Visualizing large genealogies with timelines

Daniyar Mukaliyev

Master's Thesis

ITÄ-SUOMEN YLIOPISTO

Faculty of Science and Forestry

School of Computer Science

August 2015

UNIVERSITY OF EASTERN FINLAND, Faculty of Science and Forestry, Joensuu
School of Computing
School of Computer Science

Student, Daniyar Mukaliyev : Visualizing large genealogies with timelines
Master's Thesis , 56 p. Supervisors of the Master's Thesis : Professor Markku Hauta-Kasari, PhD, Professor Pekka Kilpeläinen, PhD
August 2015

Abstract:

As people tend to understand complex information better in visual form, the visualization of genealogy data becomes very useful in a variety of fields, especially in the medical field. However, the visualization of genealogical data is a challenging task as is the case with general graphs.

In this study we have done a review of existing genealogy data visualization layouts and techniques. The analysis of the review showed that there are no existing solutions that could handle big genealogy datasets and at the same time visually encode temporal information. That motivated us to extend the Dual-tree layout technique that handles the scalability problem and visually encode temporal information into it. To test the proposed solution with a real world dataset a software prototype was build and described. It allows to build dynamic visualizations with a user interaction technique and smooth animation.

Keywords: genealogy, family trees, Dual-tree, TimeNets, lifelines

Preface

This thesis is a final project of the Masters degree studies in Information Technology at the University of Eastern Finland. This thesis work is done in collaboration with the NeuroCenter of Kuopio Hospital University.

I am deeply thankful to my supervisor Professor Pekka Kilpeläinen for all his support, guidance and his time spent on reviewing my work in such a great detail. He could always show me the right directions to move with this study and find the right words to encourage me whenever I felt stuck in a problem.

I am deeply grateful to Professor Markku Hauta-Kasari for giving me opportunity to work on that topic, his comprehensive support in organizing the meetings with the Hospital Professors and being so responsive.

Furthermore I would like to thank Professor of Neurosurgery, Juha E. Jääskeläinen for giving the idea to conduct that research and shaping the requirements for the software system. I would also like to thank Associate Professor of Neurosurgery, Mikael von und zu Fraunberg for his help and excellent explanations of the data attributes.

Finally I would like to thank my family for being very supportive throughout my studies.

Tampere, August 2015
Daniyar Mukaliyev

List of Abbreviations

ACM	Association for Computing Machinery
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
DOI	Degree-of-Interest
GEDCOM	Genealogical Data Communications
HTML	Hypertext Markup Language
ISY	Itä-Suomen yliopisto
JSON	JavaScript Object Notation
UEF	University of Eastern Finland
W3C	World Wide Web Consortium
XML	Extensible Markup Language

Contents

1 Introduction **1**
1.1 Motivation 1
1.2 Problem 1
1.3 Proposed solution 2

2 Literature review **4**
2.1 Visualization in genealogy 4
2.2 Classical Hierarchical Tree 6
2.3 Radial view 8
2.4 H-Tree Layout 10
2.5 Tree-map Layout 12
2.6 Indented Outline Layout 13
2.7 TimeNets Layout 16
2.8 Genograms 22
2.9 Focus + Context 24
2.10 Dual-tree 27
2.11 3D Visualization 32

3 Proposed solution **34**
3.1 Encoding temporal information 36
3.2 Ancestors tree 38
3.3 Tree of descendants 39
3.4 Connecting ancestors and descendants trees 41

4 Implementation **44**
4.1 Genealogy data format 45
4.2 Server-side implementation 46
4.3 Front-End implementation 49

5 Discussion and Conclusion **50**

1 Introduction

1.1 Motivation

Advances in technology gave a new impulse to genealogy, the study of family history, in the past few decades. New tools facilitate and enable new ways of searching, storing and processing big amounts of genealogy information. As a part of family history, medical family history represents a special interest in the medical studies. By analyzing medical records is possible to estimate the risk of a person to have a certain disease and to apply preventive actions on time. Previously genealogy included very little of a family medical history, but nowadays big amounts of medical history information are available for medical geneticists.

The rapid increase of amounts of genealogy data requires new helper tools to analyze it. Encoding that data into visual form makes the information perception easier and people understand that complex information better. Thus improving and building new genealogy data visualization tools can make analysis process easier and more accurate. Those tools will be especially useful for genetics researches, to help them to visualize the genes flow and to track the potential paths of genetic disease inheritance.

1.2 Problem

There are many different ways to visualize genealogy data. One of the most popular ways is to visualize it in the form of a genealogical graph that is often called *family tree*. The typical layout for those graphs is shown in Figure 1. The nodes on that graph represent individuals and its edges represent relations between these individuals. Unfortunately, standard layout types of these graphs have issue with scalability. Depicting big amount of nodes leads to exponential crowding and edge crossings [26]. When a graph gets bigger than a few hundred nodes, the length of the edges and the number of edge crossings increases. That leads to information perception difficulties and the visualization loses its initial purpose.

Another issue with the standard layout of genealogical graphs is that it does not allow encoding all the necessary and useful information. Dates of birth and death, dates when a disease was found and cured etc. are missing and are not visually encoded in typical

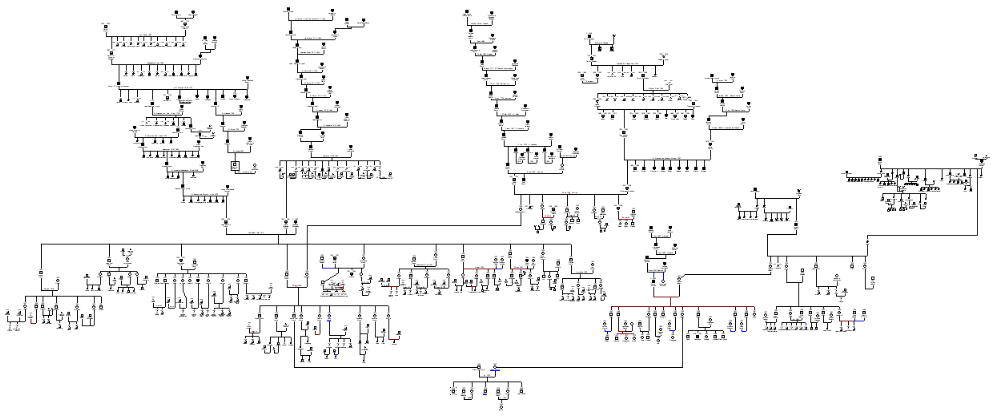


Figure 1: Genealogy graph [26].

family trees [22]. However, that temporal information is important and brings a lot of value to genetics researches. The main research question of this thesis is to investigate what are useful and applicable models and methods that solve the scalability problem and encode temporal information.

1.3 Proposed solution

When the number of individuals in the family tree is significant it becomes difficult to fit all of them on one computer screen due to its size limitations. At the same time it is hard to perceive all the information at once. Enabling the user to interact with a tree and allowing to collapse and expand some branches or subtrees of the tree may help to overcome that problem. To solve the scalability problem mentioned above McGuffin and Balakrishnan [26] proposed an interaction technique to display only a part of a family tree and to enable the user to navigate through the whole tree. Using that technique the user can explore different parts of the tree as well as expand and collapse subtrees. They called that approach a "*Dual-tree*", which stands for a *tree of ancestors* and a *tree of descendants* (Figure 2). That tree contains the whole ancestors tree of the central person and the tree of descendants for one of the central person's ancestors. By selecting another ancestor a new tree of descendants is built and displayed. Using that technique they could visualize a family tree consisting of a few thousand individuals. Even if that family tree visualization approach can solve the scalability problem it still does not include any temporal information.

Only a few family tree layouts that incorporate also temporal information seem to have

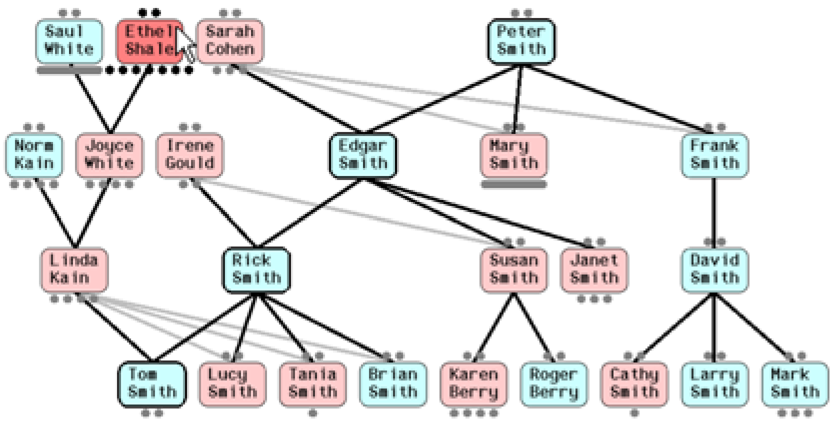


Figure 2: A Dual-tree combines a tree of ancestors with a tree of descendants [26].

been presented in the literature. Nam Wook Kim et al. [23] proposed a "*TimeNets*" layout. In that layout every individual is represented by a line where the start of the line signifies the birth of the person and the end of the line the death of the person. In their approach they also encode information about the date of a marriage in a form of a curve on the lifeline, but that information does not represent a special interest for the genetics scientist. In that layout children are represented as dropdown lines from their parents line. That TimeNets approach handles scalability problem by assigning the *Degree-of-Interest* (DOI) points to every individual and discards those nodes whose DOI is under the threshold, or if they do not fit on the screen. Omitted lines may contain important information and there is no proposed way to get these lines displayed on the screen if needed.

Genelines timeline software[1] follows the same method of representing individuals as lifelines but the layout differs in a way that all the children are placed between their parents. A limitation however is that it can only display a tree of ancestors of the central person. Thus, important information about descendants and close relatives such as uncles, aunts and cousins is missing.

To overcome all those limitations and to visually encode temporal information a new genealogy data visualization layout was designed and explained in the present thesis. The feature of Dual-tree to handle scalability problem by displaying only a subtree of a tree at a time and allow dynamically traverse the whole tree was applied and extended by visually encoding temporal information. The style of rendering an individual as a line was taken from the TimeNets and Lifelines methods and merged with the Dual-tree

[1]http://progenygenealogy.com/products/timeline-charts.aspx

technique. A software prototype was also built to test and to evaluate the proposed approach.

The rest of the thesis is organized as follows. In Chapter 2 we review existing genealogy data visualization layouts and tools. We analyze the pros and cons that these approaches have, how they handle the scalability problem and their capability to visually encode temporal information. Chapter 3 presents our solution of extending the Dual-tree approach by visually encoding temporal information into it. In Chapter 4 we present the implementation details of the prototype system of our solution that was discussed in Chapter 3. Chapter 5 is the final chapter that makes the final conclusion of the study and gives recommendations for further improvements.

2 Literature review

In this chapter we look at different tree visualization technologies. We are especially interested in different approaches of adapting the most common tree visualization techniques to genealogy data. We also introduce some techniques that help users to interact with the visualization and to explore it in more details.

The motivation for reviewing different layouts is to try to understand how existing layouts handle the scalability problem and how they can be adapted to visually encode temporal information.

2.1 Visualization in genealogy

People tend to have interest in their origin, to know a place for their family in the larger picture [11]. More over, in Middle Ages the origin of a person played an important role in determining the social status of an individual, his belonging to a particular class, caste etc. People kept the story of their family and passed it from one generation to another. Visualization of such a story is often used to help to understand, share and pass it to the next generations. Nowadays, with the increasing amount of genealogy data available, the role of the visualization has increased as well. Visual information helps people to receive and understand data faster than any other medium; the highest bandwidth between the computer screen and the human gives the ability to comprehend huge amounts of data [9]. The most widely used method of visualizing family history is to visualize it in a form of a tree often called a *family tree* (Figure 3). Trees are widely used to represent and to visualize different hierarchical data sets. Hierarchical data sets store two kinds of information: the structural information that is the hierarchy, and the content information linked with each node [21]. There are many different layouts for tree visualization and they can provide different meaning even if they represent the same data.

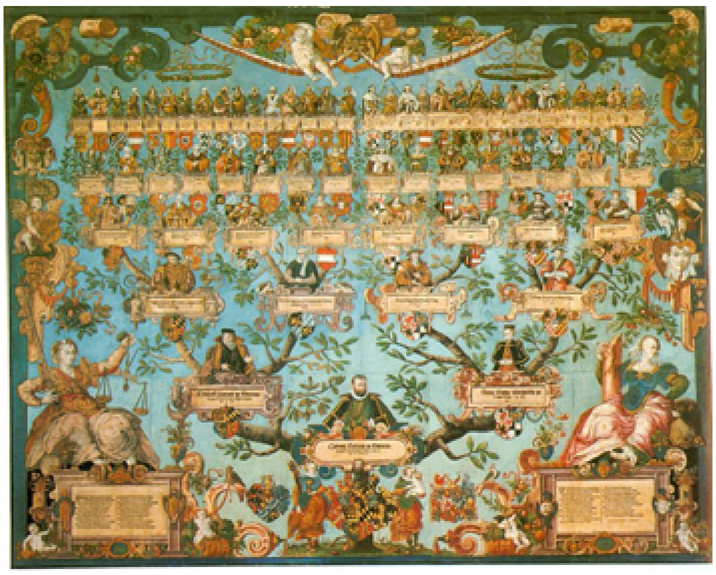


Figure 3: Family tree of Duke Ludwig (1568–1593) [2].

According to Herman et al. [19], the basic tree drawing task is to calculate the position of every node in the set and to draw edges to represent the relations between them. Such on the first glance simple task can turn to be very complicated or even undoable if aesthetic rules are applied. The distribution of nodes and links should be even, nodes that have the same depth level should be aligned, edge crossings should be avoided, edges should be straight lines etc. C. Purchase conducted usability studies which lead her to conclude that “reducing the crossings is by far the most important aesthetic, while minimizing the number of bends and maximizing symmetry have a lesser effect” [27].

The main problem that most of tree layouts suffer from is how they handle the size of the tree. Some layouts may effectively handle a tree with a few hundred of nodes but fail when it has a few thousand of nodes. A big number of nodes and edges leads to a high density of nodes, and it becomes impossible to interact with the tree and get any information from it. Thus careful analysis of the layouts is needed to choose the one that will solve a particular problem the best. It is beyond the scope of this paper to describe all the different types of layouts, however, we cannot skip mentioning the most common ones:

- Classical Hierarchical Tree
- Radial-view
- H-tree layout
- Tree-map layout
- Intended Outline Layout.

These layouts will be briefly discussed in the following sections, as well as some most popular genealogy software packages.

2.2 Classical Hierarchical Tree

One of the best known tree layouts is the *Classical Hierarchical Tree* depicted in Figure 4. It is node-link representation where parents placed above their children and lines that connect nodes represent their relationships. There is also a variation of this top-down layout where children are placed to the right of their parents [19, 20]. That kind of layout does not properly handle the above mentioned problem with the size of the tree. It leaves a lot of unused space and some parts of the tree have a high density of nodes making it difficult to interact and extract visual data [15]. Another problem with the classical tree layout is the only small amount of information can be visually encoded. If we add to each node some additional information along with a text label, it “quickly overwhelms the display space for trees with more than just a few nodes” [21].

Applying Classical Tree layout to genealogical data we can get a *tree of ancestors*, often called a *pedigree chart* (Figure 5, *a*), and a tree of descendants (Figure 5, *b*). Combining both of those trees together is called an *hourglass chart* as well as *centrifugal view* (Figure 5, *c*) [26]. By modifying visual attributes of nodes we can encode some additional data. Usually those types of trees encode gender information by depicting a node as a rectangle if the person is male and as a circle if female. Filling a

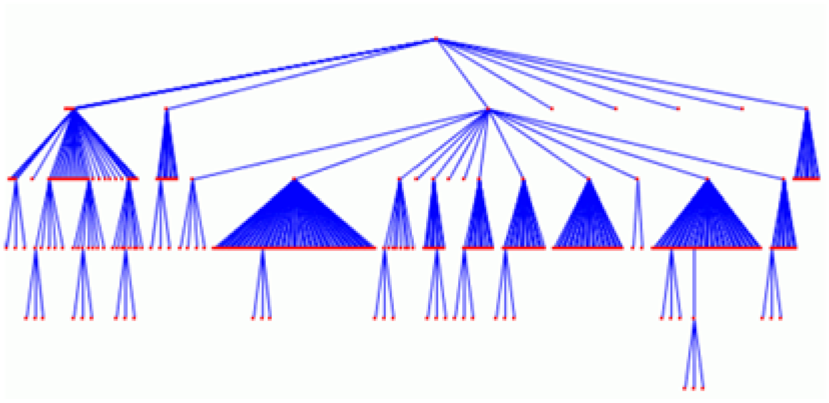


Figure 4: Classical Hierarchical Tree [19].

node with some color can help encoding some medical information, e.g., if a person was affected by a disease or not.

If we take a closer look at the tree in Figure 5, *c* we can notice that it does not show all the relationships from geological data. Every ancestor has a tree of descendants (Figure 5, *d*) as well as every descendant has a tree of ancestors [26]. Now if we try to depict all these relationships in one chart it will eventually lead to edge crossings. Edge crossings on its own turn will lead to difficulties in data perception. Moreover there is no existing solution on how to encode temporal information on that kind of chart except adding text labels.

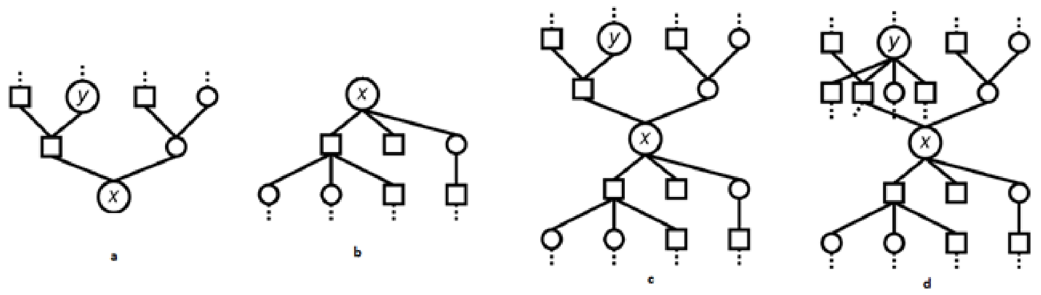


Figure 5: a: Ancestors tree. b: Descendants tree. c: Hourglass chart d: Expanded hourglass chart [26].

The advantage of the Classical Hierarchical Tree layout is that it is relatively easy to implement using existing algorithms, e.g., proposed by Reingold and Tilford [28, 19]. It also provides a good separation of generations as the nodes that represent persons belonging to the same generation are aligned.

2.3 Radial view

Although the classical tree layout is easy to understand when the number of nodes is not big, it requires a lot of effort “to achieve a mental model of the structure in large hierarchies” [21]. Moreover this layout does not use the available space efficiently as more than 50% of the pixels are used as background [16]. Thus more space efficient layouts are needed to be explored. The *radial tree layout* (Figure 6, *a*) is a variation of the node-edge tree representation. Unlike the classical tree layout where child nodes are placed beneath or above their parents, in the radial layout they are placed on an outer orbit radiating from their parents.

The positioning algorithm places nodes on concentric circles depending on their depth level in the tree structure. When a subtree is placed on the sector of a circle the algorithm ensures that adjacent sectors do not overlap [19]. It makes use of the available space more efficiently than the classical tree layout. In the classical tree layout, if every node has more than one child the shape of the tree is relatively triangular resulting in a high density of the nodes. In the radial layout, however, more space becomes available for every next level of the tree since the length of each orbit increases with the radius.

Even though radial layout makes the use of available space more efficient than classical tree layout, it still does not make it optimal. Typically the number of nodes increases exponentially with the every consecutive level but the length of each consecutive orbit increases linearly, and thus the density of the nodes increases in the outer circles.

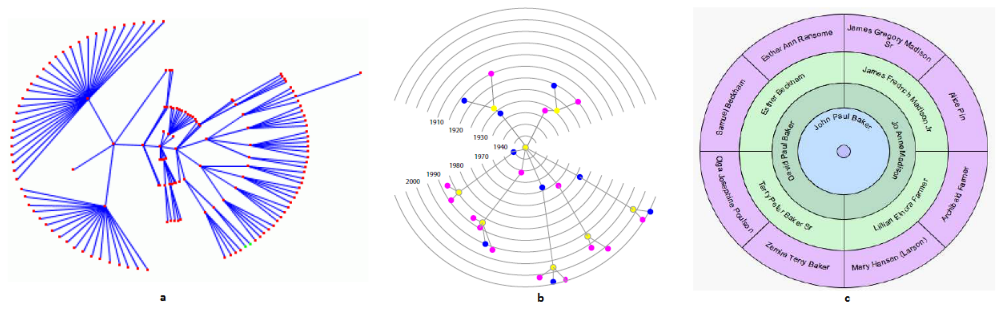


Figure 6: a: Radial tree layout [26]. b: Radial family tree layout with temporal information [22]. c: Fan Chart layout [16].

Even if the radial layout is preferable to the classical tree layout in terms of more efficient space usage, it raises some issues when used to visualize genealogical data. With the classical tree layout it was possible to incorporate the tree of descendants and the tree of ancestors of a central person by placing them beneath and above that central person respectively. This makes separation of those trees very clear, and the more the generations are separated in the time domain the further they placed from each other on the chart. In the case of the radial layout we still can follow the same strategy and divide the circle into two parts, one for ancestors and another for descendants of a central person. With every next level the density of the nodes on the border of these trees will increase and the distance between nodes of these trees will decrease. It becomes difficult to differentiate the nodes of these two different trees and eventually leads to usability issues. Keller et al. [22] solve this problem by adding some empty space as a border between the ancestors and the descendants trees of a central person (Figure 6, *b*) but again this empty space is unused space.

Keller et al. also propose a method of encoding temporal information in the radial tree layout. All the nodes are placed around the central node. The distance from the central node to any other node is the relative age difference between that node and the central node. Thus, if some node is further from the central node than another it means that that person is older than another one. That rule holds only with regard to the central node. So the distance between any other nodes that are not central does not encode the age difference. To show the absolute age difference to the central node, the central node is surrounded by circular arcs with year labels (Figure 6, *b*). Each arc is labeled with a year. The further away is the arc from the center, the larger the difference between the birth years with respect to the central person [22]. To see the age difference between other persons an interaction technique was proposed by allowing to choose the central node dynamically.

The limitations mentioned above do not allow to include all the temporal information at once. Moreover, it is not clear if only the age difference brings any value to the people using genealogical data. For example, such important information about the starting date of a disease, the date when a disease was cured, the date when a person died etc. brings a lot of value to genetics researches. In the radial tree layout persons are represented as dots on the chart, and thus no temporal information about the life of a person is visually encoded.

Another popular radial layout used for displaying genealogy data in a compact way is the *Fan chart* (Figure 6, *c*). As in the classical radial layout discussed above, in the Fan chart a central person is displayed in the center of the chart. Ancestors and descendants of the central person are drawn in the concentric rings. Nowadays many commercial genealogy software packages support the creation of fan charts [16]. There is no temporal information encoded, as well as no other additional information, except the parent-child relationship and the gender information.

2.4 H-Tree Layout

The *H-tree layout* is commonly used in binary tree drawings [31]. As ancestral data has binary tree structure, Tuttle C. et al. [31] applied that layout to the ancestral data and to build a pedigree chart visualization. They claim that the H-tree layout uses the available space effectively and that the resulting visualization turns to be aesthetically pleasing. The available space is divided evenly among nodes of the ancestors tree even if the information about some of them is missing. Relatives become placed regularly and the resulting visualization clearly shows family relationships among individuals.

The layout algorithm places the central person in the center of the tree. Parents of the central person are placed on the sides of that person and the positioning of all the other ancestors is alternating between generations. Every person in that layout is placed between own parents. With each consecutive level the layout changes the direction of placing parent nodes from horizontal to vertical and vice-versa. For example if some person's parents are placed horizontally then their parents will be placed vertically (Figure 7). By the convention proposed in that paper [31] father nodes are placed to the left or above and mother nodes to the right or beneath the current node, depending on the current placing direction. That makes the relationships more readable from the chart.

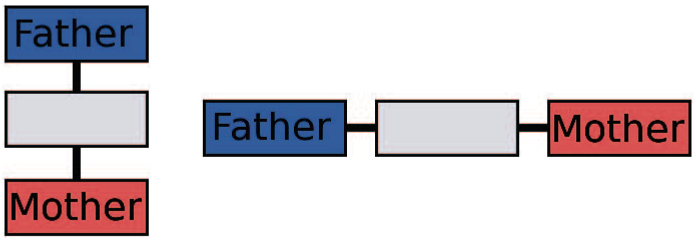


Figure 7: Parents placing direction alternating [31].

Figure 8 illustrates the algorithm of building 5 generations of the ancestors tree. From that figure we can see that even though the nodes are placed consistently we have to increase the distances between the nodes with each consecutive generation to make room for these nodes. The generations 2 and 3 are moved away from their children to give space for the generations 4 and 5 [31].

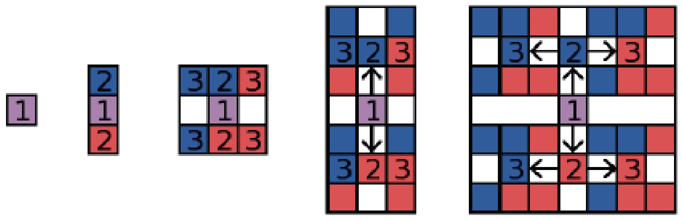


Figure 8: H-tree five generations layout algorithm [31].

The H-tree layout has one unique asset that other layouts typically miss. It allows to clearly differentiate the branches of the family. If we draw vertical and horizontal lines through the center of the square that represents the central person we would divide the chart into four squares as shown in Figure 9. Each of these squares represents the family branch of four grandparents of the central person. If we further divide any of these squares we would get the four ancestral branches of the central person of the divided square. That rule holds when applied recursively to all the subsequent squares. These squares have the following properties: they do not overlap, have the same size and each of them represents a family branch. These family branches fill the entire space of their respective squares.

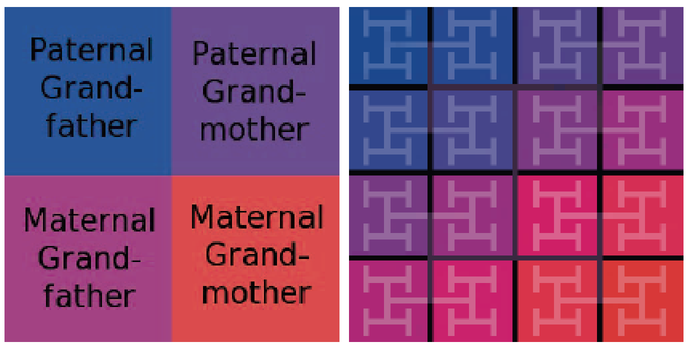


Figure 9: Family branches separation in the H-tree layout [31].

Authors of the H-tree layout also showed that it can fit more generations on the same given area than classical tree layout and fan chart preserving the ability to distinguish between individual nodes. Visual comparison of those three layouts depicting an ancestors tree of 13 generations can be seen in Figure 10.

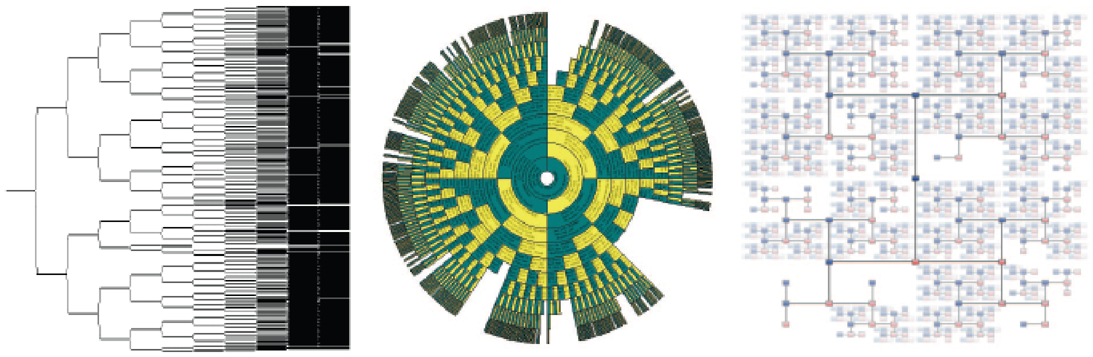


Figure 10: Visual comparison of Classical tree layout, Fan chart and H-tree [31].

2.5 Tree-map Layout

The *Tree-map layout* shown in Figure 11, *b* is a space-filling layout technique. It is applied to visualize hierarchical data by using nested rectangles and enables to use 100% of the available display space. This efficient use of space makes it a perfect technique to visualize large trees [20, 21].

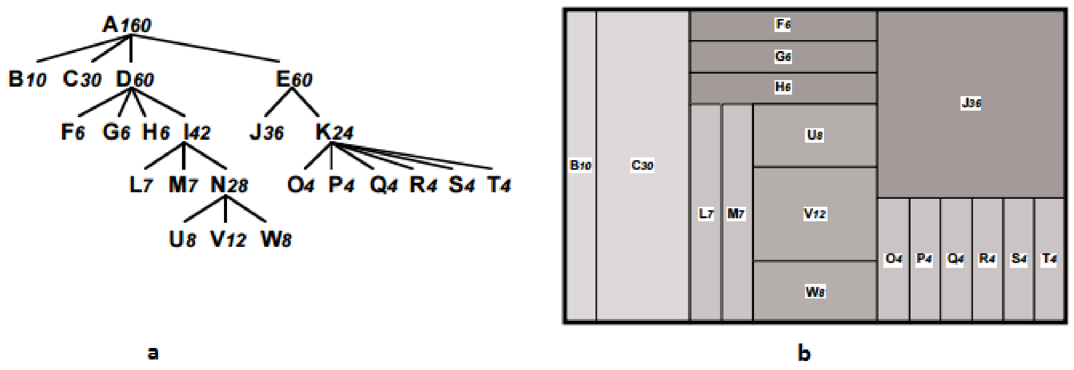


Figure 11: a: Classical tree layout. b: Tree-map layout, on the same hierarchical data [21].

By showing an entire hierarchy at once, the Tree-map layout makes it easier to navigate through large hierarchies. The user does not lose the surrounding context of a node when tracing in the structure. That problem usually happens when a zoom technique applied to the classical tree layout. Showing the whole tree structure at one place allows users to move quickly to any other place in the tree structure without losing the context and the relative position of the node in the hierarchy [21].

As was mentioned earlier the hierarchical datasets contain two types of information: the structure of the data and the content. The Tree-map layout visualizes the structural information by representing nodes as rectangles and enclosing childrens rectangles into parents rectangles. In that way the rectangle of the root element of the tree takes 100% of the available space and all its descendants are enclosed into it. As a result we have the display space divided into enclosed rectangles that represent the tree hierarchy.

The content of the hierarchical data that needs to be visually encoded determines the weight of the node. This weight is required by the Tree-map layout building algorithm to be assigned to each node of the tree. Then this weight determines the display size of the rectangle that represents the node. According to Johnson and Shneiderman [21], the designers of the Treemap layout, the following properties should hold:

- If Node 1 is an ancestor of Node 2, then the bounding box of Node 1 completely encloses, or is equal to, the bounding box of Node 2.
- The bounding boxes of two nodes intersect if one node is an ancestor of the other.
- Nodes occupy a display area strictly proportional to their weight.
- The weight of a node is greater than or equal to the sum of the weights of its children.

As an example of the Tree layout the designers of the layout show how to visualize a hard drive file system hierarchy (Figure 11). The weight of each node is the size of the file or folder represented by that node. Figure 11, *a* is a classical tree layout representing the file system structure. Every node in that tree has a label and a weight that is the size of that file or folder. The Tree-map layout shown in Figure 11, *b* visualizes the same data. We can see that it clearly shows only the content of the leaf nodes, and it is difficult to extract the content from all the other internal nodes.

Little information could be found on how to apply the Tree-map layout to genealogy data. It is very likely because of the mentioned above limitation. Another reason might be that it in case of genealogy data it is not clear how to assign weights to the nodes of the tree that determine the sizes of the rectangles.

2.6 Indented Outline Layout

Another widely used layout to visualize hierarchical data is the *indented outline* shown in Figure 12. From the first look it might appear as just another node-link layout for a tree visualization. However, it can encode both the structural and the content information of the tree without any edges drawn [26]. Even if one chooses to draw edges there are still no edge crossings. That characteristic of the indented outline style makes it a popular and useful technique. As an example of that layout we can see how Windows explorer shows the file system hierarchy. The basic rule of the layout is to place children beneath their parent and to indent them one level to the right with respect to the parent. Since every node is placed on its own line there is enough space to include extensive content information.

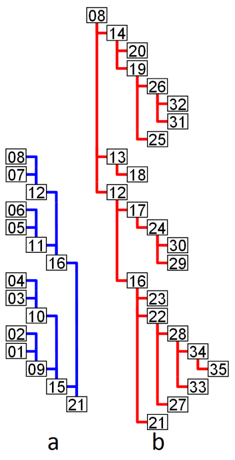


Figure 12: Indented outline style [26]. a: Ancestors tree. b: Descendants tree.

Intended outline style applied to a genealogical data results in the layout shown in Figure 12. In a case of ancestors tree typically every node has two parents. Those parent nodes are placed on the same level and have the same horizontal position (Figure 12, *a*). As can be noticed from this layout, every node has its own line and thus the number of lines needed to visualize the tree equals to the number of nodes in the tree. As the indentation can be visually perceived only a few lines at once, it becomes difficult to decide if the nodes are aligned or not in a large tree. For example it might happen that nodes that belong to the same level, e.g. sibling nodes, can be placed on lines that are far away from each other. That eventually leads to usability issues when the layout is used to visualize large trees.

Matrix-based representation is another layout that uses indentation to show the hierarchy. One of the adaptations of the matrix-based layout is the *Quilts layout* [32]. The classical tree layout and its Quilts representation are illustrated in Figure 13. Nodes are depicted as small rectangles and all the nodes belonging to the same level of the tree are aligned. Nodes on the odd levels are placed horizontally and those on the even levels vertically. Links between nodes are depicted as dots on the intersection of columns and rows on which they are placed. To make the nodes and the levels more distinguishable from each other, the nodes are colored and a unique chroma assigned to each level.

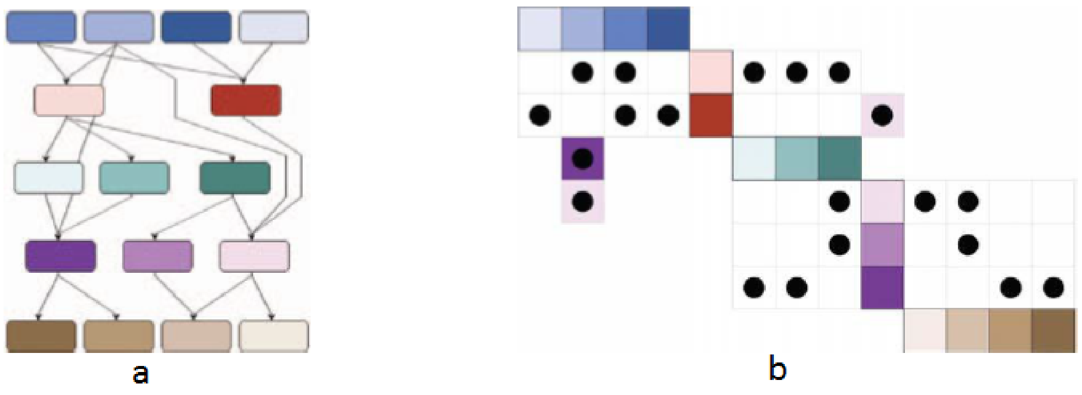


Figure 13: a: Classical tree layout. b: Quilts layout on the same data [32].

Still a problem arises when non-successive levels are linked together, that type of links are called skip links [32]. To overcome that problem, skip links are placed at the end of the matrix row or column that follows the level nodes. Those links are depicted as colored cells with dots. As a color of each node is unique the cell of the skip node is colored by the color of its destination level node. In Figure 13, *b* two colored cells have been added to the first level nodes to show the connections of these nodes to the 3rd and 4th level nodes. Even if it makes possible to encode linking between non-successive levels the solution proposed is not the optimal one. It becomes hard to find the destination node by matching the color of the skip link cell to the destination cell, especially in a large tree.

Anastasia Bezerianos et al. [10] adapted the Quilts layout to use it with genealogical data and called that technique as the *GeneQuilts*. It places individuals in the horizontal layer and nuclear families in the vertical layer. There cannot be any links between these two layers as they are of different types. Since links between these layers are not possible then there is no problem in placing skip links which was the limitation of the original Quilts layout. Those links between the two non-successive levels are placed the same way as links between successive layers.

A simple example of the GeneQuilts is shown in Figure 14. There are three lists of people representing three generations. The oldest generation is at the top and the youngest at the bottom. On the horizontal layer icons F are shown and represent three nuclear

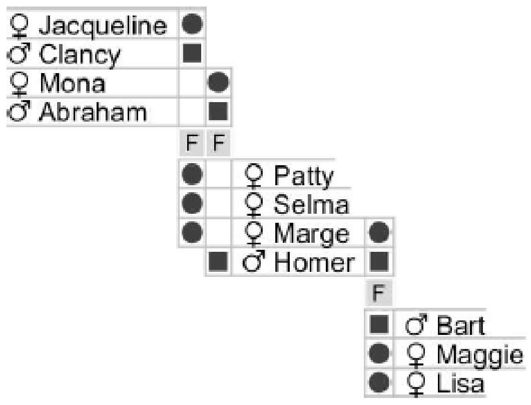


Figure 14: GeneQuilts layout [10].

families. Black dots placed above the F signs indicate parents and below the signs indicate children. Shape of the dot depends on the gender: circle point to female and square point to male. From Figure 14 we can see that couple Jacqueline and Clancy has three daughters: Petty, Selma and Marge. In the GeneQuilts layout individuals are represented as text labels with their names. Thus there is no support for encoding temporal information except outputting it in textual form as well. Another limitation is the scalability of this layout. When the generation consist of a few hundred individuals it is not possible to see the other generations in one screen because the readability of name labels should be maintained.

2.7 TimeNets Layout

Genealogy as a study of a family history is not restricted to the history of the family in general but also includes the history of every individual in the family. Genealogical data often includes temporal information, e.g., dates of birth, death, disease, marriage, the birth of children etc. Sometimes the temporal information is used to determine genealogical relations. Researchers may try to compare the birth date of an individual with the dates of marriage of its potential parents [23]. Apart from being valuable information for the genealogy researchers as well as genealogy hobbyists, that temporal information represents a special interest for genetics researchers.

The traditional genealogy data visualization layouts show the hierarchy of a family. The main focus of those layouts is to show the relations between generations [23] by aligning people by generation. They share the common set of limitations as temporal information is typically ignored, or limited to be encoded in the form of text labels.

One of the ways to visually encode temporal information in the genealogy is to depict people as lines. Lines are good at showing the story of the life of a person especially showing the temporal information. The start of the line can signify the birth date of a person whereas the end of the line represents the death of the person. The line span from the start to the end represents the life span of a person. Marriage dates, disease dates, and dates of birth of children can be visually encoded on the line as well.

That technique of depicting person as a line is not a new technique, as Joseph Priestley in 18th century used it to depict lives of two thousand famous people from 1200 B.C to 1750 A.D (Figure 15) [23]. To show the uncertainty in the birth and the death dates he put dots at the beginning or at the end of the line. The relationships of the individuals were not encoded in that chart. The horizontal axis represents time, and lines are placed horizontally according to the birth dates of the individuals. The vertical position of the line signifies the importance of the person.

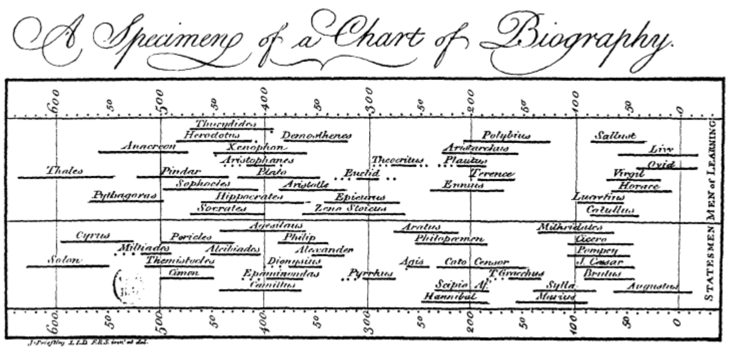


Figure 15: Biographical lifelines by Priestley [23].

To show the relationships of individuals and visually encode temporal information as well as hierarchical, Nam Wook Kim et al. [23] designed a new visualization layout for genealogical data called *TimeNets* (Figure 16). This layout places focus on the temporal information but still nicely encodes the family kinship information. To overcome the scalability problem they filter the data by calculating Degree-of-Interest (DOI) values for every individual in the family and display only those whose values are higher then the threshold value. Individuals with higher DOI values are considered to represent more interest to the user then the individuals with lower DOI values.

Similarly to Joseph Priestley's layout, the TimeNets layout represents a person as a line. The horizontal axis represents time. The horizontal position of a line is determined by

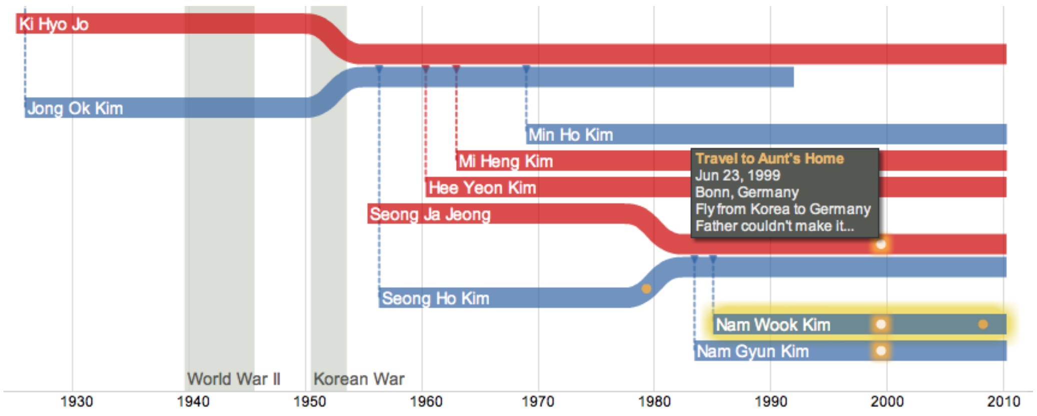


Figure 16: TimeNets layout [23].

the date of the birth as the start of the line and the date of the death of the person as the end of the line. If person is alive at the observed time interval then the line continues till the end of that interval. The color of the line encodes sex of a person, red for female and blue for male. The text labels with names are placed above or within the lines if they are thick enough.

As the horizontal axis is already used to represent time, they used the vertical axis to encode the relationships. The vertical distance between the lines encodes the conjugal relationships. To indicate the marriage the lines converge at the point representing marriage date and diverge to indicate divorce. The sequence of convergence and divergence of lifelines allows depicting multiple marriages and divorces. If a person has multiple marriages, by default authors of the layout order the spouses of the person vertically by the marriage date. That results in a chronological order of the spouses when scanning the visualization from top to bottom. The line crossings are not avoidable when using that technique but the authors proposed a different approach as well. It places spouses alternatively above and below the focal person. That approach reduces the line crossings but it breaks the chronological order of marriages of the focal person. To indicate the divorce the lines are returned to their original position. When divorce and subsequent remarriage are in the close proximity the time difference between them is increased artificially a bit to enable better perception of the crossings.

Authors of the TimeNets layout tried different approaches to encode parent-child relationships. In the first prototype the child's line started directly from the parents' marriage line and then diverged to its own space. The usability testing showed that it was confusing and difficult to differentiate between children and spouses of the focal person. Moreover that layout becomes visually noisy when lifelines have very long vertical stretches. As this approach failed to show good usability test results they proposed a different strategy on how to visually encode parent-child relations. To depict these relations they use a drop-down line from the parents' lines to the childs line. Those additional drop-down lines add some complexity to the image, and to smooth that effect and to make them less destructive they are drawn as faded dashed lines. The start of the drop-down line is annotated with an arrow head to show the directionality of the line. That also can be used to show the genes flow from parents to children. Another advantage of that technique upon the initial one is that it nicely encodes the situation when children born out of wedlock. It just simply connects parents lifelines with the drop-down line and places arrow heads to each parents lifeline. To minimize the line crossings children are sorted vertically by the birth dates. Lifelines of younger children are placed closer to the parents' lifelines.

The big challenge that one has to face when using TimeNets layout is that genealogy data often misses some temporal information. In a case of standard layout, where representation is not dependent on the temporal information, the visualization will not be affected from that missing data. However, in the TimeNets layout the structure of the tree, position and form of the lines are mostly determined by the dates of the birth, death, marriage etc. If the genealogy data of a person is missing one of these dates it becomes impossible to visualize that person. One of the solutions is to estimate missing dates. To avoid an inaccurate conclusions the uncertainty has to be visually indicated as well. So in the TimeNets layout the missing data is first estimated and then visually encoded. To show that the date of birth or death is an estimated value the start or the end of the lifeline is faded and becomes fully saturated at the estimated date.

To estimate the missing data the authors of the layout propose a few methods. One is to use machine learning methods and another is to apply a rule-based method. The birth of a person can be estimated as parents' marriage date, or as a mean value of siblings' or spouses' birth dates. The death date can be estimated the same way the as well. The marriage date can be estimated as a birth date of the oldest child. If none of these rules can be applied then the marriage date is estimated as 20 years offset from the date of birth. The date of death can be set as 100 years offset from the date of birth.

As the number of people in a family grows it is not possible anymore to display all of them on one screen. To address that issue the authors applied a Degree of Interest (DOI) function. That function assigns a number to each node in the tree that represents “the estimated relative interest of the user” towards those nodes [18]. The nodes whose DOI value is under the visibility threshold are suppressed. The DOI function works as follows. The highest DOI value is assigned to the focal person which can be, e.g., the person from the search result. Then the tree is traversed and the DOI values are decreased with increasing distance from the focal person. Across consanguine relations DOI values decrease linearly, but across conjugal relations they decrease more slowly. Thus the DOI values of the spouses of the focal person will be higher than DOI values of children and parents. Other DOI functions can be applied as well, depending on the particular use case of the layout.

Once DOI values are assigned to each node in the tree, the layout computation can be started. First it traverses the genealogical data and builds a graph of local blocks. Local blocks consist of conjugally-related people. Due to the fact that in real families intermarriages are possible, the blocks may have more than one parent. The algorithm assumes that cases of cross-generational incest do not occur, as is reasonable in most of the real-world scenarios.

Then the nodes with the DOI higher than the visibility threshold are rendered inside the blocks as lines. The block is placed horizontally along the time dimension. And the lines are drawn relatively to the origin of the block. The person with the highest DOI in the block becomes the focal person. The lifeline of the focal person is drawn first and oscillates between married and non-married positions. After that the lines of spouses of the focal person are drawn. They are placed above the focal line, and converge to and diverge from it.

Once all the blocks are built, the algorithm positions each block on its place. The horizontal position of a block is determined by the minimal birth date in the block. There are two different schemes used on how to place the blocks of ancestors and the blocks of descendants of the central person. To position the blocks of descendants a pre-order traversal of the tree is performed (Figure 17, *a*). Within each generation the nodes are visited from the youngest to the oldest child. The vertical position of the blocks depends on the visit order and ensures that the blocks do not intersect. Now to locate the position of ancestor blocks we traverse the tree using in-order traversals (Figure 17, *b*). When all the blocks are positioned the algorithm ensures that the vertical size of the chart fits on screen. If not, then the nodes with lowest DOI are discarded until the visualization fits on the screen.

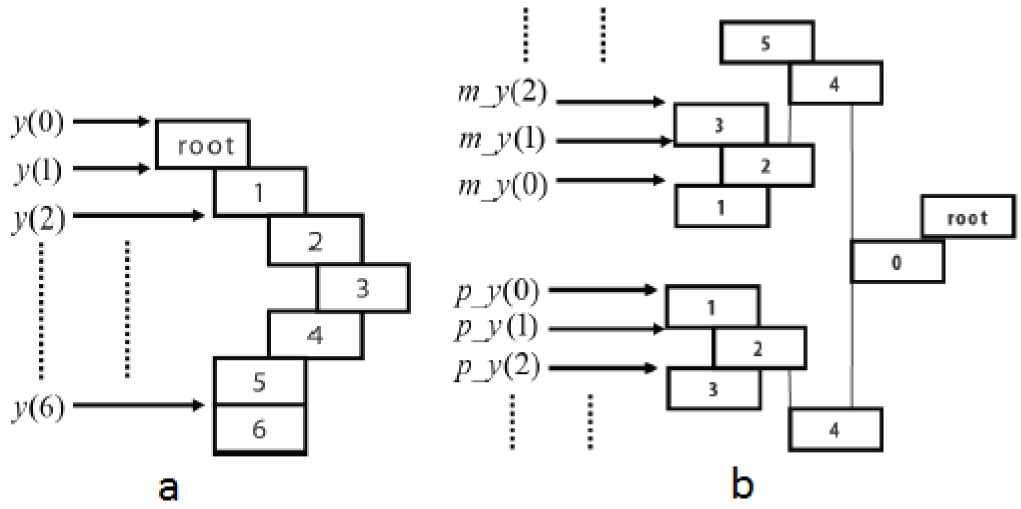


Figure 17: TimeNets layout blocks placing algorithm. a: Positioning descendants blocks. b: Positioning ancestors blocks. [23]

To enable navigation through the genealogical data, the TimeNets design implements a user interaction technique. Clicking on the line makes it the focal line and the layout is updated regarding the selected person. To support the state transition between layout updates a staged animations is used. First the elements that do not persist in the new view are faded out. Then the elements that persist in the new view moved to their new positions, and finally the new elements appear one the screen. The drawback of this method is that it still does not show all the family relations. There is no way to show the cousins, uncles, aunts etc. of the focal person.

There is another existing layout that visualizes the temporal information. The *Genelines Timeline* Software package[2] (Figure 18) represents persons as lines as well. Unlike the TimeNets layout it does not encode any other temporal information except the dates of birth and death. It supports rendering only the tree of ancestors of the focal person. It also differs in the way how the ancestors tree is laid out. The child's lifeline is always placed between parents' lifelines. Colors do not encode gender but used to differentiate father's and mother's ancestral branches of the focal person. All the ancestors from the mother's side of the focal person are colored in red and from the father's side in blue. Father's lifeline is always placed above the child's lifeline and mother's below. Dashed lines from parents to children indicate the parent-child relationships.

The drawbacks of the layout used by Genelines Timeline Software package are that it

[2]http://progenygenealogy.com/products/timeline-charts.aspx

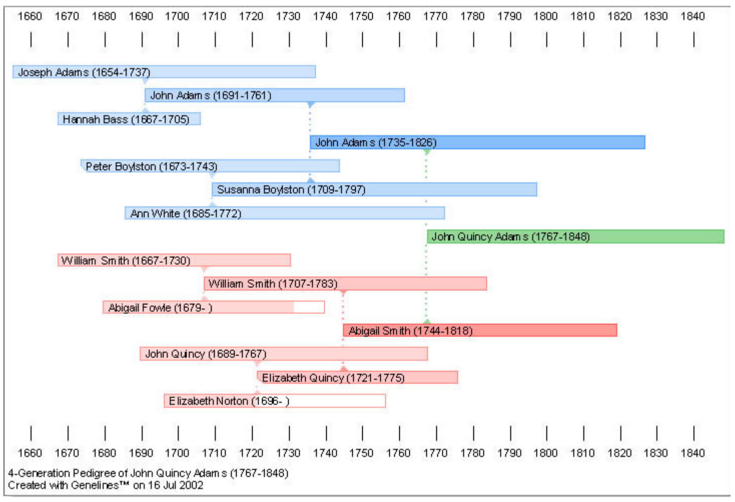


Figure 18: Genelines layout [4].

does not provide any kind of interaction with the user, and the absence of the tree of descendants of the focal person. The scalability problem is unsolved as well. When the tree of ancestors becomes big enough, the parents of the focal person are placed very far from each other due to the nature of the layout. That leads to difficulties in the information perception and eventually to usability issues.

2.8 Genograms

The *Genogram* is another visualization of genealogical data that is widely used in the medical field (Figure 19). Genograms were first designed and applied in the medical field by Monica McGoldrick and Randy Gerson. Nowadays they are used not only in the medicine field but “in a variety of fields such as psychology, social work, genealogy, genetic research, and education” [5].

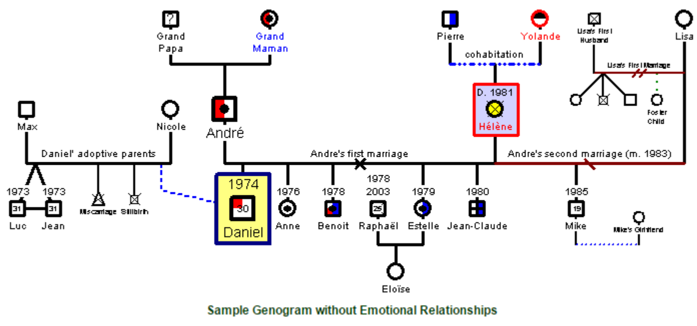


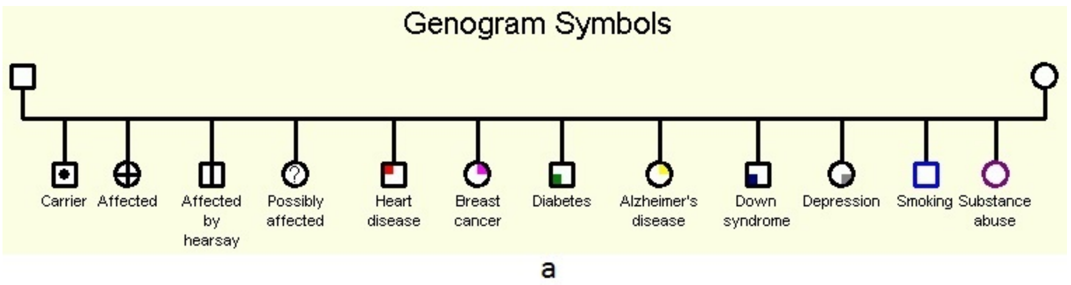
Figure 19: Sample genogram [5].

The information that genograms typically contain goes beyond the basic data found in the genealogy records. It combines both the family's health history and genetic relationships. By visually encoding disease information, medical genograms help in determining patterns of disease within a family.

To visually encode disease information various shaped, colored symbols and lines are used. The use of genograms is growing, and the genograms and the symbols are expanded and modified. Different software packages use their own conventions on how to visually encode information with different symbols and colors. Moreover, some of those software packages allow users to add custom symbols or change existing ones to reflect the current needs. Even though it adds flexibility to the tools, it is at the same time a drawback as there is no consistency in the visualization between different software packages. That requires training when changing from one tool to another. Moreover, it can lead to inaccurate or incorrect analysis results.

There are still some conventions that are followed by most of the software packages. Typically the older generations are pictured at the top of the chart and the younger generations at the bottom. The paternal branch is usually placed to the left of a person and the mother's family to the right. Persons belonging to the same generation are aligned on the same line. The order of the siblings from the left to the right is from the oldest to the youngest [30]. In genograms males are represented by squares and females by circles. The GenoPro [5] software package uses the symbols shown in Figure 20 to encode genders and disease information.

From the visualization point of view, genograms are just another modification of the standard classical tree layout that visually encodes some additional information by using various symbols. Thus it is similarly does not encode any temporal information except in the form of text labels. It has the same scalability problem when applied to the big genealogy datasets. The big number of individuals in the family will eventually lead to long edges, edge crossings and to exponential crowding. Moreover it is not strictly standardized and may lead to inaccurate analysis results, and requires additional resources to learn the symbols of each particular software package. Even though it is the most widely used tool in the medical field for visualizing genealogy data, it needs improvement to satisfy the increased needs in the field.



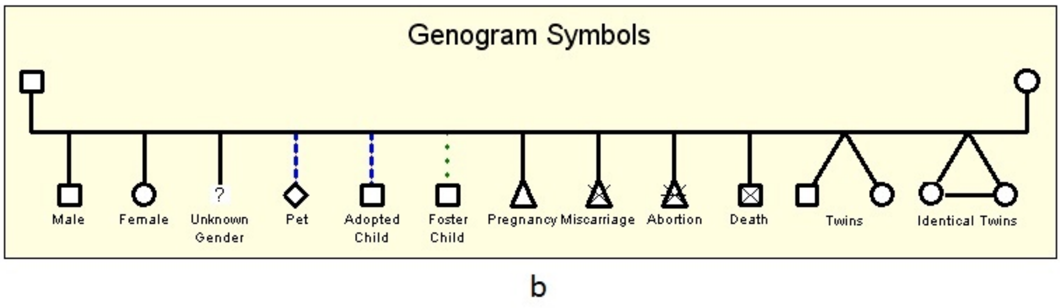


Figure 20: Genogram symbols [5]. a: Medical genogram symbols examples. b: Standard Gender Symbols for a Genogram

2.9 Focus + Context

When large trees are visualized it becomes difficult to fit everything on one screen in a human readable form. Additional navigation and interaction techniques help to overcome that problem of visualizing big datasets. The traditional techniques that are used almost in all the visualization tools are *zoom and pan* [19] techniques. Zooming is a temporal separation between views of different magnification [14] as it occurs in place. Because of this separation between views the user may think that there is discontinuity between them. To link these views, an animated transition between zoom levels is often used. According to Cockburn et al. [14] an optimal transition speed that maintains the relationship between zoom levels without disturbing the users overall interaction is between 0.3 and 1.0 second. The more exact value depends on the use case and the distance between the views. Applying a zoom technique to graphs or trees usually does not require very complex implementation as it involves only redrawing simple geometric forms and lines.

There are two different zoom techniques: geometric zooming and semantic zooming [19]. Geometric zooming just magnifies the view, that is, objects do not change their shape but just the size [17]. Semantic zooming on the contrary allows changing the amount of information displayed depending on the available space for the object.

For example map applications often use that technique. As the user zooms in to some area the more detailed information about this area appears on the screen. A proper clustering method is needed when applying semantic zooming to the tree visualization. At every zoom level it needs to be decided how many nodes, which nodes and to which degree to visualize them.

There are some weak places of zoom and pan technique when applied to tree visualization. One of them is the difficulty of the tree traversal. Consider the following example: The user zoomed in deeply into the tree structure and now wants to move to some completely different branch. If zoom level is on the depth when that other branch is not visible on the screen then it is difficult to determine where to move the view. Even if the user knows in which direction to move the view, the transition may require a lot of actions from the user. Increasing the panning speed has limitation as well because it should be kept within limits of human perception. To go to another view the user might need to zoom out first to the level where the desired branch is visible and zoom in again to that branch. Even if it is an adequate solution it still requires the additional actions from the user.

Zooming enables focusing on a particular thing, but it also may bring usability issues as the user looses the possibility to see the whole picture at once. The user might get disoriented when zoomed in as the surrounding context information is not visible.

The *focus + context* technique is widely used to complement and improve the zoom and pan technique. Some implementations of the focus + context are used as standalone techniques without using zoom and pan technique. Focus + context allow the user to focus on some particular thing and at the same time preserve the context information. One of the most well-known focus + context techniques is called *fisheye distortion*. Fisheye distortion uses the effect of a fisheye lens by amplifying the area that represents the interest and showing all the surrounding information with successively less details [19].

Herman et al. [19] describes how to apply the fisheye distortion to a graph. Typically the focus point is defined by the use,r e.g., the mouse position on the graph. Then a *distortion function* is applied to the distances from every node in the graph to the focus point. Then newly calculated distorted points are displayed and connecting edges between them are drawn. The distortion function monotonically maps a [0,1] interval to [0,1] (Figure 21). The function grows faster around 0, meaning that distance grows faster in the close proximity to the focus point. That gives more space to the nodes close to the focus point. The growth slows down towards the end of the interval meaning that nodes would be more packed in the areas that are far from the focus point.

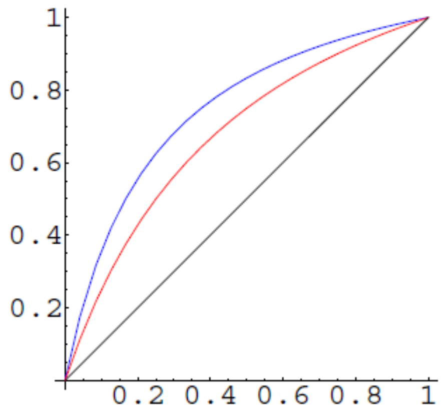


Figure 21: The SarkarBrown distortion function. Red curve is factor 2. Blue curve is factor 4. [19]

The distortion function shown on Figure 21 has the following form:

$$h(x) = (d + 1)/(d + 1/x)$$

The d parameter is the distortion factor which regulates the strength of the distortion. It should be positive; the bigger it is the stronger is the distortion. It is not the only distortion function that can be used, and other variations are possible as well.

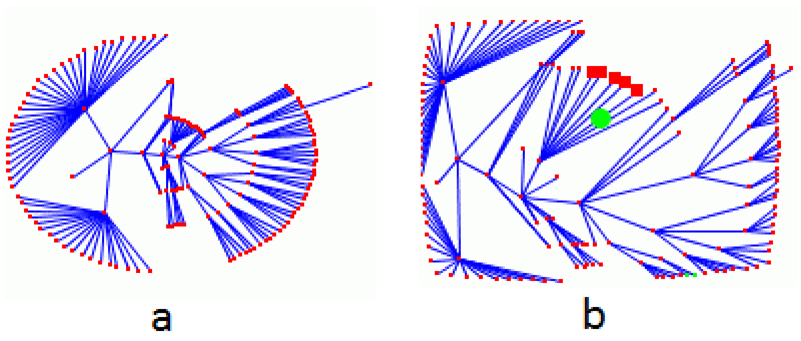


Figure 22: a: Radial view graph. b: Fisheye distortion applied to radial view graph. [19]

Even if the fisheye distortion is a good complement to zoom and pan technique, Herman et al. [19] point one complication of applying it to graphs. When nodes are distorted the edges are distorted as well and the result of edge distortion is a curve. Existing software visualization tools do not provide with the option to transform those lines into the curves. The solution could be to approximate the curve with a polyline. To make the polyline smooth, many approximating points are required. The technique is usually applied to the real time visualization and the calculation of these points will increase the response time of the system. These calculations can be avoided if the edge distortion is not applied at all and edges just connect distorted nodes with straight lines. It will eventually lead to the edge crossings as shown in Figure 22.

2.10 Dual-tree

As discussed earlier, the visualization of large genealogical datasets is a challenging task. Moreover, the visualization of genealogical graphs has different requirements than visualization of general graphs, e.g., all the individuals in a generation should be aligned. It seems that not much can be found in the literature about the graph theoretic properties of genealogical graphs [26]. When it comes to the visualization of genealogical data it often boils down to the visualization of the tree of ancestors or the tree of descendants of some individual. Sometimes these trees are combined together to form an *hourglass chart*. Those hourglass charts show only some part of the family, the descendants and ancestors of the central person. All the other relatives such as cousins, uncles, aunts etc. are discarded. Moreover, every individual of the ancestors tree has its own tree of descendants as well as every individual of the descendants tree has its own tree of ancestors. That is when it becomes challenging to visualize all these trees at once. McGuffin and Balakrishnan made an analysis of genealogical graphs [26] and identified the difficulties of drawing those graphs. They also proposed a new visualization approach called the *Dual-tree*, and made a prototype software to evaluate the proposed approach. Dual-tree was designed to use with genealogical graphs but it can also be applicable in other domains.

There are existing algorithms to draw trees without edge crossings. However, drawing genealogical trees often requires to order generations by time. According to McGuffin's and Balakrishnan's analysis results [26], such ordering makes it impossible to avoid edge crossings. We can relax the requirement by forcing such ordering only locally so that it applies only to parent-child generations order. That will allow to draw the graph without edge crossings but still long edges are unavoidable (Figure 23).

To understand other problems which arise when visualizing genealogical graphs the

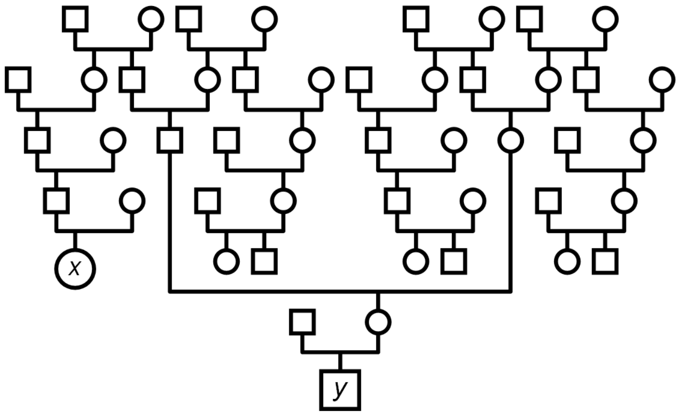


Figure 23: Long edge cannot be avoided even if only local generations ordering allowed. x and y are the individuals belonging to the same generations but it is not clear from that depiction [26].

authors of the Dual-tree layout analyzed the visualization of an idealized genealogical graph. The idealized genealogical graph that they used has the following properties: every person has two parents, one spouse of the opposite gender, two children of opposite genders and one sibling of the opposite gender. The requirement for visualization of such graph is to align nodes belonging to the same generation. The visualization of the graph is shown in Figure 24. It includes only 9 nuclear families and only 4 generations. If instead of ellipses we continue to draw more nuclear families, the edge crossings become unavoidable. Moreover, it will require to make more room for the new families and eventually will lead to long edges between siblings and spouses.

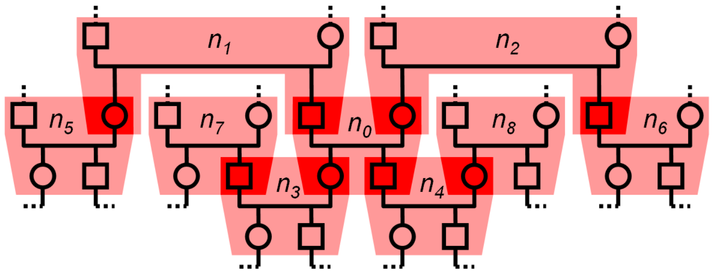


Figure 24: Visualization of an idealized genealogical graph [26].

If we look at the row where n_0 is located in Figure 24 we can notice that we have to fit there all the siblings and cousins of that node. As we see that number of the nodes in the row grows exponentially whereas the available space "grows linearly with the geometric distance from the center of the diagram". That shows another problem of the genealogy graph visualization called *exponential crowding* [26].

The exponential crowding problem motivates the analysis of alternative layouts that might solve this problem. In the previous example the allocated space for every node was equal. As we noted the number of nodes increases exponentially when the space grows only linearly. However, "by allocating progressively smaller areas to nodes" we might be able to visually encode more information into the same available space [26]. Figure 25 shows the *fractal layout* applied to the same idealized genealogical graph. As can be noticed the nodes of the graph take less and less space when moving away from the center of the figure. The depiction is similar to the fisheye and radial view layouts as the central part has more available space and is presented in more details. Following the same logic we can enable the user to dynamically choose the focal region and traverse the structure. We can continue to extend the graph applying that technique, but the size of the nodes becomes unacceptably small to retrieve information from them. Moreover there is no clear separation between the generations anymore.

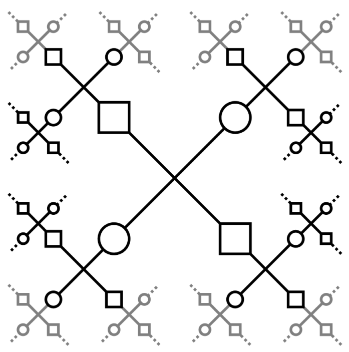


Figure 25: A fractal layout for idealized genealogical graph [26].

Further analysis of other genealogy visualization techniques such as H-tree, Intended Outline etc. showed that none of them actually solve the problems like exponential crowding, edge crossings and long edges. The results of the analysis motivated McGuffin and Balakrishnan to build a genealogy visualization tool that will show only a subset of the tree at a time. The next questions that arose were how to choose the best subset to show and how to allow the user to change the subset when needed.

In the case of genealogical data, there are two subsets of the the data that are crucial. They are the tree of ancestors and the tree of descendants. We already know that a combination of these two trees for a person results in the hourglass chart (Figure 26, *a*). To encode more relations of a person including not only direct ancestors and descendants but cousins, uncles, aunts etc. the authors of the Dual-tree layout proposed to offset these trees with respect to each other (Figure 26, *b*).

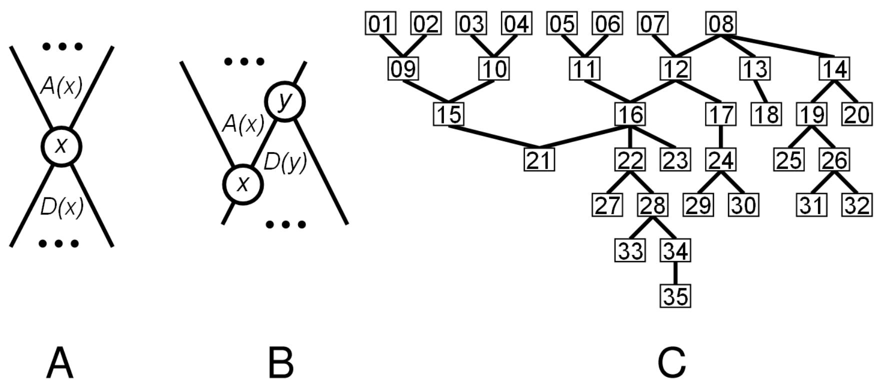


Figure 26: A: Hourglass chart. B: Dual-tree scheme. C: Dual-tree example [26].

As can be noticed from Figure 26, *b*, the tree of descendants of the person x is a subset of the tree of descendants of the person y. Similarly the tree of ancestors of the person y is a subset of the tree of ancestors of the person x. By adjusting the offset between the tree of ancestors and the tree of descendants we can choose the number of individuals to be visualized. If x is chosen in the older generations it reveals more descendants but fewer ancestors comparing to the situation if it was chosen from the younger generations. The similar thing happens if we choose y from the older generations that results in showing more descendants and fewer ancestors comparing to choosing y from the younger generations. By offsetting x from the most recent generation and y from the oldest generation we can maximize the coverage of the subset.

The crowding of the Dual-tree technique is the same as is the crowding of the individual trees. To avoid edge crossings and long edges they also proposed a specific technique on how to combine the tree of ancestors and the tree of descendants. The root of the tree of descendants has to be the right-most node of the tree of ancestors. Correspondingly, the root of the tree of ancestors has to be the left-most node of the tree of descendants. If another root node for the tree of ancestors or the tree of descendants needed to be selected then a subtree rotation is required. When a rotation is applied, the new root node of a tree of ancestors becomes the right most node and the new root of the tree of descendants becomes the left-most. At the same time the previous trees have to be collapsed.

The node-link representation is not the only style that can be applied to Dual-trees. Figure 27 shows how the intended outline style, which was discussed in Subsection 2.6, can be applied to the Dual-tree. This style is preferable when some information is encoded in the form of text labels. There is always space from the right or from the left sides of a node where these labels can be placed. To create that tree, first two separate trees of ancestors and descendants are built using intended outline style (Figure 27, *a*). Next the nodes that belong to both of the trees are identified (Figure 27, B). Finally these trees are merged by connecting the common nodes (Figure 27, *c*).

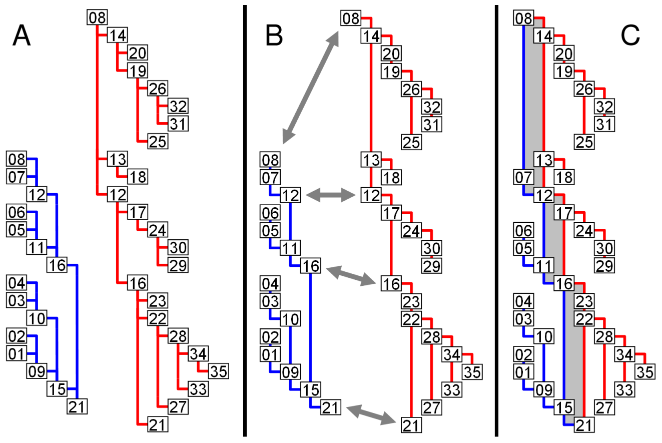


Figure 27: Intended outline layout applied to Dual Tree [26].

To understand how the Dual-tree concept will behave in real world scenarios the authors of the technique built a software prototype. The prototype can show a subset of the big tree and allows to traverse the whole tree by interactively choosing different subsets of the tree. The transition is enabled by providing the user with the possibility of choosing the roots of the tree of ancestors and the tree of descendants. Whenever a new root for the ancestors tree is chosen the the old tree collapses. The tree of ancestors rotates in the way that the newly chosen ancestor node becomes the right-most node and a new tree of descendants of this ancestor is built. To ensure that the user keeps track of the changes made, all the rotations, collapses and rebuilds are done with animated transition from one state to another.

The Dual-tree supports both layouts: the classical node-link and the indented outline styles. The orientation of the classical node-link style can be chosen from left to right or from top to bottom. The orientation can be chosen depending on how much space is available horizontally and vertically. The area taken by the intended outline tends to be smaller than the area taken by the the other two classical node-link styles. It usually does not take much space horizontally. That makes it easier to navigate the tree as the scrolling in only one direction is required in a zoomed-in view.

The Dual-tree technique gives a possibility to visualize big genealogical datasets by showing a subset at a time and allowing to change that subset dynamically. That eventually gives the possibility to explore the whole tree. The proposed layout meets the requirements of most of the standard genealogy visualization tasks. Moreover, the usability study showed that users do not have problems with using the prototype after initial familiarization with the tool. On the other hand there was no solution proposed to encode temporal information to the Dual-tree except in the form of text labels. The possibility of designing other layout styles to the Dual-tree to visually encode temporal information needs to be explored. In the Section 3 we describe our proposed solution to this just mentioned problem.

2.11 3D Visualization

All the genealogy visualization layouts discussed above are two-dimensional (2D) layouts. They all are limited when dealing with big genealogy datasets. The number of nodes grows exponentially from one generation to the next and the visualization is running out of available space very quickly. This brings up the question: "If there is not enough available space in traditional 2D visualization layouts to visualize big genealogy datasets, why not to move it to the three-dimensional space?". It seems that not much can be found from the literature about 3D genealogy data visualization. There are also few software packages that can visualize genealogy data in 3D space.

Angeline Loh et al. developed a prototype software package called Celestial3D[3] that utilizes 3D drawing algorithm for genealogy datasets [25]. They highlight the following advantages of the 3D visualization over 2D [24]:

- A substantially greater number of individuals to be displayed in a given area;
- Closely related individuals to be displayed spatially near each other, which is not always possible in 2D, thereby allowing patterns of phenotypes, common genes or exposures in families to be more clearly seen;
- Improved focus and context techniques, that allow the display of detailed information on individuals in the context of the entire family.

[3]http://www.gohad.uwa.edu.au/software/celestial3d

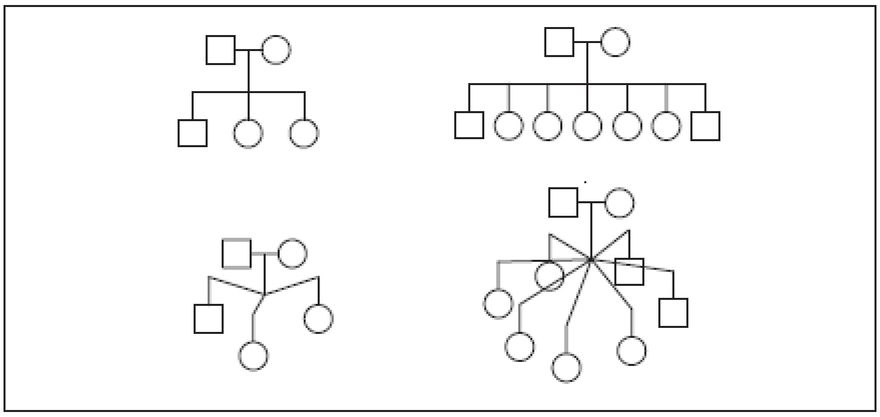


Figure 28: Celestial3D underlying principle [25].

The underlying principle of constructing a 3D tree from a 2D tree is shown in Figure 28. As can be seen from the bottom row of that figure, children are placed below their parent and are radiating outwards. That makes it possible to draw more individuals on the same given area. Another characteristic of that layout that the authors are highlighting is that closely related individuals are placed close to each other. That makes trends within families more visible.

An example of the Celestial3D visualization is shown in Figure 29. The families are clearly separated and the navigation tools allow users to explore the tree. It is possible to see the families and individuals in greater details by navigating the view closer to the interested area. The individuals are distinguished by the ID number displayed below the icon. The icons are chosen so that they are analogues to the 2D representation; such as, females are represented as spheres and males as cubes.

Even if the Celestial3D genealogy data representation looks promising and has important advantages over 2D representations it still has some problems that are not solved yet. In the prototype version of the Celestial3D, pedigrees with consanguinity or multiple marriages for one person cannot be built. Moreover there is still no solution on how to visually encode temporal information into that layout. To answer these question a further research is needed. In the Section 3 we describe our proposed solution on how to visually encode temporal information in a big genealogy dataset visualization.

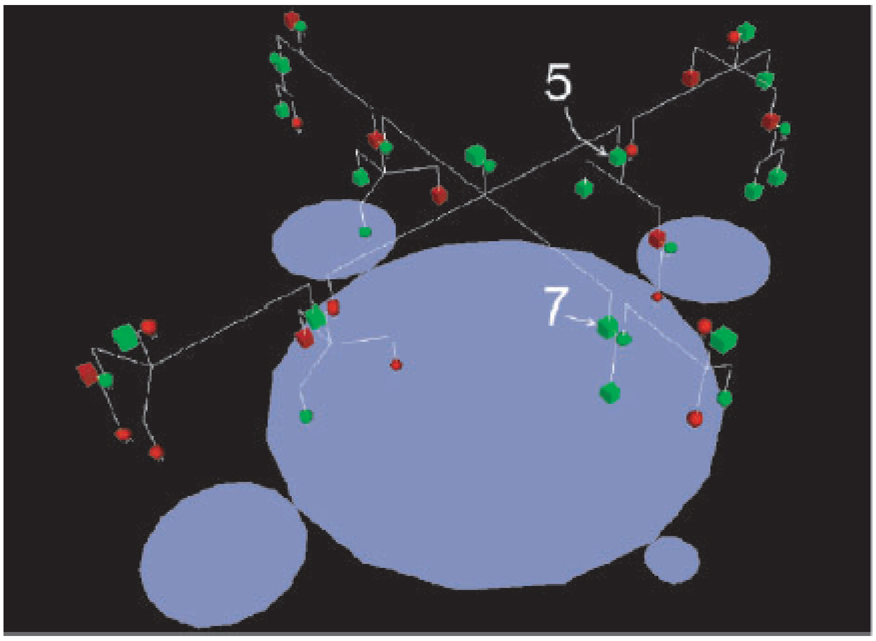


Figure 29: Celestial3D layout [25].

3 Proposed solution

One of the main problems that genealogy visualization tools face is the scalability problem. The existing genealogy data visualization tools can solve most of the problems when applied to a relatively small dataset. However, when the size of a dataset increases to a few hundred individuals, the visualization with the existing tools turns to be a difficult task. The problem arises because of the complex family relationships of hundreds of individuals have to be visually encoded on a small computer screen. One of the existing solutions is to show only a portion of a family at a time instead of showing the whole family. Now the question is how to choose the right part of the family and still get the maximum information from it. Some visualization tools show only the tree of ancestors, some can show both the tree of ancestors and the tree of descendants. Sometimes it is not enough, and information about a more extended family is required. The Dual-tree solution that was discussed earlier proposed a new way of choosing the subset of a family to visualize. The interaction technique allows exploring the whole family by dynamically changing the subset. The proposed layout allows to avoid another genealogy data visualization problem—edge crossings. The chosen subset of the family consists of the ancestors tree of a focal person and the descendants tree of one of the ancestors of the focal person. The user can dynamically choose the focal person and the ancestor whose descendants tree is displayed. By choosing the offset between these individuals, the chosen subset represents the family in greater or less details. In that way it handles the scalability problem of the genealogy data visualization.

The available genealogy information becomes more precise and includes many additional details besides family relationships. That information also includes temporal information such as dates of birth, death, disease etc. That temporal information represents a special interest for genealogy researchers. The temporal information such as "How old was a person when the disease was diagnosed?", "How long did the person live after the disease was diagnosed?" etc. gives additional value for the genealogy specialists and helps to make more precise analysis. To get full advantage of that information it needs to be visually encoded as well. Unfortunately the Dual-tree solution does not provide any way of encoding temporal information in a visual form, except in the form of text labels.

The TimeNets layout, which was discussed earlier, proposes a solution of encoding the temporal information. A natural way of ordering chronological events is to put them on a line. The line can represent a time interval, thus a line can also represent the lifespan of a person. The life events of a person can be placed on the lifeline as well. TimeNets layout uses that technique and represents every individual in a family as a line. One of the main focuses of the TimeNets layout was placed on visually encoding marriage information of individuals. However, the marriage information does not represent a special interest for genetics researchers as it is irrelevant with respect to genes flow. Moreover the way that marriage information is encoded adds a visual noise to the final visualization. The lines become curved in case of a marriage and that makes a line to occupy more space and line crossings become unavoidable.

The TimeNets design also proposes a way of solving the scalability problem. The algorithm assigns the DOI values to every individual in the family and discards those individuals whose DOI is under the visibility threshold. That solution is not very flexible and important information might get discarded.

As we can see there are genealogy data visualization techniques and tools that focus on some particular problem and solve it. However, it seems that there is no existing solution that incorporates both the solutions for handling big genealogy datasets and for visually encoding temporal information. Thus the effort of this work was put on finding a way of merging the Dual-tree characteristic of handling big datasets and encoding the temporal information of the lives of individuals in the form of lifelines.

3.1 Encoding temporal information

The visual representation of temporal data needs to ensure that “it is easy and quick to understand and does not lead to ambiguous interpretation” [13]. Shneiderman [29] defines temporal data as “data with a start time, finish time, and possible overlaps on a timescale, such as that found in medical records, project management, or video editing”. The life of a person is the interval that takes place over period of time. There are also life events that take place over periods of life or at some point of the life. Lines can be easily used to visually encode the duration of the life of a person and events that happen in the person’s life. Moreover, that approach was successfully applied to genealogy data in TimeNets and Genelines layouts.

As the representation of the temporal information of the life of a person as a line did not show any limitations we used the same approach and visualized every individual in a genealogy data as a line. The horizontal axis of our visualization represents time dimension progressing from the left to the right. Thus the lines are placed horizontally where the start of the line represents the birth date of a person and the end of the line represent his or her death. If a person is alive at a given time interval, the line continues till the end of that interval.

As was described earlier in the TimeNets layout the focus is placed on the marriage dates of individuals. To visually encode marriage information the lines converge and diverge from each other signifying the marriage and divorce. That makes lines to take more space than just straight lines and adds visual noise as line crossings become unavoidable. In our design we decided not to follow TimeNets approach but instead follow Genelines approach and depict individuals as straight lines as it is a more space-efficient representation.

The color of the line can encode different information. In the Genelines layout the focal person’s line is colored in green and does not carry any information except the fact that this is the focal person of the visualization. Other individuals are colored in red, if they belong to the maternal ancestral branch of the focal person, and blue if they belong to the paternal ancestral branch. The line color fill is empty if the date of birth or death is unknown or estimated. That color coding scheme does not actually encode any additional information as these ancestral branches are separated in Genelines layout. The paternal ancestral branch is located above the focal person and the maternal beneath. In the TimeNets layout the color of the line represent the gender of an individual. A blue line is drawn for a male and red is for female. In our design we follow the same scheme and encode the gender of a person by coloring the lifeline in red or blue colors.

Moreover, the color encoding scheme can be applied to show different life events of a person. For example the period of life when a person had some disease can be encoded as a differently colored segment of the lifeline. The start of that segment signifies the date when disease was diagnosed and the end signifies when it was cured. To signify the start and the end of an event, dots colored with the same color are placed at both ends of that segment. Different events (e.g. diseases) can be encoded using different colors. The dots at the ends of the segments also help to distinguish when different events start and end even if they overlap.

The thickness of a line can also be different. The Genelines layout uses thick lines and places the text labels of names of individuals inside these lines. The authors of the TimeNets layout use the same technique but they also note that a different approach is possible. The lifeline can be thin and the text labels can be put above them. That technique makes the lines more distinguishable and leaves more space between them when visualizing big datasets. In our layout we draw the lines with 1 pixel thickness. When the visualization becomes highly dense we draw the lines half transparent to make them more distinguishable from each other.

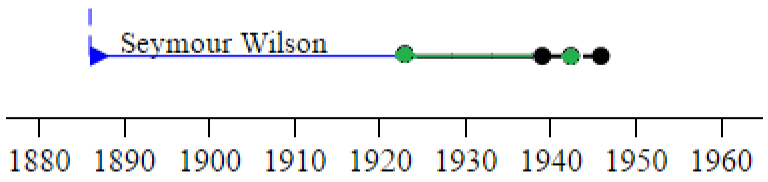


Figure 30: The lifeline representation.

The final lifeline representation used in our design is shown in Figure 30. As can be seen from that figure the person depicted is male as the color of the line is blue. Two overlapping events are represented as differently colored segments on the lifeline. The points at the ends of these intervals are colored with the same color as the intervals they represent, which helps to distinguish these intervals even when they overlap. The start of the lifeline has an arrow heading colored in the same color as the lifeline. It signifies the start of the lifeline. In the case if some life event of a person takes the whole lifespan (e.g. congenital incurable disease) the lifeline will be colored in the color of that event and the lifeline color that represents the gender will not be visible. In that case the arrow heading of the lifeline still can help to see the gender of the person. The gender is also visible in the color of the dashed line from the parent, if the parent is given.

3.2 Ancestors tree

As discussed earlier, the Dual-tree layout is a combination of two trees: the tree of ancestors of the central person and the tree of descendants of one of the ancestors of this person. The root node of the descendants tree is always placed as the right-most node of the tree of ancestors of the central person (Figure 26 on page 30). That positioning allows avoiding edge crossings. The positioning algorithm ensures that the nodes do not overlap and that the nodes are aligned by generations. However, in the case of representing individuals as lifelines the same approach cannot be applied. The reason is that the length of the lines is not equal and depends on the lifespan of every individual. The horizontal position of the lines also depends on the dates of birth of the individuals. Thus if we try to align lines by generations, we would eventually end up with a lot of overlapping lifelines. Thus to position the nodes of the Dual-tree that incorporates the temporal information in form of lifelines, a different positioning algorithm needs to be applied.

To build a tree of ancestors we used the Genelines layout algorithm[4]. The central idea of that algorithm is that the lifeline of a person is vertically placed between the parents' lifelines. The horizontal position of the lifeline of an individual is determined by the date of birth of that individual. An example of the ancestors tree of our approach is shown in Figure 31. The root node of that tree has green text label with the name of the person. The vertical dashed lines, called *birthlines*, connect the children with their parents and have the color that matches the gender of the child. The ends of these lines have arrow heads that show the direction from parents to the children. They also show the genes flow from parents to their children. As many lifelines can cross some particular birthline it becomes difficult to understand to which lifeline that birthline belongs. To highlight the corresponding lifeline we place an arrow heading at the

[4]http://progenygenealogy.com/products/timeline-charts.aspx

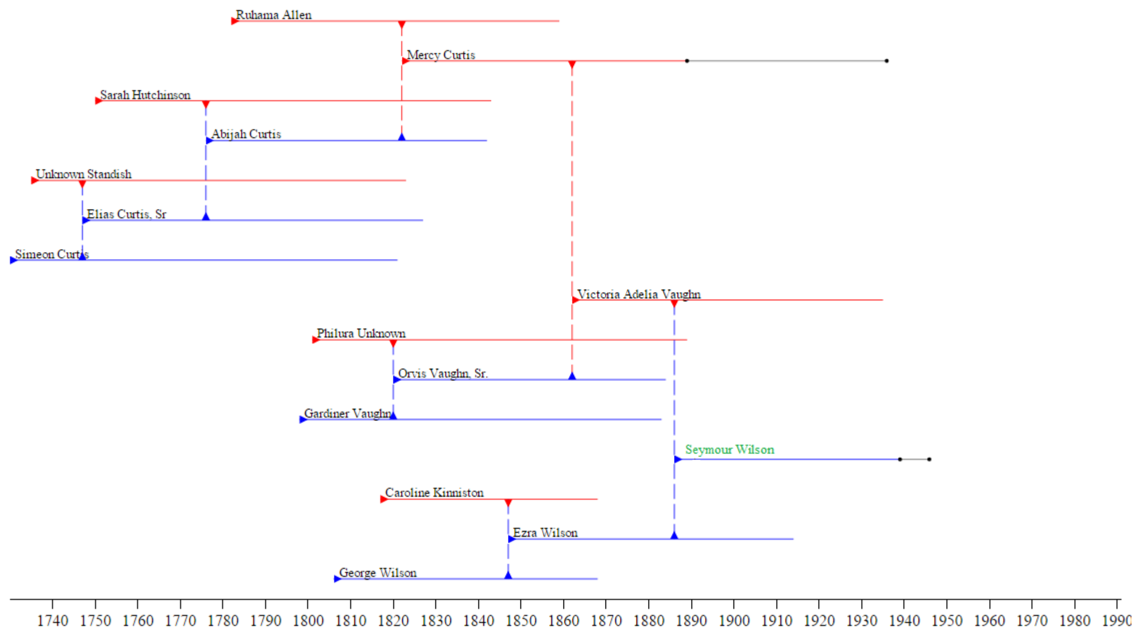


Figure 31: Example layout of the Tree of ancestors.

beginning of that lifeline. The corresponding parent is also emphasized by an arrow head which matches the color of parent's gender.

The underlying algorithm of vertical ordering is very simple and based on the tree traversal. First we build the tree of ancestors of the central person and then do an in-order traversal of that tree (Figure 32). It traverses the left subtree by recursively calling the in-order function. When the algorithm reaches a node that has no left subtree it renders that current node and then traverse the right subtree, again by recursively calling the in-order function. As a result of this traversal, left and right subtrees become above and below lines respectively. In the final visualization after that traversal the children's lifelines are always placed between their parents' lifelines.

3.3 Tree of descendants

To build a tree of descendants we used the TimeNets layout approach that was discussed in Section 2.7. Children lifelines are depicted as lifelines that emanate from their parent lifeline. We use the same dashed birthlines that connect the parent with a child. The same way as with the ancestors tree the start of that birth line is annotated with an arrow heading showing the genes flow from ancestor to descendant. Authors of the TimeNets design proposed a solution on how to avoid lifelines crossings with

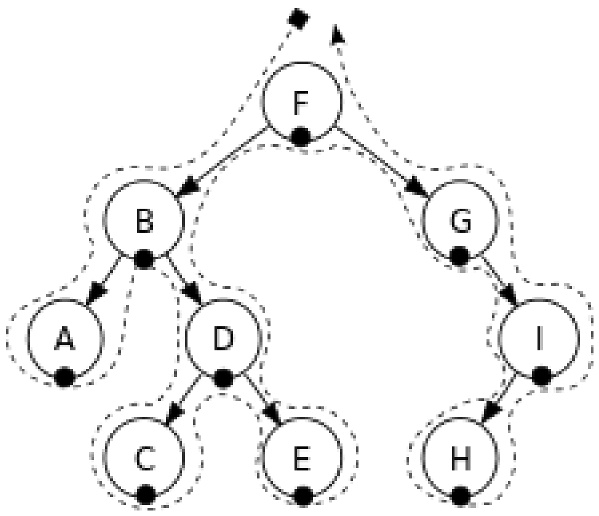


Figure 32: In-order traversal of the ancestors tree. The order of visited nodes: A, B, C, D, E, F, G, H, I. [8].

birthlines. The solution is in vertically ordering children by their date of birth. We start drawing the lifelines of the youngest child and continue to the oldest. We apply that procedure recursively to every descendant so that final tree includes all the descendants of the greatest ancestor. An example tree of descendants is shown in Figure 33.

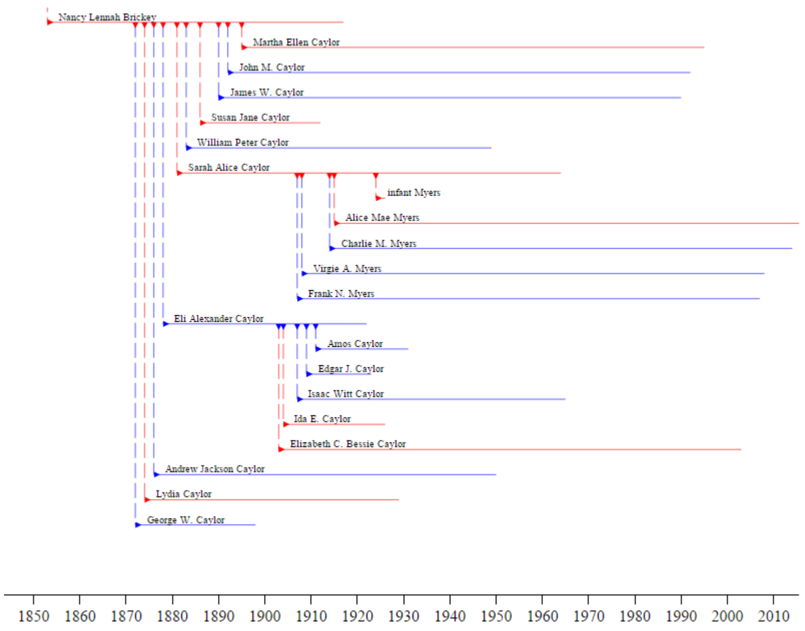


Figure 33: Example layout of the Tree of descendants.

As can be seen in Figure 33, the greatest ancestor for whom that tree of descendants is built has 10 children. They are vertically ordered in the above described way so that no edge-lifeline crossings occur. Two of the greatest ancestor's children have children on their own. The same recursive approach is applied to build their own trees of descendants. The disadvantage of that approach is that sometimes it happens that children become placed far from their parent. It is especially the case for the oldest children as all the descendants of their younger siblings are placed between him/her and the parent.

3.4 Connecting ancestors and descendants trees

Now when we already have designed the layouts for ancestors and descendants trees we need to merge them together. As we mentioned earlier, McGuffin and Balakrishnan [26] have analyzed the difficulties of visualizing big genealogy datasets using traditional tree layouts. If we look at Figure 5, *d* on page 7, we can see that every ancestor in the ancestors tree of a central person has a tree of descendants on his own. If we try to visualize all these trees of descendants we will get a very difficult to perceive visualization. Moreover, it will not be possible to avoid edge crossings.

Almost the same problem happens if we try to display all the descendants for every ancestor in our layout for the ancestors tree shown in Figure 31. We will eventually have a lot of lifelines crossings with birthlines. For example if we look at Figure 34 we can see that even for that simple case the birthline of the central person got crossed many times by lifelines of his close relatives. In the case that is shown in Figure 34 we assume that the mother and the father of the central person have only one common child which is the central person. If there were more than one common child we would have to place all of them between the parents. That would result in numerous lifeline crossings with birthlines. There will not be any use of such a visualization as it becomes very noisy with line crossings.

It becomes clear that if we cannot visualize all the family members at once we have to visualize only some subset of the data and allow the user to dynamically choose that subset. McGuffin and Balakrishnan [26] used that technique in Dual-tree layout and proposed the solution how to choose the subset and enable the user dynamically explore the whole dataset. They showed the descendants tree of the ancestor that is the right-most node of the tree (Figure 26). To enable the user to traverse the whole genealogy data, they allow to choose another ancestor. When another ancestor is chosen, they rotate the tree of ancestors to make that newly chosen ancestor the right-most

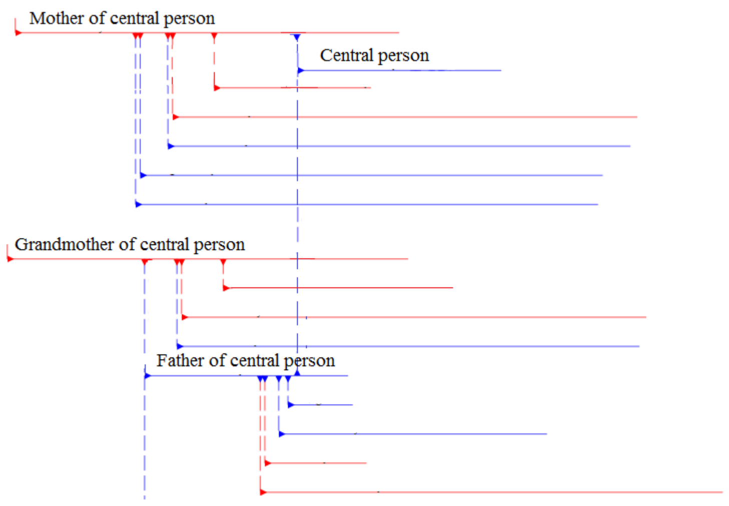


Figure 34: Example of multiple crossings of lifelines with birthlines.

node. The tree of descendants of the central person is also a subset of the tree of descendants of any of his ancestors. By the placement of the ancestor for whom the tree of descendants is built they make it possible to avoid edge crossings.

In our layout we decided to follow the same approach and show only a subset of the genealogy data at a time. To avoid lifelines crossing with the birthlines we are showing the whole descendants tree for only those ancestors for whom the path of birthlines to the focal person goes only upwards. The motivation of using that rule is that when we draw the descendants tree of such an ancestors, it is placed beneath the ancestor and the birthlines path towards the focal person is always above as the rule requires. That allows avoiding lifelines crossings with the birthlines. An example of the visualization after applying the above mentioned rule is shown in Figure 35.

In fact the rule described above results in the same subset of data chosen as for the Dual-tree technique. All the ancestors for whom the descendants trees are built are the nodes that are placed on the right-most branch of the ancestors tree of the central person. That allows us to adapt the same technique of the tree rotation if we want to display the descendants tree of some other ancestor.

We designed the flow of the control as follows. First the user selects the focal person from the list of all the individuals in the dataset. Then we build the tree of ancestors for that person. As the ancestor for whom we want to build the descendants tree is not chosen by the user yet, we by default choose the greatest paternal ancestor. Then we

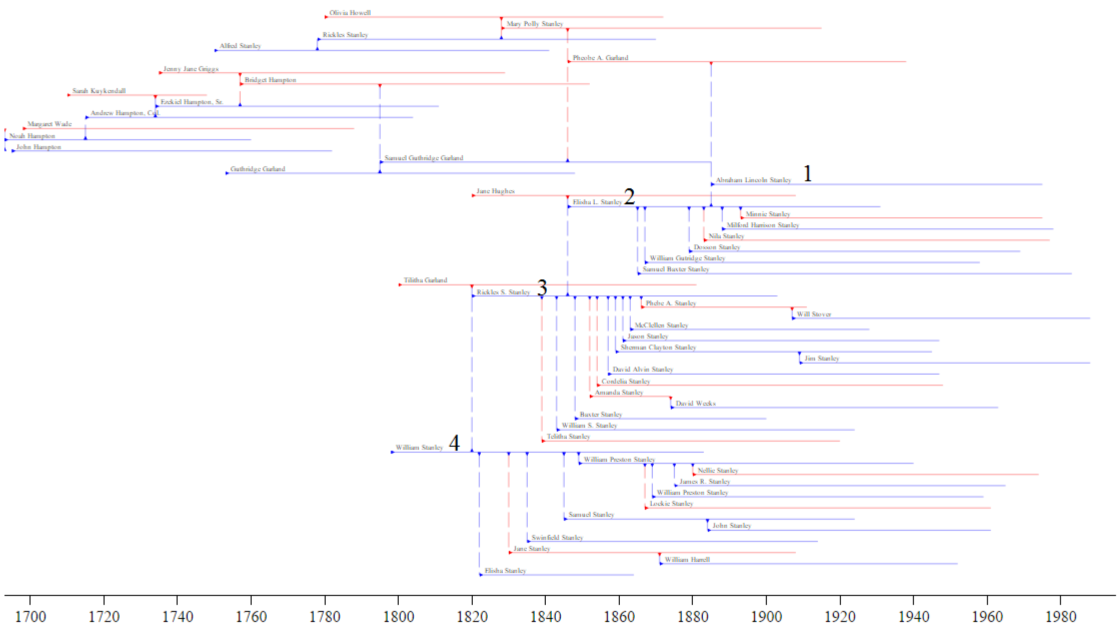


Figure 35: Example of combined ancestors and descendants trees. Person 1 is the central person. Persons 1,2,3, and 4 are the persons whose descendants trees can be built according to the rule.

build the tree of ancestors in that way so that birthlines path from the greatest paternal ancestor to the central person is always goes upwards as is required by the above stated rule. In fact that chosen ancestor gets positioned as the lowermost person in the visualization. After the tree of ancestors is built we build the trees of descendants for the ancestors that satisfy the rule. In fact they are the individuals that connect the chosen greatest ancestor with the central person. When the final visualization is build we allow the user to traverse the whole dataset by making all the lifelines clickable. In case if the user wants to choose another ancestor for whom the descendants tree is built he clicks on the lifeline of that ancestor. If the selected person has ancestors as well then we find his/her the greatest paternal ancestor. After we found the ancestor for whom to build the descendants tree, the tree is rotated so that the newly selected ancestor satisfies the rule and becomes the lowermost individual in the visualization. The descendants of the previously selected ancestor get collapsed and a new descendants tree is built.

In case if the user clicks on the lifeline of some individual that is not an ancestor of the central person then that newly clicked person becomes the central person, and everything is rebuilt from scratch as described above. In that way the user can explore the whole dataset by choosing the focal person and the ancestor who is the root of the descendants tree.

To allow the user to follow the transitions between states when tree gets rebuilt, we implemented animation. First the lines that do not persist into the new view are faded out, after which the lines that remain are smoothly moved to their new positions. Finally the new lines appear with a smooth animation. The transition time is about one second, which is fairly enough to follow the changes that are happening on the screen.

In the final visualization we show the descendants trees for all the individuals that satisfy the above stated rule. All of these descendants trees are subtrees of the descendants tree of the chosen greatest ancestor. In the case if the user does not want some of these descendants trees to be displayed, we allow to collapse them by a right mouse click on the corresponding individual. Another click expands the tree again.

Even though we show only a subset of a dataset it is still possible to the subset to be big in size. In that case the visualization gets also densely packed with lifelines. To still allow the user to interact with the visualization and to extract the needed information we implemented a zoom and pan technique. We decided that even if the age difference between the youngest and the oldest person in the dataset is around thousand years (which is not typical for the real world datasets) the horizontal extend of a screen is enough to get visually acceptable length of lifelines. Thus we implemented zoom only in the vertical direction that allows the user to increase the vertical distances between lifelines. Zooming in only in one direction makes it easier to explore the visualization as with the drag the user can move from one part of visualization to another without even zooming out.

4 Implementation

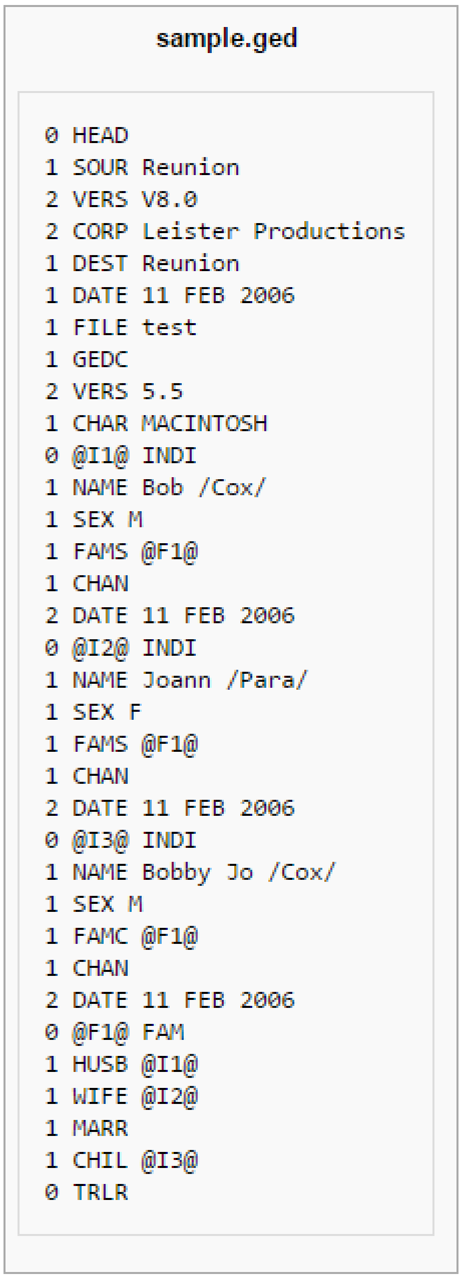
To evaluate how the proposed solution works in real world scenarios we built a software prototype and tested it on a real world genealogy dataset. It was also kept in mind that it could be tested and evaluated by non-IT-related people and it was important to make it easily installable and configurable. Nowadays the web applications are the easiest to start using as the user only needs a browser to run them. A web application can also be hosted on the internet and it can be used from any computer that has a browser and an internet access. Taking all that into account we built a web application prototype of our genealogy data visualization method. In the next sections we describe the technical details of our prototype such as data format, system architecture, tools used etc.

4.1 Genealogy data format

As our prototype software has been planned as a visualization tool only, we did not implement the functionality of manipulating the data itself such as editing existing data or adding new genealogy records to the data. The user should be able to import the data in some format and the tool will take care to visualize it. The "de facto specification for exchanging genealogical data between different genealogy software is GEDCOM" (Genealogical Data Communication) format [1]. A GEDCOM file contains genealogical information about individuals and meta data in plain text form. There are plenty of software packages that can create, export and import genealogy data in GEDCOM format. In our software prototype we allow the user to import genealogy data to be visualized in GEDCOM format.

An example of a GEDCOM file is shown in Figure 36. As can be seen from that figure every line in GEDCOM starts with a number. These numbers show the level of indentation and thus encode the hierarchy to indicate where that line belongs to. After the number there is a maximum four letters word (such as INDI, NAME, BIRT, DEAT, SEX, FAM) that is called *tag*. The tags indicate the type of record. For example INDI stands for the individual and FAM for the family. There are quite many tags that are defined in the standard but there is also a possibility to create custom tags. The problem with the custom tags is that most likely other software packages would not understand them. There are three main sections in the GEDCOM file. First goes the header section then the section of individuals' records and at last the trailer section. Every individual in a GEDCOM file as well as every family has an id that uniquely identifies them. Those IDs are also used to link the individuals with each other. As the GEDCOM format is very widely used in the genealogy community, there are quite many tools and libraries to parse the GEDCOM files and to extract genealogy information from them. For our backend software we used Java technology and thus to parse the GEDCOM files provided by the user, we used the gedcom4j java library[5]. It is an open-source and free-to-use Java library for parsing genealogy data. It also provides extensive functionality to manipulate data and do data queries such as get parents or children of an individual etc. For our testing we used genealogy dataset stored in GEDCOM format. That dataset includes records of 3,778 individuals dating back to 1600s. That dataset is submitted by visitors to Genealogy Forum and accessible through that forum [3].

[5]http://gedcom4j.org/main/


sample.ged

```
0 HEAD
1 SOUR Reunion
2 VERS V8.0
2 CORP Leister Productions
1 DEST Reunion
1 DATE 11 FEB 2006
1 FILE test
1 GEDC
2 VERS 5.5
1 CHAR MACINTOSH
0 @I1@ INDI
1 NAME Bob /Cox/
1 SEX M
1 FAMS @F1@
1 CHAN
2 DATE 11 FEB 2006
0 @I2@ INDI
1 NAME Joann /Para/
1 SEX F
1 FAMS @F1@
1 CHAN
2 DATE 11 FEB 2006
0 @I3@ INDI
1 NAME Bobby Jo /Cox/
1 SEX M
1 FAMC @F1@
1 CHAN
2 DATE 11 FEB 2006
0 @F1@ FAM
1 HUSB @I1@
1 WIFE @I2@
1 MARR
1 CHIL @I3@
0 TRLR
```

Figure 36: GEDCOM example [1].

4.2 Server-side implementation

As a server side technology for the prototype of our web application we chose Java Servlet technology. Servlet technology is a popular choice when it comes to building interactive Web applications. Servlets allows us to use all the advantages of Java programming language such as portability, performance, reusability etc. The HttpServlet class handles HTTP-specific services by providing methods such as doGet and do-

Post [6]. It is considered to be a good practice to separate the logic of the application from the representation. Probably the most widely used web architectural pattern is the Model-view-controller (MVC) pattern (Figure 37). As the name of the pattern tells for itself the application has three components. The central component is the *model*. The model manages the data and logic of the application. The *view* is the component that represents the information to the user such as html page. The third part is the *controller* which is the communication tool between the two other components.

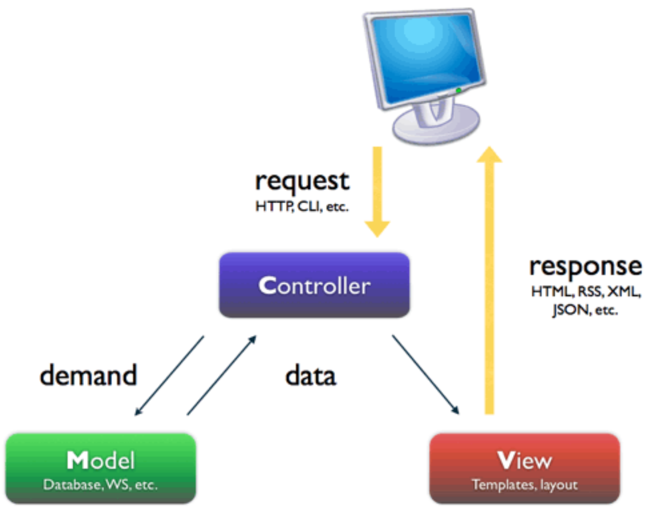


Figure 37: MVC pattern [7].

In our application the user sends a request from the HTML page, which is the view in that case, and passes parameters to the Servlet such as the id of a central person for whom to build the tree of ancestors and the id of an ancestor for whom to build the tree of descendants. The Servlet, which plays a role of the controller, receives the request and extracts these passed parameters. Then it passes these parameters to the model, which reads data from a GEDCOM file and builds the Dual-tree. After the Dual-tree is built it is passed to the Servlet controller and the Servlet controller updates the view to visualize that newly built Dual-tree. In a visualization task it is very important to make the interaction smooth and responsive, thus it was not acceptable to reload the page every time the user interacts with it. The page reloading always leads to a process of cleaning the view and redrawing the new view with the browser page going blank in between. That eventually leads to usability issues and the user looses track of the changes happened. To overcome that problem and to make the interaction continuous we used the AJAX (Asynchronous JavaScript and XML) technology. AJAX allows the web application to send and receive data from the server asynchronously in background. As it happens in the background it does not interfere with the display and existing page view. The process of sending and receiving the data does not refresh the page. Despite the name, we did not use the XML format for our server-client communication but instead we used data in the JSON (JavaScript Object Notation) format.

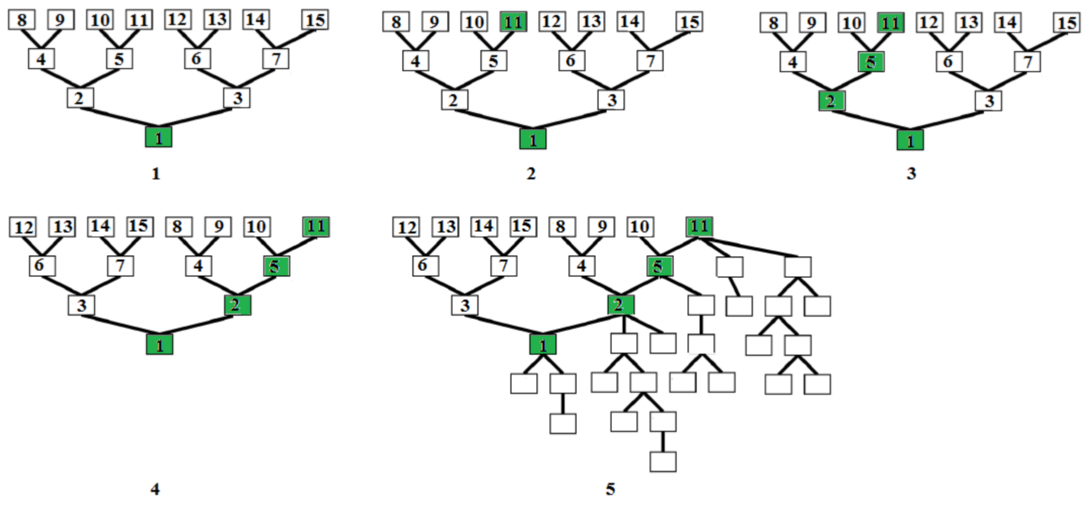


Figure 38: Dual-tree building algorithm.

The model part handles the Dual-tree building for the person whose id was provided from the view by the user. The Dual-tree building algorithm has 5 steps and is shown in Figure 38. The first step is to build the ancestors tree for the central person whose id was provided by the user.

After the ancestors tree is build we start the second step of our algorithm (Figure 38, *2*). We traverse the ancestors three that we just built and find the ancestor of the central person for whom we are going to build the tree of descendants. The id of that ancestor was provided by the user. If it was not provided then we take the greatest paternal ancestor of the central person.

After we found the ancestor, which is to become the root of the new descendants tree, and marked it, we mark all the descendants of that ancestor in that ancestors tree (Figure 38, *3*). Those marked individuals are the individuals for whom the descendants trees will be build.

Now as discussed earlier we need all the individuals, for whom descendants trees will be built, to be placed on the right edge of the tree (Figure 38, *4*). Once the tree is rotated we traverse it once again and build the trees of descendants of all the marked individuals (Figure 38, *5*). The result of that algorithm is the Dual-tree. We pass the newly built Dual-tree back to controller and the controller sends it to the view to be visualized.

4.3 Front-End implementation

When we were choosing the visualization library for our prototype we kept in mind that the visualization should be dynamic. The user should be able to choose the subset of the genealogy data to be visualized. As the user interacts with the view, the new data would come from the server side and the visualization would change according to the new data. The D3.js JavaScript library perfectly fits our needs as it allows to manipulate documents based on data[6]. The main functionality of that library consist of following steps: select DOM (document object model) elements, bind these elements to data and change the attributes of these elements according to the data. The selection operand filters the set of elements from the current document. D3.js uses the W3C Selectors API to identify and select document elements [12]. Once the elements, whose attributes we want to change, were selected, we bind them to the data. That data binding process is called data joins by the authors of the library. The data joins map selected elements to the data by key functions (e.g function that map id of elements to the id of data value). Typically it results in the situation where we have DOM elements that have corresponding values in the data, DOM elements that do not have corresponding values in the data and data values that do not have corresponding DOM elements (Figure 39).

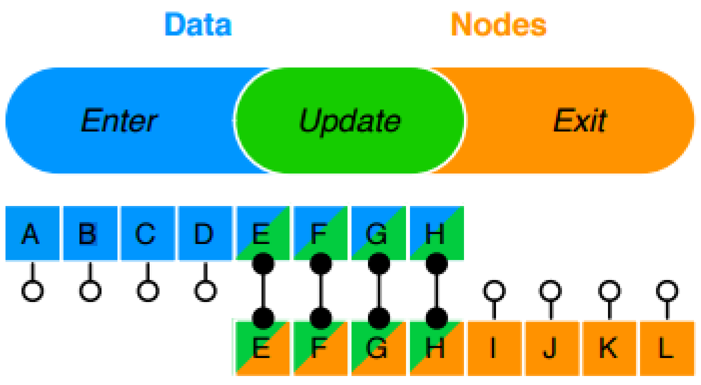


Figure 39: New data (blue) are joined with old nodes (orange), three subselections result: enter, update and exit. [12].

[6]http://d3js.org

For the data values that do not have corresponding DOM elements, a new set of elements are created and that set of elements is called the Enter set. The set of elements that have corresponding data values is called the Update set. At last the set of DOM elements that do not have corresponding values is called the Exit set. Now once the data binding is finished we can change the attributes of elements of these three sets.

In our case when the new Dual-tree data comes we bind it with the lifelines that are currently displayed. We find the lifelines that have corresponding values in the new data by id of the person (Update set) and update their positions according to that new data. The lifelines that do not have the corresponding values in the new data we fade them out from the view with an animated transition. For the new data values that do not have corresponding lifelines in the current view, we create new lifelines and make appear with an animated transition. The processes of sending a request to the server and receiving new data happen in the background with the help of AJAX technology as was discussed above. This makes it possible to update the view without refreshing the browser page and to keep the workflow continuous and smooth.

That data bounding to the DOM elements makes it possible to create dynamic visualizations. The visual attributes of the elements completely depend on the data and change whenever new data comes. The animated transition between the states allows the user to keep track of the changes that happened in the view.

5 Discussion and Conclusion

In this study we have done a review of existing genealogy data visualization approaches and tools. The motivation of this study was the need for new visualization tools because of the increased amount of genealogy data. Precise and extensive genealogy data opens a new horizons in the medical domain. Especially temporal information brings a lot of value for genetics researchers and gives more valuable facts to predict certain deceases that can be genetically passed from one generation to another with more precision. It can help to diagnose decease in the early stages and to take preventive actions on time. Most of the existing genealogy data visualization tools fail to handle big datasets and put little focus on the temporal information.

In our study we proposed a method of visualizing genealogy datasets that addresses the above mentioned limitations of handling big datasets and visually encoding temporal information. We extended the Dual-tree [26] approach by encoding temporal information about the lives of individuals in the form of lifelines. We also proposed a solution on how to visually encode decease information including such temporal facts as when the disease was diagnosed and cured.

Even though our software prototype was successfully applied to visualize a genealogy dataset consisting of more than 3000 individuals it still has the same limitations as the original Dual-tree method. One of them happens because of intermarriages. Even though intermarriages are not common in the real world families the statistical analysis suggests that all humans alive today have a common ancestor (not necessarily unique), implying that all humans alive are blood relatives [26]. Such situation creates problems when drawing one's tree of ancestors. In our prototype when we built the tree of ancestors, we kept track of individuals already added to the tree and did not allow duplicates. That problem needs investigation and a better solution on how to handle it without loosing important information and still be able to show these complex relationships.

Another problem that we encountered in our approach was the missing temporal information. As the position and visual attributes such as length of lifeline completely depends on the dates of birth and death of individuals, the absence of even one of these parameters makes it impossible to visualize the lifeline. In our approach we used a simple rule to estimate these missing parameters by limiting the age of an individual by 100 years. In case if both dates of birth and death are missing we just skip such a person and continue to the next person. Because the missing temporal information is often the case in the real world genealogy datasets that problem should be carefully studied and addressed. As a future work a better missing birth and death dates estimation algorithm needs to be implemented. On the other hand our prototype software has proven that the Dual-tree method that addresses the scalability problem can efficiently be extended by visually encoding temporal information.

References

[1] GEDCOM. https://en.wikipedia.org/wiki/GEDCOM. Wikipedia article. Accessed: 2015-07-16.

[2] Genealogy. https://en.wikipedia.org/wiki/Genealogy/. Wikipedia article. Accessed: 2015-05-10.

[3] Genealogy forum's gedcom library. http://www.genealogyforum.rootsweb.com/gedcom/gedr6152.htm. Accessed: 2015-05-31.

[4] Genelines. https://progenygenealogy.com/products/timeline-charts/genelines-sample-charts/pedigree.aspx. Software Homepage. Accessed: 2015-05-31.

[5] GenoPro. http://www.genopro.com/. Software Homepage. Accessed: 2015-05-31.

[6] Java Servlet Technology. http://www.oracle.com/technetwork/java/overview-137084.html. Online documentation. Accessed: 2015-07-16.

[7] MVC. http://code.tutsplus.com/tutorials/from-beginner-to-advanced-in-opencart-understanding-mvc--cms-2162 Internet article. Accessed: 2015-07-16.

[8] Tree traversal. https://en.wikipedia.org/wiki/Tree_traversal/. Wikipedia article. Accessed: 2015-06-21.

[9] Robert Ball and David Cook. A Family-Centric Genealogy Visualization Paradigm. In *Proceedings of 14th Annual Family History Technology Workshop*, 2014.

[10] Anastasia Bezerianos, Pierre Dragicevic, J Fekete, Juhee Bae, and Ben Watson. GeneaQuilts: A system for exploring large genealogies. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1073–1081, 2010.

[11] Ronald Bishop. In the grand scheme of things: An exploration of the meaning of genealogical research. *The Journal of Popular Culture*, 41(3):393–412, 2008.

[12] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3: Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 2011.

[13] L. Chittaro and C. Combi. Representation of temporal intervals and relations: information visualization aspects and their evaluation. In *Proseedings of 8th International Symposium on Temporal Representation and Reasoning*, pages 13–20, 2001.

[14] Andy Cockburn, Amy Karlson, and Benjamin B Bederson. A review of overview detail, zooming, and focus + context interfaces. *ACM Computing Surveys (CSUR)*, 41(1):2, 2008.

[15] Jesus Miguel de la Fuente. Visualization in Genealogical Data. `http://cs.lnu.se/isovis/theses/finished/13991.pdf`, 2011. Bachelor's Thesis. Accessed: 15.05.2015.

[16] Geoffrey M. Draper and Richard F. Riesenfeld. Interactive fan charts: A space-saving technique for genealogical graph exploration. In *Proceedings of 8th Workshop on Technology for Family History and Genealogical Research*, 2008.

[17] George W. Furnas and Benjamin B. Bederson. Space-scale Diagrams: Understanding Multiscale Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 234–241, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[18] Jeffrey Heer. DOITrees revisited: scalable, space-constrained visualization of hierarchical data. In *Proceedings of Advanced Visual Interfaces*, pages 421–424. ACM Press, 2004.

[19] Ivan Herman, Guy Melançon, and M Scott Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[20] Danny Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[21] Brian Johnson and Ben Shneiderman. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Proceedings of IEEE Conference on Visualization*, pages 284–291. IEEE, 1991.

[22] Kerstin Keller, Prahalika Reddy, and Shimul Sachdeva. Family Tree Visualization. http://vis.berkeley.edu/courses/cs294-10-sp10/wiki/images/f/f2/Family_Tree_Visualization_-_Final_Paper.pdf, 2010. Course project report. University of Berkeley. Accessed: 15.05.2015.

[23] Nam Wook Kim, Stuart K Card, and Jeffrey Heer. Tracing genealogical data with TimeNets. In *Proceedings of the International Conference on Advanced Visual Interfaces*, pages 241–248. ACM, 2010.

[24] A.M. Loh, K.W. Carter, S. Wiltshire, and L.J. Palmer. Celestial3D User Manual. Version 1.0.0. http://www.gohad.uwa.edu.au/software/?a=1250486.

[25] Angeline M Loh, Steven Wiltshire, Jon Emery, Kim W Carter, and Lyle J Palmer. Celestial3D: a novel method for 3D visualization of familial data. *Bioinformatics*, 24(9):1210–1211, 2008.

[26] Michael J McGuffin and Ravin Balakrishnan. Interactive visualization of genealogical graphs. In *Proceedings of IEEE Symposium on Information Visualization, INFOVIS*, pages 16–23. IEEE, 2005.

[27] Helen C. Purchase. Which aesthetic has the greatest effect on human understanding? In *Proceedings of the 5th International Symposium on Graph Drawing*, GD '97, pages 248–261, London, UK, 1997. Springer-Verlag.

[28] Edward M Reingold and John S. Tilford. Tidier drawings of trees. *IEEE Transactions on Software Engineering*, (2):223–228, 1981.

[29] Ben Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, VL '96, pages 336–, Washington, DC, USA, 1996. IEEE Computer Society.

[30] D. Paul Sullins. Creating a genogram. http://faculty.cua.edu/sullins/SOC206/Creating%20a%20genogram.pdf. Sociology course material. Accessed: 2015-05-31.

[31] Claurissa Tuttle, Luis Gustavo Nonato, and Cláudio T Silva. Pedvis: A structured, space-efficient technique for pedigree visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1063–1072, 2010.

[32] Benjamin Watson, D Brink, Thomas A Lograsso, D Devajaran, Theresa-Marie Rhyne, and Himesh Patel. Visualizing very large layered graphs with Quilts. Technical report, North Carolina State University. Dept. of Computer Science, 2008.