchronous Browsing on the World Wide Web," In *Proc. Fourth International World Wide Web Conference* (Boston, Massachusetts, USA, December 11-14, 1995), pages 51-62.

[19] D. R. Morse, N. Ytow, and D. M. Roberts, "Comparison of multiple taxonomic hierarchies using TaxoNote," In *Proc. IEEE InfoVis Poster Compendium* (Seattle, Washington, USA, 19-21 October, 2003), pages 126-127. IEEE Computer Society Press.

[20] M. Ghoniem and J.-D. Fekete, "Animating Treemaps," In *Proc. 18th HCIL Symposium - Workshop on Treemap Implementations and Applications* (University of Maryland, College Park, Maryland, USA, May 31, 2001).

[21] K. Wittenburg and E. Sigman, "Visual Focusing and Transition Techniques in a Treeviewer for Web Information Access," In *Proc. Visual Languages* (Capri, Italy, September 23-26, 1997), pages 20-27. IEEE Computer Society Press.

[22] J. Y. Hong, J. D'Andries, M. Richman, and M. Westfall, "Zoomology: Comparing Two Large Hierarchical Trees," In *Proc. IEEE InfoVis Poster Compendium* (Seattle, Washington, USA, 19-21 October, 2003), pages 120-121. IEEE Computer Society Press.

[23] G. W. Furnas and J. Zacks, "Multitrees: Enriching and Reusing Hierarchical Structure," In *Proc. ACM CHI* (Boston, Massachusetts, USA, April 24-28, 1994), pages 330-336. ACM Press.

[24] T. Dwyer and F. Schreiber, "Optimal Leaf Ordering for Two and a Half Dimensional Phylogenetic Tree Visualisation," In, N. Churcher and C. Churcher, editors, *Proc. Australasian Symposium on Information Visualisation* (Dunedin, New Zealand, January 23-24, 2004), pages 109-115. Australian Computer Society.

[25] D. M. Hillis, T. A. Heath, and K. St. John, "Analysis and Visualization of Tree Space," *Systematic Biology*, vol. 54, no. 3, pp. 471-482, 2005.

[26] Information Visualization Benchmarks Repository, "InfoVis 2003 Contest - Visualization and PairWise Comparison of Trees," http://www.cs.umd.edu/hcil/InfovisRepository/contest-2003/, Last accessed March 28, 2007.

[27] E. A. Wernert, D. K. Berry, J. N. Huffman, and C. A. Stewart, "Tree3D - A System for Temporal and Comparative Analysis of Phylogenetic Trees," In *Proc. IEEE InfoVis Poster Compendium* (Seattle, Washington, USA, October 19-21, 2003), pages 114-115. IEEE Computer Society Press.

[28] M. McGuffin and M. C. Schraefel, "A Comparison of Hyperstructures: Zzstructures, mSpaces, and Polyarchies," In *Proc. ACM Hypertext* (Santa Cruz, California, USA, August 9-13, 2004), pages 153-162. ACM Press.

[29] G. Melançon and I. Herman, "DAG Drawing from an Information Visualization Perspective," In, R. van Liere and W. de Leeuw, editors, *Proc. Eurographics/IEEE TCVG Symposium on Visualization (VisSym)* (Amsterdam, The Netherlands, May 29-31, 2000), pages 3-12. Springer-Verlag.

[30] G. Robertson, K. Cameron, M. Czerwinski, and D. Robbins, "Animated visualization of multiple intersecting hierarchies," *Information Visualization*, vol. 1, no. 1, pp. 50-65, March 2002.

[31] M. Graham, "Visualising Multiple Overlapping Classification Hierarchies," PhD Dissertation, School of Computing, Napier University, Edinburgh, UK, 2001.

[32] P. Eades and N. C. Wormald, "Edge crossings in drawings of bipartite graphs," *Algorithmica*, vol. 11, no. 4, pp. 379-403, 1994.

[33] W. Barth, P. Mutzel, and M. Jünger, "Simple and Efficient Bilayer Cross Counting," *Journal of Graph Algorithms and Applications*, vol. 8, no. 2, pp. 179-194, 2004.

[34] K. Sugiyama, S. Tagawa, and M. Toda, "Methods for visual understanding of hierarchical systems," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 11, no. 2, pp. 109-125, 1981.

[35] R. Martí and M. Laguna, "Heuristics and meta-heuristics for 2-layer straight line crossing minimization," *Discrete Applied Mathematics*, vol. 127, no. 3, pp. 665-678, May 1 2003.

[36] T. A. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, no. 2, pp. 109-133, March 1995.

[37] F. Glover and F. Laguna, *Tabu Search*. Norwell, Massachusetts, USA: Kluwer Academic Publishers, 1997.

[38] E. R. Gansner, S. C. North, and K. P. Vo, "DAG— A Program that Draws Directed Graphs," *Software, Practice and Experience*, vol. 18, no. 11, pp. 1047-1062, November 1988.

[39] E. Catmull and R. Rom, "A class of local interpolating splines," - *Computer Aided Geometric Design*, R. E. Barnhill and R. F. Riesenfeld, eds., New York: Academic Press, pp. 317-326, 1974.

[40] M. Wattenberg, "Visual Exploration of Multivariate Graphs," In *Proc. ACM CHI* (Montréal, Québec, Canada, April 22-28, 2006), pages 811-819. ACM Press.

[41] P. Riehmann, M. Hanfler, and B. Froehlich, "Interactive Sankey Diagrams," In *Proc. IEEE Symposium on Information Visualization* (Minneapolis, Minnesota, USA, October 23-25, 2005), pages 233-240. IEEE Computer Society Press.

[42] J. Nielsen, "Guerrila HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier," - *Cost-Justifying Usability*, 1st ed, R. G. Bias and D. J. Mayhew, eds.: Academic Press Professional, pp. 245-272, Chapter 11, 1994.