Interactive Visualization of Genealogical Graphs

Michael J. McGuffin* Ravin Balakrishnan†

Department of Computer Science, University of Toronto, http://www.dgp.toronto.edu

ABSTRACT

The general problem of visualizing "family trees", or genealogical graphs, in 2D, is considered. A graph theoretic analysis is given, which identifies why genealogical graphs can be difficult to draw. This motivates some novel graphical representations, including one based on a *dual-tree*, a subgraph formed by the union of two trees. Dual-trees can be drawn in various styles, including an indented outline style, and allow users to browse general multitrees in addition to genealogical graphs, by transitioning between different dual-tree views. A software prototype for such browsing is described, that supports smoothly animated transitions, automatic camera framing, rotation of subtrees, and a novel interaction technique for expanding or collapsing subtrees to any depth with a single mouse drag.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—interaction techniques; G.2.2 [Discrete Mathematics]: graph theory

Keywords: genealogy, genealogies, family trees, kinship, multitrees, graph drawing, graph theory, graph browsing and navigation

1 INTRODUCTION

Genealogy, the study of "family trees", plays a significant role in history (e.g. of royal families, and of human migration), genetics, evolutionary biology, and in some cases, religion. It also shows no sign of waning as a hobby of the public, especially given new software tools, databases, and means of communication and sharing made available by the internet.

Unfortunately, the depiction of relationships in a large family is challenging, as is generally the case with large graphs. The diagram in Figure 1, for example, contains many long edges, and doesn't clearly show which nodes are all in the same generation. Although there are a few hundred nodes in the diagram, these are organized around just a few lineages and nuclear families — many lines of ancestry and descent have been omitted. In addition, family trees (or genealogical graphs, as we will call them) are not arbitrary or unconstrained graphs — they have special structural properties that can be exploited for the purposes of drawing and interactive visualization. Interestingly, other than Furnas and Zacks [5], we have been unable to find previous work in the mathematical, graph theory, or graph drawing communities that analyzes the graph theoretic properties of genealogical graphs.

Although genealogical graphs are often referred to as family *trees*, this is misleading. Every individual has a tree of ancestors (sometimes called a *pedigree*), as well as a tree of descendants (Figure 2, left), each of which can be drawn in familiar and easily understood ways. A drawing of both of these trees is sometimes called an *hourglass* chart in the genealogical community, and has been called

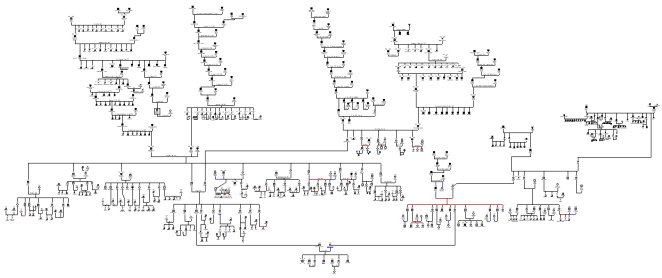

Figure 1: Portion of a genealogical graph for an actual family, laid out manually, containing well over 600 individuals and spanning almost 400 years. (Sample data set supplied with GenoPro [7]).

a *centrifugal view* [5] in the literature. (It is also similar to [21].) Hourglass charts only show some information, however. Each ancestor has themself a tree of descendants, and each descendant has a tree of ancestors (each of whom has a tree of descendants, etc.). It is not uncommon for users to experience frustration with diagramming software, where the user must repeatedly and manually move increasingly large subsets of nodes to create room for new data. It is also not obvious that the underlying structure is best described as a topological tree. Finally, trying to automatically draw such graphs leads to problems and design tradeoffs.

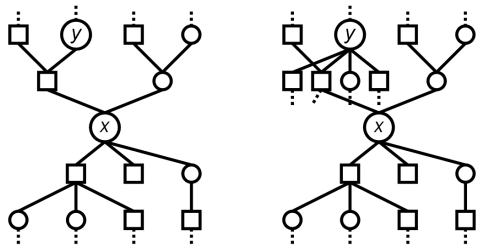

Figure 2: *Left:* Node x has a tree of ancestors (parents, grandparents, etc.) and a tree of descendants (children, grandchildren, etc.), both of which may be drawn with conventional tree-drawing techniques. *Right:* It is more challenging, however, to also show the descendants of y, or worse still, to show the descendants of *every* ancestor of x, and the ancestors of *every* descendant of x. Note: in this and other figures, squares represent males, circles females.

We present a brief analysis of genealogical graphs and identify how and why it is difficult to draw them. This motivates an investigation of alternative graphical depictions, leading to the development of a *dual-tree* scheme that generalizes hourglass charts, and that may be used for visualizing any multitree [5]. We describe a software prototype that implements this scheme, that supports smoothly-animated rotations and transitions between dual-trees, and that uses a novel interaction technique for expanding or collapsing subtrees to any depth with a single mouse drag. Although this work is geared toward genealogy, some of the design principles and techniques used are also applicable in other domains.

*e-mail: mjmcguff@cs.toronto.edu

†e-mail: ravin@dgp.toronto.edu

IEEE Symposium on Information Visualization 2005
October 23-25, Minneapolis, MN, USA
0-7803-9464-X/05/$20.00 ©2005 IEEE.

2 Background

Genealogical relationships have been recorded and depicted for centuries, however the traditional charts appearing in books tend to be simple, usually showing at most a few dozen individuals, and are often organized around simple patterns such as lineages (e.g. one's father, paternal grandfather, etc.), or a single tree of ancestors, or a single tree of descendants. Commercial software packages enable the compilation of datasets with hundreds to thousands of individuals, but are not designed to automatically visualize such large data sets. They either require the user to arrange data manually, or have automatic layout algorithms that only operate on a subset of the data or that don't work well in all cases.

Yet, there is a significant demand for automatic visualization of data. The documentation for [7] states "GenoPro wrote the AutoArrange routine to import Gedcom files, but noticed many are using the AutoArrange to layout their genealogy tree. This routine took several months to write, debug and test, yet generated more emails than all the other features combined. About 95% of all the genealogy trees GenoPro received by email were AutoArranged."

In addition, whether automatically generated or not, conventional charts of large, extended families inevitably contain at least some long edges or nodes displaced far away from their close relatives, to make room for other nodes (e.g. Figures 1 and 5). Thus, even given a robust automatic layout algorithm, it is not clear that displaying *entire* genealogical graphs of thousands of nodes would be ideal, since numerous long edges or edge crossings would make navigation and interpretation difficult.[1] A better solution may be to display subgraphs that are automatically laid out, and allow the user to flexibly transition between subgraphs.

Bertin [2] mentions an elegant way of drawing genealogical graphs, where each individual is a single line segment (thick for men, thin for women) and where nuclear families are points. Each line segment may connect two nuclear families: one in which the individual is a parent, and one in which they are a child (this is similar to p-graphs [22]). Although such diagrams are much simpler looking than traditional ones, they ultimately suffer from the same exponential crowding (see § 3.4).

Ted Nelson has proposed zzstructures (the generic name for ZigZag®) as a general structure for storing information. It has been shown [12] that zzstructures are equivalent to a kind of directed graph. Nelson has demonstrated that genealogical graphs can be encoded within zzstructures, using the scheme in Figure 4, D. The choice of this scheme, however, is due more to its compatibility with typical zzstructure visualizations, rather than due to an inherent appropriateness for genealogical graphs. For example, many visualizations of zzstructures are based on a 2D cursor centric view (described in [12]), which can show one nuclear family at a focal point (parents and children arranged along perpendicular directions), surrounded by some extended family nodes. Unfortunately, such visualizations make it difficult to see which nodes are all within the same generation.

Multitrees [5] are a kind of directed acyclic graph (DAG) where any two nodes are either connected by zero or 1 directed paths. In other words, multitrees are diamond-free DAGs, where a *diamond* is a pair of distinct directed paths from one node to another node. As a consequence, every node x in a multitree has a tree $D(x)$ of descendants and a tree $A(x)$ of ancestors (Figure 3). Furthermore, the trees in a multitree can overlap: given nodes x and y in a multitree, $D(x)$ and $D(y)$ may share one or more subtrees, and if not, then $A(x)$ and $A(y)$ may share one or more subtrees. Furnas and Zacks [5] explain how genealogical graphs constructed according to Figure 4, C can correspond to multitrees, if there is no intermarriage (i.e. diamonds). They also propose two visualization techniques for multitrees: a centrifugal view (essentially Figure 2, left) and a view of a directed path ("lineage") between two nodes along with children and parents of the path [5].

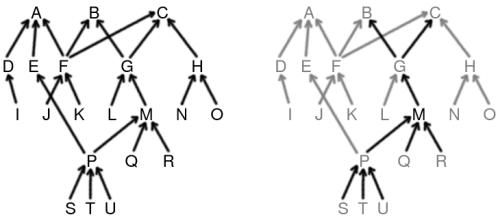

Figure 3: *Left:* an example multitree. Observe that the two trees of descendants rooted at nodes A and C, respectively, share two subtrees, rooted at nodes F and P, respectively. *Right:* Node M is highlighted, along with its tree of ancestors and tree of descendants.

Anthropologists have studied systems of kinship, examining, for example, how family structures and terminology for describing one's kin vary across cultures, and how these relate to genealogy (e.g. [15]). The current work focuses instead on issues relevant to graph drawing and visualization.

Our research differs from the previous work by analyzing in more detail some of the properties specific to genealogical graphs, and by proposing some novel graphical depictions of them. In particular, our dual-tree scheme generalizes the Furnas-Zacks centrifugal view/hourglass chart, and also generalizes the "lineage" view of the same authors [5]. We investigate novel ways of displaying and interacting with dual-trees.

3 Analysis of Genealogical Graphs

In the following, some of the observations and concepts generalize to various non-traditional family arrangements, such as individuals having multiple spouses, or having more than two parents (e.g. adoptive in addition to biological). However, a traditional family model is a useful one to keep in mind, at least initially. Also, for convenience, the word "marriage" is used in a loose sense, to refer to the relationship between the parents of one or more children.

Some in the genealogical community [6] have called for the ability to encode richer information and more kinds of relationships, e.g. foster children, family friends, etc. Increased freedom in a genealogical system would make it approach a general hypermedia system, with a correspondingly general interface. However, we have found that the constraints imposed by first following a traditional family model inspire interesting design and visualization possibilities. Future work may possibly extend or adapt our designs to include more kinds of family relationships.

3.1 Preliminaries

We first establish some terminology to describe relationships between individuals. Beyond the familiar relationships of *parent*, *child*, *ancestor*, and *descendant*, we also consider *consanguine* relatives, i.e. individuals with a common ancestor (also called "blood relatives") such as siblings and cousins. In addition, we define *conjugal* relatives as individuals connected by an undirected path through one or more marriages. For example, brothers-in-law are conjugal relatives, as would be x and any of x's spouse's consanguine relatives.

Cousins are consanguine relatives whose most recent common ancestor occurs at n generations prior to the cousins, and in which case the cousins are $(n-1)$th cousins (i.e. 1st cousins if they share

[1] One anecdote concerning a family reunion recounts how participants exceeded the area of four picnic tables in trying to layout their genealogical information. Another story reports the existence of a single data set containing 30000 interconnected individuals.

a grandparent, 2nd cousins if they share a great-grandparent, etc.). Note that the cousin relationship is not transitive: individual x may have a cousin y on x's maternal side, and another cousin z on x's paternal side, however y and z are not, generally, cousins, though they are related conjugally through the marriage of x's parents. More generally, consanguine relationships are not transitive, but conjugal relationships are, since our definition allows them to pass through multiple marriages.

Finally, we use the term *nuclear family* to refer to (noramlly two) parents and their children.

3.2 Intermarriage and Pedigree Collapse

Intermarriage corresponds to an *undirected cycle* (i.e. a cycle in the underlying undirected graph) in a genealogical graph. We distinguish between two kinds of intermarriage: *Type 1* intermarriage is between consanguine spouses, e.g. spouses who are also (possibly distant) cousins. *Type 2* intermarriage is between spouses who are conjugal relatives via a path going through one or more marriages other than their own marriage. Examples of type 2 intermarriage include two sisters (or cousins) from one family marrying two brothers (or cousins) from another family not initially related to the first family. In the graphs we consider, all marriages are modelled — even those that are eventually dissolved. Thus, if a woman divorces a man x and marries his brother y, this constitutes type 2 intermarriage, because the woman was already conjugally related to y through her first marriage to x.

Assuming that the ancestry of an individual x is free of type 1 intermarriage, then x has 2^n ancestors at the nth generation prior to x. At a conservative 30 years per generation, this exponential number of ancestors exceeds the physical capacity of the earth at less than 2000 years into the past. We can therefore conclude that the ancestry of x must contain type 1 intermarriage. The phenomenon of encountering type 1 intermarriage in *every* individual's ancestry, when traced back far enough, is called *pedigree collapse* [18].

In addition, statistical modelling suggests that all humans alive today share a (not necessarily unique) common ancestor who lived just a few thousand years ago [17], implying that all living humans are "blood relatives".

Pedigree collapse guarantees that type 1 intermarriage occurs in every real-life genealogical graph, if extended back far enough in time. The presence of such diamonds in one's "tree" of ancestors obviously creates problems for drawing such a graph. Fortunately, many genealogical data sets are free of intermarriage because they do not extend back far enough in time, and in any case are usually locally free of intermarriage. Furthermore, algorithms and visualization techniques designed for acyclic graphs may be adapted to genealogical graphs containing intermarriage, by creating virtual duplicates of individuals to "hide" the cycles.

3.3 Conditions Resulting in Trees, Multitrees, and DAGs

When are genealogical graphs really trees, or multitrees, or neither? This depends on the presence of type 1 and type 2 intermarriage, and on which scheme is used to construct the genealogical graph.

Let G be a genealogical directed graph (digraph) constructed according to one of the schemes B–E in Figure 4. If scheme B or C or E is used, then edges are always incident from younger to older nodes, thus G is a DAG. If scheme B or C or E is used, and there is no type 1 intermarriage (which would correspond to a diamond in G), then G is a multitree. If scheme B or D or E is used, and there is no type 1 or type 2 intermarriage, then the underlying undirected graph G' is a free tree (also called a topological tree).

In many cases, then, a genealogical graph may be a free tree, or at least a DAG. Trees are planar, and many techniques exist for drawing them with no edge crossings. However, it is often desirable to see the nodes in a genealogical graph ordered by time, to make the generations in the graph apparent. Such an ordering is impossible to achieve in general without edge crossings. Partially relaxing the ordering by generation, so that each node is only "locally ordered"[2] with respect to its parents and children, allows edge crossings to be eliminated in a free tree. However, long edges are still generally unavoidable (Figure 5).

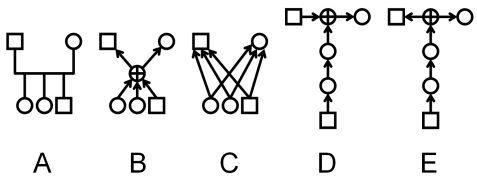

Figure 4: *A:* conventional notation for a nuclear family: squares are male, circles female, and children extend downward from an edge connecting the parents. *B–E* show different ways of modelling such a family within a directed graph. *B:* the $\oplus$ symbol denotes a "spousal union" node. *C:* alternative scheme that avoids any special, intermediate node, but requires more edges when there are 3 or more children. *D:* Nelson's scheme for encoding families within zzstructures. Each child links to its next older sibling, and the eldest child links to the "spousal union" node. *E:* a variation on *D* that prevents cycles in the directed graph.

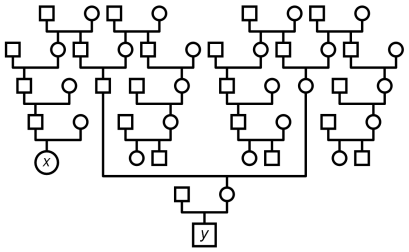

Figure 5: Example situation where a long edge cannot be avoided, even if some branches are rotated. Also, the vertical ordering of nodes by generation is broken: it is not immediately apparent that nodes x and y are of the same generation — they are 3rd cousins. The ordering by generation could be restored by introducing edge-crossings, but at least one edge would still be long.

DAGs can be drawn automatically using standard algorithms, such as Sugiyama et al.'s [19]. In this case, however, edge crossings and long edges are both unavoidable, and as with any automated graph drawing technique, the output from a 2D DAG embedder is increasingly difficult to use and understand as the size of the graph becomes very large. It is also possible that new algorithms designed with the specific properties of genealogical graphs in mind may scale better than generic DAG embedders.

The "bushiness" apparent in Figure 5 illustrates a core problem in genealogical graphs, of nodes quickly becoming crowded as the graph is extended in various directions. The next section examines and quantifies this problem in more detail.

3.4 Crowding Within Genealogical Graphs

We now consider an idealized, simplified genealogical graph G^*, and show that problems arise in trying to draw even this idealized graph. This motivates some non-traditional visual representations.

Let G^* be a genealogical graph, constructed according to Figure 4, B, where every node has two parents, one sibling of the opposite gender, one spouse of the opposite gender, and where every

[2] In graph drawing terminology, locally ordered means *upward*, and (globally) ordered by generation means *upward* and *layered* by generation.

marriage produces one child of each gender. Also assume that generations are well-defined, e.g. births are synchronized within each generation. Furthermore, G^* contains no intermarriage, hence the underlying undirected graph is a free tree, and thus G^* is planar.

Assume we want to draw a connected subset of G^* such that nodes are all allocated the same size, and nodes in the same generation have the same vertical coordinate, so that each generation corresponds to a single row of nodes.

Figure 6 shows such a drawing, for 9 nuclear families spanning 4 generations. Ellipses indicate the directions in which G^* extends. Intuitively, extending the portion of G^* shown in all directions would require not only crossing edges (to maintain alignment of generations), but also lengthening certain edges to make room for expansion, causing certain spouses and/or siblings to become distant from each other.

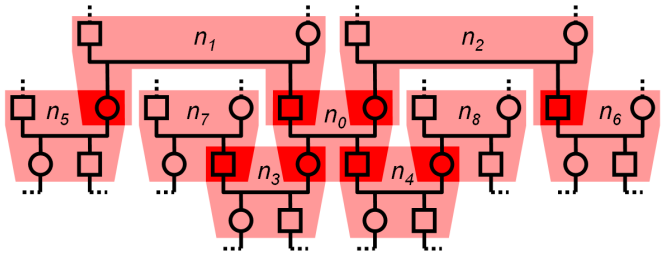

Figure 6: A portion of an idealized genealogical graph, G^*. Nine nuclear families are shown (each outlined in pink), labelled $n_0,\ldots,n_8$. Ellipses indicate the many directions in which this diagram could be extended, suggesting that nodes would rapidly become crowded.

To reinforce this intuition, consider the set S of nuclear families at the same generational level as n_0. Figure 6 shows $S = \{n_0, n_5, n_6, n_7, n_8, \ldots\}$. Notice that n_0 is connected (via the intermediary families n_1, n_2, n_3, n_4) to 4 other families n_5, n_6, n_7, n_8 in S. Following the ellipses, each of n_5, n_6, n_7, n_8 is connected (again through intermediaries) to 3 other nuclear families in S, each of which is in turn connected to another 3, etc. Even though S corresponds to a single generation of nuclear families, the paths connecting families in S correspond to a free tree, and the number of nuclear families in S that are r edges away from n_0 grows exponentially with r. Similarly, if we consider connections through increasingly distant ancestors, each node has 1 sibling, 4 first cousins, 16 second cousins, and 4^n nth cousins. Unfortunately, these nodes must be fit within a 1-dimensional row, where the space available only grows linearly with the geometric distance from the centre of the diagram. The consequence is that the edge-length-to-node-size ratio becomes arbitrarily high.

This is reminiscent of Munzner's observation [13] that, when embedding a tree in a Euclidean space of any dimensionality, the number of nodes grows exponentially with the level, but the space available only grows geometrically. The case in Figure 6 is qualitatively worse, however, because the "exponential crowding" occurs within *each and every* generation as more and more of G^* is displayed, rather than worsening progressively with deeper levels.

4 Some Alternative Graphical Representations

The rapid crowding of nodes that occurs in genealogical graphs inspired us to explore graphical depictions that show different parts of the graph at different scales. By allocating progressively smaller areas to nodes, we might usefully pack more information into a single representation.

Figure 7 shows a fractal layout for G^*. (More generally, such a fractal layout could also be used to depict any free tree.) There is no limit to the extent of the graph that could be drawn this way, however nodes eventually become imperceptibly small. Also notice that this depiction trades away an ordering of nodes by generation to gain non-crossing edges of bounded length.

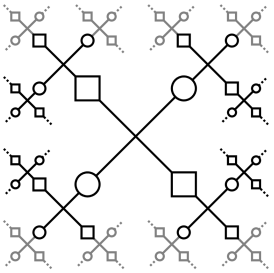

Figure 7: A fractal layout for G^*, showing the same 9 nuclear families as in Figure 6, along with some additional nodes in grey.

Interactive browsing of the tree in Figure 7 could be done by zooming and panning, or by having the user dynamically select the "focal" region that is shown largest in the centre. In the latter case, the resulting interactive visualization might be similar to fisheye graph browsers (e.g. [13]), though it would differ in the details of how nodes surrounding the focal region are shifted and scaled.

In the process of exploring graphical depictions for genealogical graphs, we found it useful to consider the different ways in which rooted trees are represented. Figure 8 shows what we consider to be the most basic styles for drawing rooted trees, 3 of which are identified in [2, 9]. A familiar example of nested containment (Figure 8, B) are treemaps [8]. The indented outline (Figure 8, D) representation may appear to simply be a variation on the node-link (Figure 8, A) representation, but in fact the indented outline style would still be unambiguous without any edges drawn: its essential feature is the use of indentation to imply structure. Many variations on the styles in Figure 8 have been described in the literature, based, for example, on polar coordinate systems, or on embeddings in 3D rather than 2D, or on combinations of existing styles.

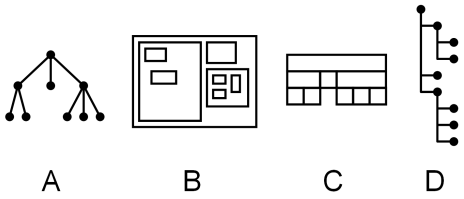

Figure 8: Different graphical representations of the same rooted tree. *A:* node-link. *B:* nested containment, or enclosure. *C:* a layered "icicle" diagram, that uses adjacency and alignment to imply the tree structure. *D:* an indented outline view.

The majority of new tree representations, however, have been applied to rooted trees, whereas free trees are drawn almost exclusively using the node-link style (Figure 8, A). Nevertheless, representations based on rooted trees could be applied to free trees, if the user had a way of dynamically choosing a node to serve as a temporary "visual" root. The user would then be able to see the tree from different perspectives, by transitioning from using one node as a root to another. Such interaction might be useful for temporarily and visually highlighting various regions of the free tree.

This idea allowed us to adapt the nested containment style (Figure 8, B) to genealogical free trees resulting in a novel representation (Figure 9). In general, nested containment representations could be used with any free tree, and thus with any genealogical

graph where there is no intermarriage of type 1 or type 2. However, the representation can be simplified if we assume that, in addition to there being no intermarriage, every node participates in at most two nuclear families: one in which they are a child, and one in which they are a parent (in other words, nodes cannot have multiple spouses in different nuclear families). This assumption allows us to omit the "spousal union" $\oplus$ nodes (Figure 4, B) and leave these implicit, as we have done in Figure 9. In Figure 9, lower left and lower right, each individual corresponds to a rectangle, and each rectangle may have one nuclear family nested within it, and also be part of another nuclear family containing the rectangle. Parents appear in the upper half of a rectangle, and children in the lower half. Note that this representation would easily accommodate the case of nuclear families containing more than 2 parents, by simply subdividing the upper half of rectangles into more than 2 sub-rectangles.

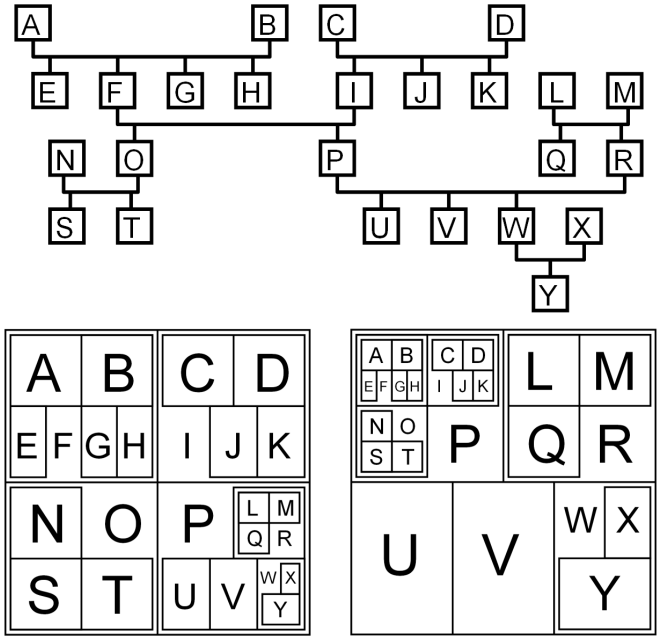

Figure 9: A free tree can be drawn using the nested containment style of Figure 8, B, if the user's current "focus" is used as a temporary root. *Top:* a genealogical graph, drawn using conventional notation. For simplicity, squares are used for all individuals, not just males. *Lower left:* the same graph, drawn using nested containment, with the nuclear family $\{F,I,O,P\}$ as the root. This is analogous to the representation in Figure 7, with larger nodes *containing* smaller nodes rather than being connected to them with line segments. *Lower right:* now, the nuclear family $\{P,R,U,V,W\}$ is the root.

5 Dual-Trees

Although the novel representations in Figures 7 and 9 are interesting, they do not order nodes by generation. Their unfamiliarity might also make them difficult to interpret for many users. We now describe a scheme that is closer to traditional diagrams.

The general problem of scaling a visualization to graphs of thousands of nodes, and the added problem of dense crowding in genealogical graphs, convinced us to focus on visualizing only a subset of the graph at a time, and therefore to identify which subset might be best. Some general questions to ask in such a situation are: What are the *canonical*, or standard, subsets of the data that would be familiar to users? Which of these canonical subsets, or *combinations* of them, can be shown at once in a manner than is *easy to interpret* and that *scales well*?

In the case of genealogical graphs, two obvious canonical subsets are trees of descendants and trees of ancestors. As already mentioned, showing both of these at once (Figure 2, left; Figure 10, A) results in an hourglass chart. To show more information, we propose offsetting the roots of the trees with respect to each other, as in Figure 10, B. The result, which we call a *dual-tree*, is a more general kind of union of two rooted trees. (The result can also be thought of as a single free tree, or a "doubly rooted tree", following the observation in [5] that the ancestors and descendants of a directed path in a multitree form a free tree.)

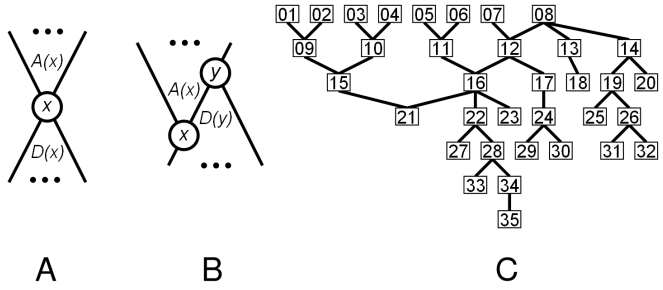

Figure 10: Combinations of canonical subsets of genealogical graphs. *A:* The tree $A(x)$ of ancestors and tree $D(x)$ of descendants of x form an hourglass diagram. *B:* This dual-tree scheme shows more information, by showing $D(y) \supset D(x)$. *C:* An example dual-tree.

The dual-tree $A(x) \cup D(y)$ contains a superset of the information in an hourglass chart, because $A(x) \supset A(y)$ and $D(y) \supset D(x)$. In an hourglass diagram of $A(x) \cup D(x)$, the choice of x is a tradeoff between the number of ancestors and number of descendants revealed: choosing x in an older generation reveals a larger tree of descendants, but reduces the number of ancestors shown. In contrast, with dual-trees, we can always choose x and y to be in the most recent and oldest generations, respectively, to maximize the coverage of the subset displayed.

Because a dual-tree diagram consists of only 2 trees, it can be drawn in a straight-forward manner, and may prove to be easy to understand and interpret. It can be drawn with no edge crossings, with nodes ordered by generation, and it scales relatively well, since the crowding of nodes within it is no worse than the crowding that occurs in individual trees.

To combine two trees in the style of Figure 10, B and C, the root y of the tree of descendants must be a right-most node in the tree $A(x)$ of ancestors. Likewise, x must be a left-most node of $D(y)$. Thus, changing x or y generally requires rotating subtrees to make the new roots right- and left-most. One scenario in which the dual-tree might be particularly useful is in families where surnames are passed down from the paternal side. In such a family, if y is chosen to be the oldest paternal ancestor of x, then the dual-tree would simultaneously contain every ancestor of x (in $A(x)$), as well as every individual having the same surname as x (in $D(y)$), or alternatively every individual having the same surname as any chosen ancestor of x. We are not aware of any other traditional and scalable depiction of families that can show this. For example, Figure 13 shows Tom Smith, his ancestors, and other Smiths in single dual-tree.

Figure 10, C is based on the node-link style of drawing trees (Figure 8, A). The indented outline style (Figure 8, D), however, is often more space-efficient, especially when nodes have long text labels, so we tried to adapt it to dual-trees. Figure 11 shows the steps involved in this. The key to combine the two trees was to use an alternative convention for drawing edges taken from Venolia and Neustaedter [20], and analogous to the left-child, right-sibling pointer implementation of tree data structures [4]. The result in Figure 11, C accommodates long text labels to the right or left of

nodes without requiring new whitespace to be introduced between nodes, as would be the case in Figure 10, C.

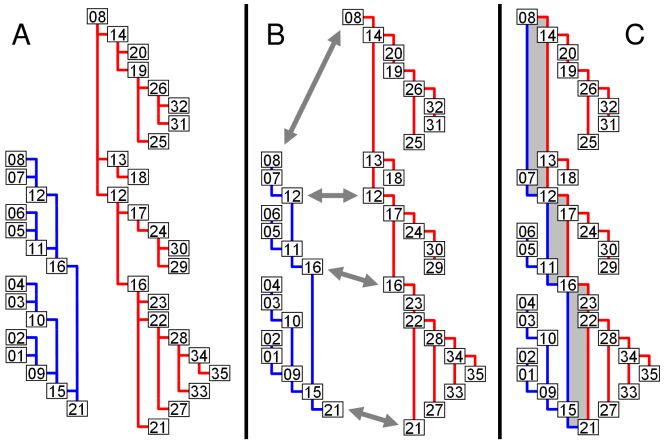

Figure 11: Three stages in adapting the indented outline style to dual-trees. The nodes and labels are the same as in Figure 10, C. *A:* Each tree is drawn in indented outline style. *B:* Edges are drawn in an alternative way, to clear the space between the trees. Arrows show matching nodes in both trees. *C:* The two trees combined.

Note that dual-trees can be used to browse and visualize any multitree, even if some nodes have multiple spouses, or there is type 2 intermarriage, or some nodes have more than 2 parents. Note also that, in Figures 10, C and 11, C, nodes in the same generation are clearly shown as such, as they correspond to a single row or column, respectively.

6 Software Prototype for Dual-Trees

To experiment with browsing based on dual-trees, a software prototype was developed, written in C++ using OpenGL and GLUT. The prototype reads in a GEDCOM file as input, from which a directed graph is constructed according to Figure 4, C.

The digraph is then pre-processed to remove directed cycles and diamonds to obtain a valid multitree. To do this, a breadth-first traversal identifies all undirected cycles in the underlying undirected graph. For each cycle, we count the number of times the edges change direction along the cycle, yielding a non-negative even integer. If the result is zero, we have a directed cycle in the digraph; if the result is 2, we have a diamond; if the result is 4 or more, this may or may not correspond to type 2 intermarriage but in any case is allowed in a multitree. So, if the result is zero or 2, we mark one of the edges involved to be skipped in the embedding algorithm. The display routine, however, can draw these special edges in an alternative colour, to highlight them.

The prototype only displays one dual-tree subset of the graph at a time, but allows the user to interactively transition from subset to subset and browse the entire graph, which might be very large. Each time a new dual-tree subset is chosen, the embedding routine is invoked to determine its layout. Two styles of layout are supported: classical node-link style, and indented outline style. Regardless of the style used, the embedding involves two stages: first, computing two preliminary embeddings E_A and E_D of the tree of ancestors and the tree of descendants, respectively, and second, combining E_A and E_D into a final embedding E_F of the dual-tree.

In the case of classical node-link layout, E_A and E_D are computed with an adaptation of the Reingold-Tilford algorithm [16], though a slightly better implementation would use Buchheim et al.'s improvements [3]. To combine E_A and E_D and produce E_F, the embedding routine shifts E_D so that it is beside E_A, such that nodes in the same generation are aligned. Next, consider the set of nodes that appear in both trees, which we call the *axis* of the dual-tree, i.e. the path between the two roots. Each node n in the axis has a position p_A given by E_A and a position p_D given by E_D. The final position of n is computed as the weighted average $p_F = (ap_A + dp_D)/(a+d)$ where a and d are the number of ancestors and descendants, respectively, of n. Our rationale for this weighting is that we don't want the change in n's position to result in many edges having an extreme slope; thus, the more edges n has in one of the trees, the closer its final position should be to its position in the preliminary embedding of that tree.

In the case of the indented outline layout, E_A and E_D are computed in a simple recursive bottom-up pass. Next, pairs of consecutive nodes on the axis are "stretched out" so that E_A and E_D match up along the axis, and finally E_F is produced (cf. Figure 11, B, C).

In both cases, the time required for the entire embedding process is linear in the number of nodes embedded.

Figure 12 shows screenshots of output. The classical node-link layout can be done along two different orientations (Figure 12, top left and bottom left) yielding different total areas and aspect ratios. The area of the bounding rectangle for the indented outline layout (Figure 12, right) tends to be smaller than that of the other two layout styles, however its aspect ratio also tends to be far from 1. Such an aspect ratio could be an advantage, however, as it could simplify navigation, requiring the user to scroll mainly along just one direction in a zoomed-in 2D view. Figure 13 shows the visual design of nodes in more detail.

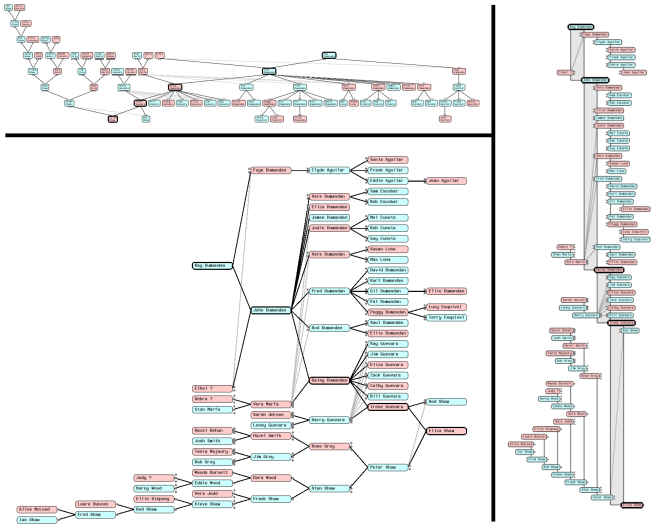

Figure 12: A dual-tree laid out 3 different ways by the prototype. Nodes are coloured by gender. *Upper Left:* Classical node-link, similar to Figure 10, C, with generations progressing top-to-bottom. *Lower Left:* Classical node-link, with generations progressing left-to-right. *Right:* Indented outline style, similar to Figure 11, C.

6.1 Interaction Techniques

To transition between different dual-tree subsets, the basic operations performed by the user are: expanding/collapsing parents of a given node, and expanding/collapsing children of a node. These actions can be invoked through a 2-item marking menu [10] affording ballistic "flick" gestures, in the direction of parents or children, to toggle their expansion state. Expanding a node can also cause automatic rotation. For example, if node n is in the tree of descendants, expanding upwards toward its parents requires that n first be

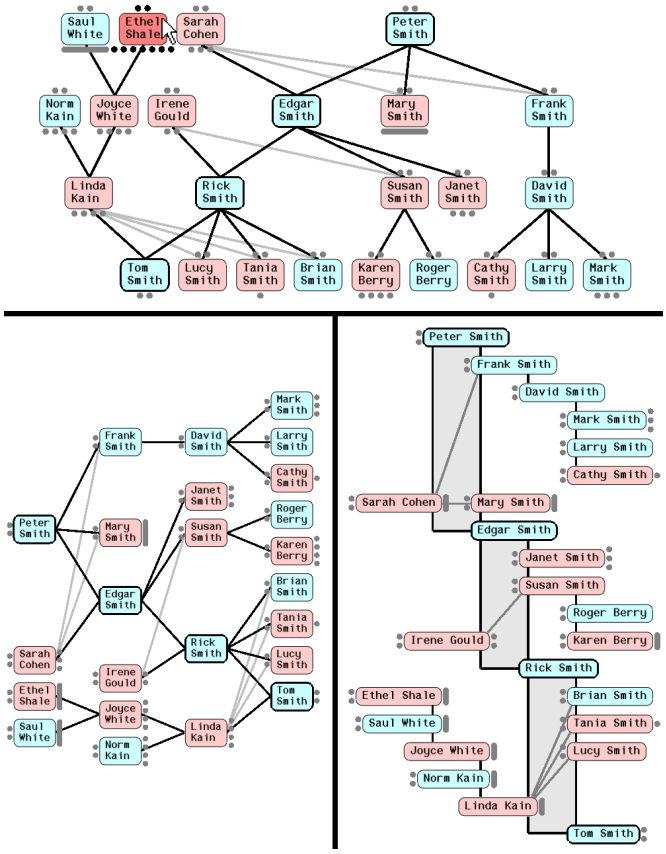

Figure 13: Another dual-tree under 3 layouts: classical node-link top-to-bottom (*Top*) and left-to-right (*Lower Left*), and indented outline style (*Lower Right*). Black edges are part of the dual-tree, and nodes with bold borders lie on the *axis*, or path between the two roots. Additional edges from children to parents are shown in grey, to make nuclear families more apparent. For example, Rick and Susan are siblings, sharing the same parents Irene and Edgar, however Janet is a half-sibling with a different mother, as shown by the lack of a grey edge from her to Irene. Grey dots on either side of a node provide previews of the number of parents or children that the node has. Preview dots that are too numerous are collapsed into oblong shapes (e.g. under Saul and Mary), and shown in full when the cursor rolls over the node, as shown under Ethel (*Top*). Once over a node, the user can reveal hidden parents and children by flicking in the appropriate direction to expand the node.

rotated onto the axis. Such rotations generally require that certain other nodes be collapsed, to maintain the dual-tree scheme.

Expansion, collapsing, and rotation of nodes is shown with smooth, 1-second animations to help the user maintain context [1, 23]. As in [14], our animation has 3 stages: fading out nodes which need to be hidden, moving nodes to their new positions, and fading in newly visible nodes. Although many nodes may need to move in different directions during a transition, the user may benefit from tracking even just a few nodes that serve as visual anchors or landmarks. We feel that even a complicated animation is better than no animation at all, and could always be slowed down if the user wishes with a technique such as the dial widget in [11].

The user may zoom and pan the 2D view of the graph with the mouse, and optionally activate automatic camera framing that is animated during transitions.

In browsing genealogical graphs, we have found it is often desirable to expand downward from an individual to their most recent descendants, or to expand upward to their oldest ancestors. This can be done with the marking menus using a sequence of flicks, with one or more flicks for each generation. However, an even faster method is available through a *subtree-drag-out widget* for "dragging out" subtrees to any depth. To use this widget, the user first clicks down (with a secondary mouse button) on a node (Figure 14, A), and then drags either up or down (i.e., toward ancestors or descendants) to select the subtree on which they want the widget to operate. After this initial drag, the length and colouring of the widget (Figure 14, B) indicate both the maximum depth of the subtree, and also the depth to which the subtree is currently expanded. The user may drag towards the subtree, to expand it further one level at a time, or away from the subtree, to collapse it one level at a time. In keeping with a metaphor of relative adjustment, the user may also release over the centre of the widget, to dismiss it with no effect, which is useful for cancelling.

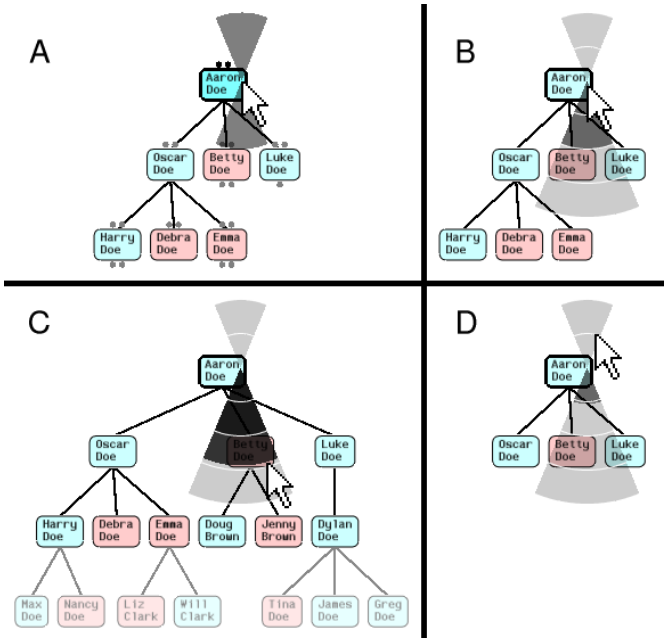

Figure 14: A semi-transparent popup widget for expanding or collapsing subtrees in a single drag. The user pops up the widget over a node (*A*), and may now drag up or down to select whether to operate on the tree of ancestors or descendants. After dragging down slightly, the tree of descendants has been selected (*B*), and now the widget displays the number of levels the user could drag to change this tree: at most 4 levels down to expand to the full depth, or at most 2 levels up to collapse. Furthermore, the first 2 levels down are shaded in to indicate that they are already partially expanded. *C* and *D* show the subsequent feedback after dragging down almost 3 levels, or up 1 level, respectively. Releasing the mouse button completes the operation. The node's tree of ancestors could similarly be expanded or collapsed in a separate invocation of the widget.

After popping up this widget and performing the initial drag to select the subtree to operate on, the user may then drag ballistically to quickly open or close the entire subtree. Although in general subtrees may be quite large after just a few levels, the trees of descendants and ancestors in typical genealogical data tend to be fairly shallow, seldom spanning more than a few hundred years. Furthermore, even though the user may ballistically expand multiple subtrees upward and downward in quick succession, the automatic rotations that result from expansion often cause other nodes to disappear, thus the user is much less likely to experience an "explosion" in the number of expanded nodes.

7 Initial User Feedback

As a first step toward evaluating our prototype and informing design changes, an informal session was held to solicit feedback from a practicing genealogist who has also lectured on genealogy. The user reported using computers an average of 2 hours/day, and is familiar with two common genealogy software packages. The session lasted 1 hour, and consisted of a mixture of free-form exploration by the user, demonstration and explanation by the first author, and semi-structured navigation tasks given to and performed by the user.

After some time interacting with the prototype, the user reported finding it "very clear" and "very easy", and was "impressed with its manoeuvrability". (Note that, at the time the session was held, the subtree-drag-out widget had not yet been implemented. The user did, however, discover and successfully operate the marking menus with no help.)

The user also commented that the "unfamiliarity" of the depiction of family relationships "takes getting used to". The user mentioned the lack of a symbol explicitly linking spouses, which is shown in conventional diagrams.

The user successfully completed all navigation tasks, even though these required expanding upward and downward multiple times, and even when using the indented outline style dual-tree. The user was also able to correctly interpret indented outline depictions, pointing out the parents and children in nuclear families.

The user was also shown printouts of sample output from a commercial genealogy software package, and asked for opinions, comments, or personal preferences in comparing the different diagrams and the output of the prototype. The user seemed rather neutral, and so was given an explanation of some potential positive and negative differences between the dual-tree scheme and other representations. The user remained neutral, however, saying "I can understand [each of the depictions]. [...] I don't know that there are any pros or cons."

Of course, more sessions with other users would be necessary to gain a fuller comparative picture, however we are encouraged by the fact that the user was able to interact with and interpret the output of our prototype.

8 Conclusions and Future Directions

We have analyzed the nature of genealogical graphs, characterized how they are difficult to draw, and presented novel graphical representations for them. In particular, our dual-tree scheme scales as well as a single tree, orders nodes by generation with no edge crossings, is easy to interpret, can be used for browsing any multitree, and generalizes both the hourglass chart/centrifugal view of Furnas and Zacks and the "lineage" view of the same authors [5]. Furthermore, our interaction technique for expanding or collapsing subtrees to any depth with a single mouse drag could be used in other domains for general tree browsing, and might possibly be adapted for general graph browsing.

Dual-trees can show many generations vertically, but have a horizontal extent limited to ancestors or descendants of 2 root nodes. Although these roots can be changed interactively to traverse a data set, only a fraction of a large data set may be visible at any given time. To increase the horizontal extent of nodes shown, without introducing long or crossed edges, the dual-tree scheme could be generalized to a sequence of N trees, laid out left-to-right and alternating between ancestor and descendant trees, all shown at once. Another possibility is to embed combinations of ancestor and descendant trees in 3D, e.g. by arranging intersecting trees on perpendicular planes. Use of 3D could eliminate the need to rotate subtrees, possibly giving the user a more consistent view of the data.

It would also be useful to have graphical representations that are oriented toward higher-level groupings of individuals, such as family units. For example, a viewer for a dual-tree $A(x) \cup D(y)$ could be augmented to also show siblings of nodes in $A(x)$, and spouses of nodes in $D(y)$. This would increase the crowding of nodes somewhat, but would make complete nuclear families visible.

9 Acknowledgements

Many thanks to Derek G. Corneil, Shigeru Owada, Alicia Servera, June E. McGuffin, Daniel Morin, Dallan Quass, Jean Bryan, our anonymous reviewers, and the user who gave us feedback on our prototype, for their valuable support, help, suggestions, and time.

References

[1] L. Bartram. Can motion increase user interface bandwidth? In *Proc. IEEE Conf. Systems, Man and Cybernetics*, pages 1686–1692, 1997.

[2] J. Bertin. *Sémiologie graphique: Les diagrammes, Les réseaux, Les cartes*. Éditions Gauthier-Villars, Paris, 1967. (2nd edition 1973).

[3] C. Buchheim, M. Jünger, and S. Leipert. Improving Walker's algorithm to run in linear time. In *Proc. Graph Drawing (GD)*, 2002.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[5] G. W. Furnas and J. Zacks. Multitrees: Enriching and reusing hierarchical structure. In *Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, pages 330–336, 1994.

[6] G. B. Hoffman. Genealogy in the new times, 1999. http://www.genealogy.com/genealogy/61_gary.html.

[7] GenoPro Inc. GenoPro. http://www.genopro.com/.

[8] B. Johnson and B. Shneiderman. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Proc. IEEE Visualization (VIS)*, pages 284–291, 1991.

[9] D. E. Knuth. *The Art of Computer Programming, Volume I: Fundamental Algorithms*, pages 309–310. Addison-Wesley, 1968.

[10] G. Kurtenbach and W. Buxton. The limits of expert performance using hierarchic marking menus. In *Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, pages 482–487, 1993.

[11] M. J. McGuffin, G. Davison, and R. Balakrishnan. Expand-Ahead: A space-filling strategy for browsing trees. In *Proc. IEEE Symp. Information Visualization (InfoVis)*, pages 119–126, 2004.

[12] M. J. McGuffin and m. c. schraefel. A comparison of hyperstructures: Zzstructures, mSpaces, and polyarchies. In *Proc. 15th ACM Conference on Hypertext and Hypermedia (HT)*, pages 153–162, 2004.

[13] T. Munzner. H3: Laying out large directed graphs in 3D hyperbolic space. In *Proc. IEEE Symp. Information Visualization (InfoVis)*, 1997.

[14] C. Plaisant, J. Grosjean, and B. B. Bederson. SpaceTree: Supporting exploration in large node link tree. In *Proc. IEEE Symp. Information Visualization (InfoVis)*, pages 57–64, 2002.

[15] D. W. Read. Formal analysis of kinship terminologies and its relationship to what constitutes kinship. *Mathematical Anthropology and Cultural Theory*, 1(1), November 2000. 46 pages.

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

[17] D. L. T. Rohde, S. Olson, and J. T. Chang. Modelling the recent common ancestry of all living humans. *Nature*, 431(7008), 2004.

[18] A. Shoumatoff. *The Mountain of Names: A History of the Human Family*. Simon & Schuster, Inc., 1985.

[19] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-11(2):109–125, February 1981.

[20] G. D. Venolia and C. Neustaedter. Understanding sequence and reply relationships within email conversations. In *Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, pages 361–368, 2003.

[21] J. Wesson, MC du Plessis, and C. Oosthuizen. A ZoomTree interface for searching genealogical information. In *Proc. ACM AFRIGRAPH '04*, pages 131–136, 2004.

[22] D. R. White and P. Jorion. Representing and computing kinship: A new approach. *Current Anthropology*, 33(4):454–463, 1992.

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