Graph Layout Algorithms - Readings

# SUGIYAMA, TOGAWA AND TODA

DO RESEARCH ON THIS, EVERYONE SEEMS TO BE TALKING ABOUT THIS!

Sugiyama framework / Sugiyama method.

# Representations of polygamy

Hott, J. R., Martin, W. N., & Flake, K. (2018). Visualization of Complex Familial and Social Structures. Electronic Imaging, 2018(1), 314-1-314–319. \

Based on chord and flow diagrams for family units.

Mormons -> polygamy, concept of “marriage”

Recognises complexities.

The objects of study in this research are the relationships people form at many different levels: parent/child, spouse/spouse, individual/family unit, family unit/family unit, individual/group, and group/group.

“We found that current visualization techniques are insufficient to fully express the level of complexity found in our familial structures.”

“This visualization breaks a few guiding principles for family unit depictions: the depiction should maintain a temporal ordering (temporality), participants should be displayed together as a unit (locality), and the types of spousal and parental relationships should be quickly discernible (distinguishability).”

GeneaQuilts [6], was created by Bezerianos, et al, to display large-scale genealogies of thousands of individuals

TimeNet visualization for genealogical data to better address and visualize each family unit’s temporal relationships.

Ball and Cook [2] define a similar time-line-based visualization scheme to address the connection of individuals in a common family unit. Their work depicts time vertically, with individuals as time-lines from their birth to death;

Ball and Cook [2] define a similar time-line-based visualization scheme to address the connection of individuals in a common family unit. Their work depicts time vertically, with individuals as time-lines from their birth to death; Our approach attempts to capture and visualize the temporal aspects of family units, similar to Kim and Ball, while at the same time maintaining family unit cohesiveness and providing a depiction of the larger genealogical flow and its evolution

We don’t care about timelines though.

A lot of them distinguish between gender, but in our case it is irrelevant / not important.

Also too many colours, difficult to discern.

The genealogical diagram is standard (Figure 4) yet the expanded version is difficult to understand. The chord diagram is highly unintuitive as a genealogical chart. Very esoteric.

Plus would certainly not work for other complexities such as disputed connections, and also multiple types of unusual relationships. Only works for polygamy types. Doesn’t look right without accompanying text.

Also the unusual parts of the polygamy is only distinguished by colour – ununintuitive. Something that looks more like a standard genealogy chart would be better. We also don’t care about gender, or whether the relationship was a co-parent or spousal relationship, so this style is unnecessariy complicated (Rube Goldberg machine)

Despite its emphasis on showing the famly units and the density, the whole timeline diagram is extremely unintuitive.

However I like the interactivity elements – clicking on the nodes to show the specific lineage (chord diagram) of that node clicked. For us the edges would make more sense though, and just change to the new node’s page when clicked.

Seems like each diagrammatic representation has a different purpose – different cultures value different types of representations:

“These lineage flow diagrams are still in their infancy: they can showcase familial interactions, cross-generational anomalies, and multiple concurrent marriages. We are considering further refinements and extensions to express more of the richness within the Nauvoo dataset.”

# Representations of Divorce/Remarriage - TimeNets

Kim, N. W., Card, S. K., & Heer, J. (2010). Tracing Genealogical Data with TimeNets. Proceedings of the International Conference on Advanced Visual Interfaces, 241–248. https://doi.org/10.1145/1842993.1843035

TimeNets – new visualisation technique to genealogical data.

According to TimeNets, there are two types of genealogical data:

“In a broad sense, there exist two types of genealogical relations. Parent-child relationships (consanguine relations) define a hierarchy in genealogical data. Relationships through marriage (conjugal relations) are non-hierarchical and merge family trees. Together these form a network of relationships— complex but simpler than a general graph. The most common genealogical research is ancestral research—tracing ancestry of self—and descendant research—finding descendants of an ancestral couple. They correspond to constructing a tree of ancestors and a tree of descendants. This observation verifies why ancestor (pedigree) and descendant charts (Figure 2) are canonical charting methods for genealogical data.”

*“*We present TimeNets, a new visualization technique for genealogical data. Most genealogical diagrams prioritize the display of generational relations. To enable analysis of families over time, TimeNets prioritize temporal relationships in addition to family structure. Individuals are represented using timelines that converge and diverge to indicate marriage and divorce; directional edges connect parents and children. This representation both facilitates perception of temporal trends and provides a substrate for communicating non-hierarchical patterns such as divorce, remarriage, and plural marriage. We also apply degree-of-interest techniques to enable scalable, interactive exploration. We present our design decisions, layout algorithm, and a study finding that TimeNets accelerate analysis tasks involving temporal data.*”*

*Really useful for temporal data.*

TimeNets prioritizes temporal relationships and family structure over traditional genealogical representations.

Interesting premise, but given the immortal nature of gods, this approach is not really feasible for our project.

The reasoning is:

“The most common task confronting genealogists is to correctly identify individuals and their familial and temporal relations. To keep track of their findings, people typically use genealogical diagrams, or “family trees,” such as ancestor (pedigree) charts and descendant charts (Figures 2a-b). By aligning people by generation, the charts prioritize the display of kinship relations, facilitating the identification of marriages, parent-child relations, siblings, and cousins. However, such representations often omit other aspects of genealogical data, particularly time. For instance, genealogists must frequently cope with temporally ambiguous evidence in order to establish kinship [15].“

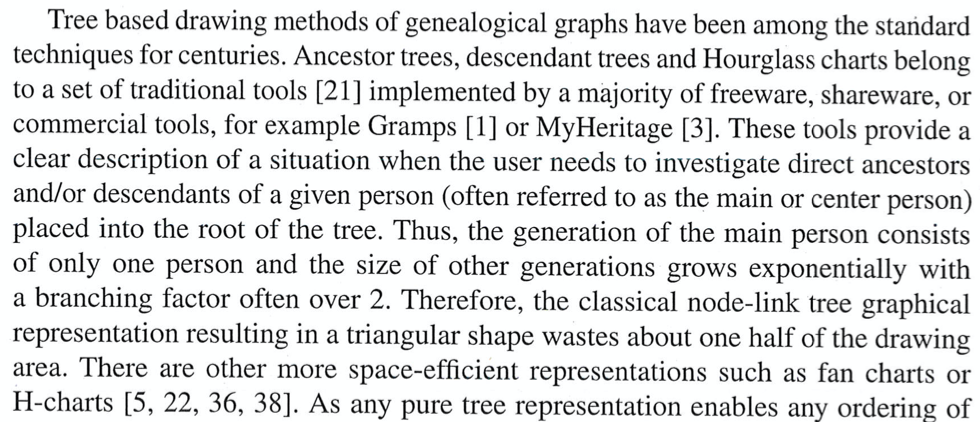
However I liked their ideas in Figure 8 – laying different connections over each other. Separating different spousal relationships – but in our case not over time but to reduce total overlap.

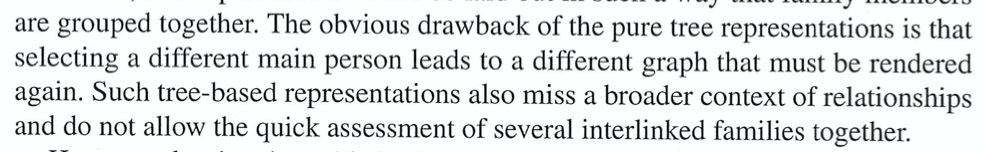
# Efficient Genealogical Graph Layout

Marik R. (2017) Efficient Genealogical Graph Layout. In: Cherifi H., Gaito S., Quattrociocchi W., Sala A. (eds) Complex Networks & Their Applications V. COMPLEX NETWORKS 2016 2016. Studies in Computational Intelligence, vol 693. Springer, Cham

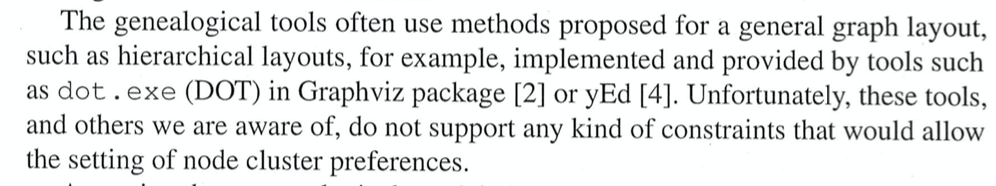
“

<https://link.springer.com/chapter/10.1007/978-3-319-50901-3_45>





“The situation with family grouping changes if the idea of a main person and their direct ancestors/descendants are dropped. Sometimes it’s beneficial to show the entire network of families instead in one layout – however we face issues with node clustering and edge crossing.



Want to minimize edge-crossing

“Formally, a genealogical graph is an acyclic bipartite directed graph G(V­p, Vm, E) with two sorts of nodes, people Vp and marriages/partnerships Vm. The edges E are directed from parent nodes to marriage nodes and from marriage nodes to children nodes.

In this paper, a node order reflects a linear sequence of nodes in a given layer. Does not address marriage between blood relatives (without blood marriages, forms a multi-trees).

Each node is assigned a “rank” level base on the number of successors, with regards to a base level. Uses a depth first search using an explicit LIFO queue, ensuring that all successors and predecessors get processed. Rank levels: Parents < marriage node < children. Might not work with intergenerational incest, however we can use this idea of rank levels (recursively) to determine the placement of nodes on the screen. (-1, 0, 1)

The idea of clustering nodes so that all the relevant connections are close to each other – limits edge crossing.

The layout design they propose for undirect spanning tree subgraph selection is interesting.

Automatically start from the node with the bottom-most rank in the family tree.

Use to determine where to subgroup the children in the layout to reduce the number of crossed edges.

Staggering node processing – LIFO stack for processed nodes, empty array for each node rank.

Useful for a huge number of generations, but it seems to rely too much on colours to distinguish between relationships. Would not be as clear if there were disputed relationships. Also the graphs we present don’t have to have this many generations, and needs to be clearly interactive. It needs to make the disputed relations more obvious, and with this quantity of relationships (+ how tightly squeezed they are), there is no room for the name of the entity or any other useful additions.

Also not immediately clear that the movements between generations are a single entity or a marriage between two entities. Seems to be single lineage, which is inaccurate.

An interesting idea, however this approach may not work for our type of genealogy chart – there are many cases where parentage is disputed (aka. someone could have multiple possibilities of parents), and this would result in many crossed edges. However if implement Greta’s solution about alternative parentage buttons, might be useful.

However this idea could be useful when determining interim layouts when depth search is implemented.

# Interactive graph layout – Henry

<http://scholar.google.com.au/scholar_url?url=https://dl.acm.org/doi/pdf/10.1145/120782.120788&hl=en&sa=X&scisig=AAGBfm1slYZtFUQg5PXAB1WNPQyp7WCDLQ&nossl=1&oi=scholarr>

This paper presents a novel methodology for viewing large graphs. The basic concept is to allow the user to interactively navigate through large graphs learning about them in appropriately small and concise pieces. An architecture is present to support graph exploration. It contains methods for building custom layout algorithms hierarchically, interactively decomposing large graphs, and creating interactive parameterized layout algorithms. As a proof of concept, examples are drawn from a working prototype that incorporates this methodology.

* Ask for links from Greta and Ben

# Hierarchical edge bundles: Visualisation of adjacency – Holten

http://scholar.google.com.au/scholar\_url?url=http://citeseerx.ist.psu.edu/viewdoc/download%3Fdoi%3D10.1.1.220.8113%26rep%3Drep1%26type%3Dpdf&hl=en&sa=X&scisig=AAGBfm2kGCn\_dINd\_7Sr2WjCb\_CWYJZqTg&nossl=1&oi=scholarr

Interesting idea of using polygonal edges (spline curves) for connections between nodes. (Fig 3)

Really interesting idea of graphs, using alpha blending. Look at the density of connections, particularly in the radial graphs (e.g. Figure 11 and 12). Would be incredibly useful for lots of data since is very visually appealing. However methods wouldn’t really work for genealogy chart, because the lines are blended together in the middle. We want the nodes to be distinct, interactable, and easy to read. The given examples won’t be easy to interact with (how would we visually trace the lines?). Also other difficult parts of greek mythology such as disputed relationships and strange marriages wouldn’t make sense here.

Plus they mentioned that there are collinearity problems with the rooted tree layouts (Figure 17) that make the tree less visually appealing.

# Online hierarchical graph drawing – North

http://scholar.google.com.au/scholar\_url?url=https://link.springer.com/content/pdf/10.1007/3-540-45848-4\_19.pdf&hl=en&sa=X&scisig=AAGBfm0hU\_yjwtQ-9WiSeQUOWq5a9qOEXQ&nossl=1&oi=scholarr

Abstract. We propose a heuristic for dynamic hierarchical graph drawing. Applications include incremental graph browsing and editing, display of dynamic data structures and networks, and browsing large graphs. The heuristic is an on-line interpretation of the static layout algorithm of Sugiyama, Togawa and Toda. It incorporates topological and geometric information with the objective of making layout animations that are incrementally stable and readable through long editing sequences. We measured the performance of a prototype implementation.

This one is about better interactivity and providing a dynamic hierarchical graph. Not so important for us because our information is unchanging – largely static. So this paper would be useful for constantly updating data, but not so useful for us. “Informative dynamic graph displays should direct attention to changes while also revealing the graph’s global structure.*”*

Uses something called “Dynadog” but I don’t know what that is.

# An Algorithm for Drawing a Hierarchical Graph – P EADES

<http://www.cse.unsw.edu.au/~lxue/publication/tute.ps>

Produces planar and symmetric drawings where possible. Uses the Sugiyama algorithm.

Works best for hierarchical graphs that are not too dense.

Creates an inwardly-bent data structure (Figure 2).

So much maths and theorems, wtf can’t understand.

But also does things through a path level system (k, k-1 path).

# A layout algorithm for hierarchical graphs with constraints – M Slade

<https://scholarworks.rit.edu/cgi/viewcontent.cgi?article=1638&context=theses>

Another one about minimizing edge crossings.

Talks about creating virtual nodes to compact a collection of nodes and links into a single node to simplify the visualisation when the graph gets cluttered with nodes and edges. The virtual node can be expanded to show additional detail.

Talks about how to show different hierarchies (Figure 7)

It changes the graph through stages (level passes) (Figure 12)

Works somewhat well (see last diagrams) but not the best.

Clearly there is a lot of work on minimizing edge crossings! Maybe I should do something with this? I don’t understand the mathematics behind them though.

Implement a edge-crossing-minimization algorithm in future works?

# Graph Hierarchical Layout Algorithm – ASAP Scheduling Algorithm – Lecture

<https://stackoverflow.com/questions/13861130/graph-hierarchical-layout-algorithm>

Taught me that that simple hierarchical layout algorithms are the ASAP scheduling algorithms.

Imagine, that you have list of node and you know dependence between them.

Node list

node4

node2

node5

node1

node3

node6

Dependency list

node1 -> node2

node2 -> node4

node3 -> node5

node1 -> node3

node3 -> node6

* As your first step, you should find nodes with no dependance - this would be your layer#1 nodes. Draw them.
* Then find all nodes that depends on layer#1 nodes - this would be your layer#2 nodes.
* And the same thing for the layer#2 and etc. Finally, you'll get:
* node1
* / \
* node2 node3
* / / \

node4 node5 node6

This will work only for non-cyclic directed graphs. For the undirected ones you should modify the algorithm a bit (take random node as root), but the main idea, I think, is understandable.

A screenshot of a cell phone

Description automatically generated

Now these ideas are okay in principle, would highlight what it means to draw edges between entities in the graph. However doesn’t really help with layout, because we don’t know what the dependencies in the graph will truly be. What about disputed nodes? Aka. an entity is both level 1 and level 0 at the same time? What about if we increase the generational depth – someone who used to be in level 0 could be in level 3 now?

HOWEVER, do more research into this. Determine how exactly this could be useful for our thesis.

Will it help with positioning the items? Determining how many nodes are in the same level?

# Hierarchical Drawing Algorithms – Brown CS – rtamassi

<http://cs.brown.edu/people/rtamassi/gdhandbook/chapters/hierarchical.pdf>

“A hierarchy is a cycle-free digraph where it’s useful for nodes of the graphs to be stratified into discrete, parallel layers” Drawing digraphs. Involves a pre-processing step that makes the connections acyclic.

Not particularly useful. Just a critique on the Sugiyama method really.

# Graph Layout Algorithms

<http://www.bii.a-star.edu.sg/achievements/applications/cellware/tutorial/page7-4.html>

“The general problem of the layout of a graph is NP-complete”.

Cellware, deals with reaction pathways. Want to avoid the crossing of edges and avoid the overlapping of nodes.

Has two types of graphical representations:

**Force-directed methods**

Based on Fruchterein, and Eades.

Aesthetic, potentially useful for tests.

We can try out a prototype of this in the meantime.

Also not every useful because we don’t have cycles.

Better for greater number of nodes.

**Hierarchical methods**

Better for graphs with linear and branched parts. Better for less number of nodes.

See diagrams in the above link.

# A Hierarchical Layout Algorithm for Drawing Directed Graphs – J Reynolds

<https://www.collectionscanada.gc.ca/obj/s4/f2/dsk2/ftp04/mq20694.pdf>

Focus of the thesis is on hierarchical layout drawing algorithms for directed graphs – specifically drawing larger graphs well in a reasonable amount of time. Solution improves efficiency dramatically, and a pre-processing step can be used to modify the structure of the initial input graph.

Also aims to reduce the amount of edge crossing in the layouts.

Mentions something called “flagdag” graph notation on top of “DOT” notation?

Graphs are difficult to understand. Nodes are well-laid-out, but the act of trying to remove the edge crossings the edges are extended outside (and hence loop around) the outside of the graph. This is highly confusing, and would be difficult to read for the general genealogy-graph-user.

# Layout of hierarchical flow charts – T Tusla

<https://dspace.cvut.cz/bitstream/handle/10467/70124/F3-DP-2017-Tusla-Tomas-Layout_of_hierarchical_flow_charts.pdf?sequence=1&isAllowed=y>

Deals with hierarchical flow charts. Modifies existing hierarchical layout algorithms. Maintains layout stability in subgraphs, and supports dynamic heights of layers.

Interesting idea of using flow charts as inspiration, since flow chart nodes contain content. Allow require different types of connections, which is not unlike what we’re trying to achieve with the greek graphs.

They want to use hierarchical layouts to help with creating hierarchical flow charts (Figure 6).

This thesis uses the idea of subgroups/clusters – only showing specific sections of the graphs, that the person can then click on to expand that part of the graph. This is a great method of decluttering! Not sure how useful this subgrouping will be for our project though, because people aren’t really in subgroups (except collectives) and we want to show just how crazy these greek realtionships are somehow, but it makes me think about ways in which I can declutter graphs with a dense number of relationships (e.g. Zeus).

AGAIN they reference Sugiyama’s framework!

Use old dummy layers? And assigning layers to nodes. This all helps with minimising edge crossovers.

Also a lot of these graph layout algorithms mention removing cycles, however this is not so necessary for our genealogical graph algorithm.

Consider dynamic graph gap tactics and remove edge decluttering (Figure 30).

The final result (Figure 34) isn’t the best, as is highly unintuitive. The edges are absolutely everywhere despite them having decent gappage. Howeever (Figure 34) the hierarchies are clear and there are no issues with ambiguity.

Application: “Ideal Graph Visualizer”.

# A fast heuristic for hierarchical Manhattan layout – G Sander

<https://link.springer.com/content/pdf/10.1007/BFb0021828.pdf>

Interesting approach, not sure if it’s the sort of thing I’m looking for though. This thesis aims to construct drawings of graphs from the viewpoint of compiler construction.

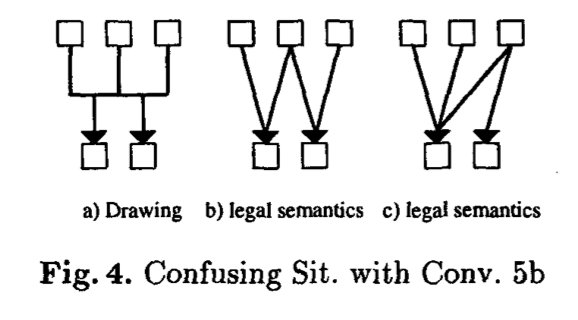
Manhattan drawings have edges that are sequences of line segments with a strict horizontal or vertical orientation (kind of like genealogical graphs!) and are used in VLSI design.

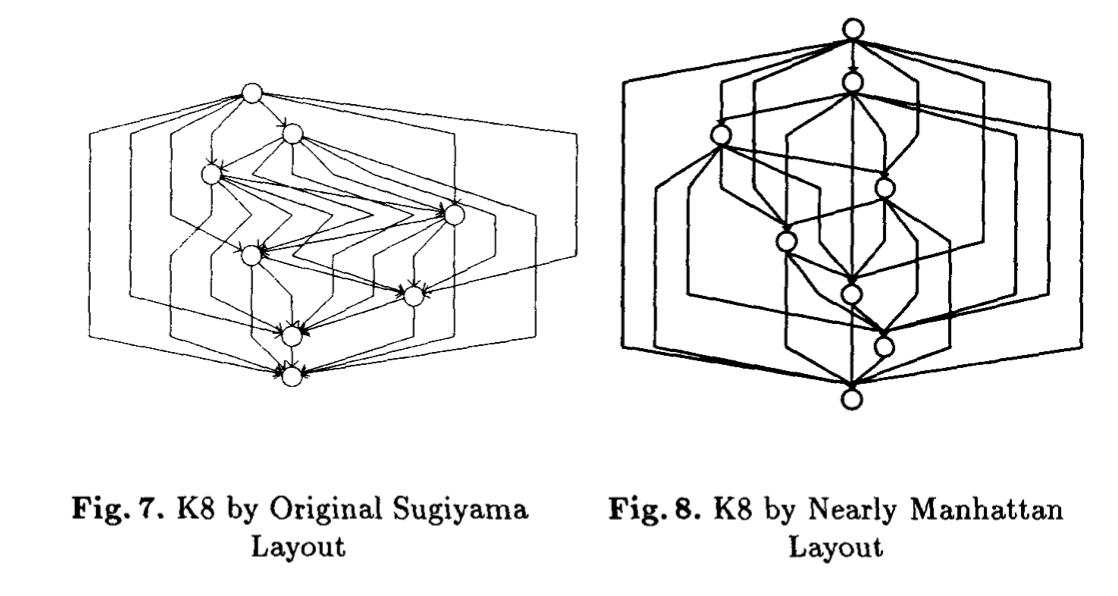
In VLSI, normally the nodes have a fixed place, while only the edges (wires) need to be routed.

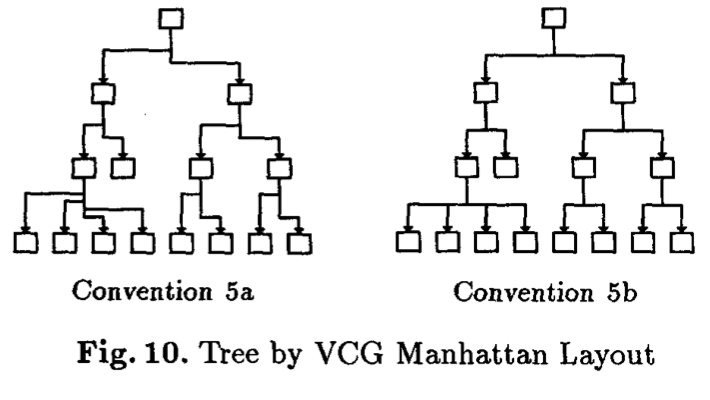
(Figure 4(a)) Drawing – kind of like a genealogical graph!

Also considered adding dummy nodes!

Also references the Original Sugiyama Layout!





**

A bit confusing to understand though.

# Steps of a hierarchical layout algorithm – Hanspeter Mossenbock

<https://www.researchgate.net/figure/Steps-of-a-hierarchical-layout-algorithm_fig18_221302634>



1. Remove cycles from the graph
2. Assign layers to the graph and make sure the nodes adhere to them
3. Insert dummy nodes so the connections are less bad (allows for segmented/bent edge lines)
4. Make them align vertically, and reduce the edge crossings if they’re obvious (current node placement does not matter)
5. Then finally, align the nodes to their correct spots on the page (horizontally)
6. Remoce the dummy node that you put in before and substitute with a bend in the edge!

May not be directly applicable to my thesis, but the adherence to layers, inserting dummy nodes, and reducing edges crossings could be incredibly useful! Would help me understand the theories behind all those other layout algorithms!

Now an issue with this – notice that A was in the middle at the start, and now A is at the top? It’s an interesting idea, however graph layouts if there are multiple disputed parents and the parents need to be at the top, could be confusing to read. What happens if we want to use that T-connection we see from standard genealogical charts? How would we go around organising that?

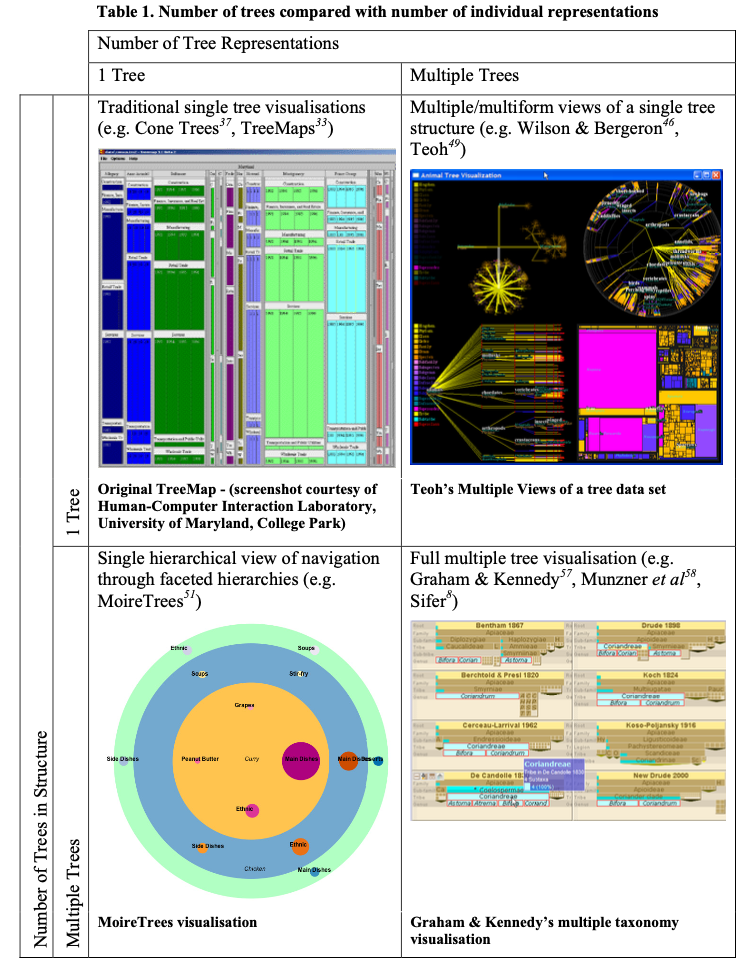
# A Survey of Multiple Tree Visualisation (Ben’s reommendation in the abstract)

Graham, M., & Kennedy, J. (2010). A Survey of Multiple Tree Visualisation. Information Visualization, 9(4), 235–252. <https://doi.org/10.1057/ivs.2009.29>

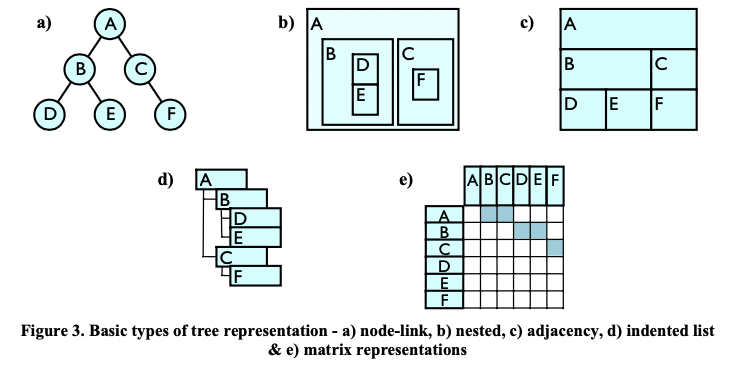
This article summarises the current state of research into multiple tree visualisations. It discusses the spectrum of current representation techniques used on single trees, pairs of trees and finally multiple trees, in order to identify which representations are best suited to particular tasks and to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

Goes through more different types of tree structures

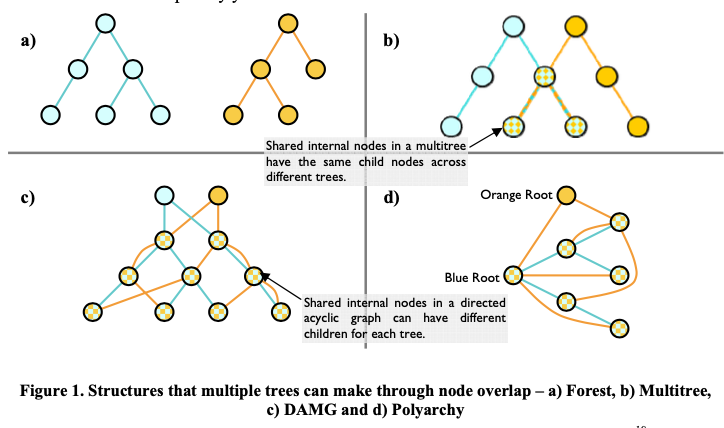
Multiple trees in multiple formats:



Single tree format types:



Considerations when connecting between them



Not really useful for us unfortunately. We don’t need to represent multiple trees. We can consider that the possibility of subgroups of trees linking together such as collectives or e.g. the titans. But generally we’d just show all of them on the page in one go. When would we ever use cases of two graphs in greek mythology joining together, that’s not just adding generations?