Thesis

# Introduction

Puzzle – other man – “Brothers and sisters I have none, but this man’s father is my father’s son”.

This puzzle’s solution is “my son”. However in this particular situation, with greek mythology, it could be something far more obscure – divine father, disputed father.

# Existing research papers

# Existing Greek Graphs – in society

## Greek graphs from Smith & Trazkoma book

Incomplete

## Greek graphs from Greek Gods Famly Tree – Chaos + NYX (see Graeme’s reference image)

Unintuitive

Just chooses to repeat connections instead of notify of incest

Entity1 + Entity2 rather than separate them into two nodes like in normal genealogical charts.

Unintuitive, since if same person mentioned in two different generations, don’t know if is the same person mentioned twice, or two different people with the same name.

e.g. Gaia actually bears Uranus with Aether, but Aether is ignored in the Gaia connection => it seems like Gaia bore Uranos through parthenogenesis. Also, Uranos is mentioned as Uranos + Gaia, but unless you knew it was meant to be Uranus and Gaia is the co-parent, you wouldn’t know if it was Uranos or Gaia that was the child of Gaia.

Can’t seem to address inconsistencies in the text – e.g. Iapetus + Clymene but in other sources it’s Iapetus + Clymene

Tasteful omission of different relationships?

Rather than recognises duplicates – chooses to omit duplicates altogether (avoidance!)

page2image6643968

## Gantz, Timothy. Early Greek Myth: A Guide to Literary and Artistic Sources. Baltimore: 1993

Another genealogical graph representation

## Fowler, Robert L. Early Greek Mythography. Vol. 2. Oxford: OUP, 2013.

Another genealogical graph representation

# Graph data formats considered

## Straight JSON

Easy to import the current relationship data in

## DOT Notation

## D3 Force-directed graph layouts

Interesting

Automatic force-directed, uses physics engine

Displayed perfectly centre, very efficient and fast rendering

Lots of different styles of data – can have many:many linkeages with no issues

Highly interactable components

However can’t modify the layout easily

Also doesn’t look like a genealogy chart – confusing from a UX perspective.

## D3 Hierarchical layouts – “treemap” layout

D3 is very commonly associated with graphs. Closest thing is a hierarchical layout diagram. Can easily centre, and is interactive. Looks like a family tree structure.

However Doesn’t do the t-linkeages. It’s just single node linking to another node, no t-join in-between

Is also in a standard linear/vertical tree format. The data structure is also structured distinctly hierarchical. Doesn’t recognise duplicate notes (which is prevalent in greek myths with multiple parents and intergenerational incest). And wouldn’t be able to do intergenerational or sibling relations.

A close up of a map

Description automatically generated

<https://www.d3indepth.com/layouts/>

## OrgChart Hierarchical / family tree layouts

OrgChart is a bit better – has t-joins in between and looks more like a genealogical chart.

Decently fast rendering, and highly intuitive family trees.

Unfortunately also uses the same hierarchical data format (linear) as the D3 hierarchical chart. Not feasible for greek myths

<https://codesandbox.io/s/react-orgchart-demo-sjq85>

<https://balkangraph.com/OrgChartJS/Demos/BasicUsage>

But is also a family tree generator

## GraphLib

Not directly a data representation format, however is used to store the information about nodes and edges.

Contextualises the information into nodes and edges, saves having to generate this ourselves. Has a lot of implicit functions already provided for finding edges and nodes, and formatting them etc.

However does not create an actual database – we generally use an existing database of information to use GraphLib. Difficult to understand the complexities of this kind of data.

However upon investigation this seems to be the best method of storing our information. A combination of JSON converted information from the original CSV file, to graphlib objects.

## GEDCOM

The standard format for genealogical data.

(GEnealogical Data COMmunications)

Developed by the Church of Jesus Christ and Latter-Day Saints. Used for recording baptisms(?)

Text file.

**Pros:**

Recognised standard for genealogical data

**Cons:**

According to 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 : “Although a variety of genealogical data formats exist, the de facto standard within the genealogical community is GEDCOM [8]. Accordingly, we parse GEDCOM files as one data source for TimeNets. Unfortunately, the GEDCOM specification can not represent many types of interpersonal relationships, including same-sex marriage, polygamy, and incest. In response, we developed our own data model for genealogical data. The first step in our pipeline is thus to ingest data from an external source—such as a GEDCOM file or web repository such as Freebase [6]—and map it to our data model.”

# Types of graph layout software / graph visualisation tools considered

## GraphViz + Dot

<https://stackoverflow.com/questions/2271704/family-tree-layout-with-dot-graphviz> – making it so it looks like t-junctions.

## DagreJS asnd D3.js + SVG?

Really good

Allows for interactable elements – good for linking back to the source content and highlighting the individual unusual relationships for focus.

Unfortunately is impossible / difficult to format the existing layout algorithm. And also genealogical graphing is not accessible.

Automatic formatting. Also doesn’t do the t-relations required in genealogical charts.

## Canvas.js

<https://canvasjs.com/react-charts/>

No direct visualisation algorithm for directed acyclic graphs

But can use as a blank canvas for my own graph layout algorithms

Allows for interactable elements of the graph, see canvas.js layouts

Interactive but doesn’t seem to be able to do diagrams properly

Also seems to always include axes.

## OrgChart

<https://github.com/dabeng/OrgChart>

Simple family tree generator

Looks exactly like what we want, but not necessarily customizable.

## STORM React-Diagrams

<https://github.com/projectstorm/react-diagrams>

## Roots (MacOSX)

Not built for Mac Catalina – currently in development

## React-Konva (React + CanvasJS)

The most initial prototype

<https://github.com/yayalu/greekgraphs/commit/233c2d14694716608da15a46c94b471ad71f0ac0>

See here that an attempt was made to get an initial setup working.

Decided to first get a generic prototype using existing software working, see how we feel about what it looks (see GraphLib) but once we tested it we realised it was unaesthetically pleasing. So now going back to Konva.

Not easy to use – with React

## HTML Canvas tag

Interesing, most effective and has most support, however the components, once drawn, are not easily interactable.

## React-diagrams

## Storm react diagrams

## FabricJS

## Legacy 9 – not PC compatible

Accessible for different types of data input, for example can import GEDCOM text files.

**Pros:**

* Relatively intuitive. Used to import information about general genealogies.
* Can put as much information as necessary into the graph (e.g. can just input first name, date of birth is optional, etc.)
* Can input the number of generations wanting to show
* Automatically displays the information in the desired genealogical format
* Can determine whether to show ancestry or descendants (but not both)
* Can choose which entity is the focus entity
* Imports existing file data such as GEDCOM
* Allows for different genealogical (family tree) formats, e.g. traditional nodes and edges, or ancestry fan
* Allows for different layouts depending on page size (e.g. horizontal and vertical family tree representation)
* Immediately obvious is a genealogy chart
* Allows for mistresses, not just spouses

**Cons:**

* Costs money
* Can’t modify the layout once generated
* Layout algorithms not directly accessible and not able to be converted (no API) for putting that data on the screen – generate dynamically, is not interactable
* Assume every entity is human (but can name them “void”)
* Can handle brother and sister relationships, but not inter-generational relationships (integral to greek mythology)
* Can’t handle disputed parents (e.g. multiple mothers, disputed)
* Can’t list alternative parents (‘other parents’) and cannot express uncertainty/dispute in the graph

**Different formats for Graph layouts**

### Ancestor Fan

<Image here from Legacy 9>

Focus is on the main node in the middle – exactly the kind of focus we want for our implementation.

Doesn’t know how to address second marriages.

The ancestor fan can’t handle incestuous relations – it creates duplicates rather than link the same names.

Duplications mean excess space used in the fan. Duplicates the parents for the siblings as well.

Unintuitive

Also doesn’t work because only shows the immediate parents for each inner node – could not handle normal parents + divine parents OR disputed parents either.

Also automatically add different colours for different kinds of nodes, don’t know why or what they mean.

The relationships between the parents are also not obvious.

### Timeline

Interesting graphing format.

If know dates is useful as it gives an indication of who is alive at the same time.

However is not the best one that addresses the thesis’ research question.

Also not so useful for greek myths – big disparity between different entities, because gods are immortal => humans will be dots and gods would expand the entirety of the timeline.

Also timelines are disputed / uncertain, so there’s no way to show disputed timelines in this graph.

<See Legacy 9 sample graph – Asa Clark Brown>

<Also see TimeNets research paper>

### Booklet

There are two types of booklets:

1. Booklet of ancestors of Entity X
2. Booklet of descendants of Entity X

You can add notes about each of the entity relationships – but incest is not immediately obvious.

<Image – Ancestors of Epimetheus [55]>

A page for every generation, with table of contents and a name index at the back, with details about the people in every generation.

Really useful, textual information, formatted really well. But not visually appealing.

Not appropriate for this kind of project.

Could be useful for descriptiveness about disputed relationships – maybe useful for the info pages, but not useful for the visualisation itself.

<Image – Descendants of Aether[31]>

Lists all the descendants in textual form in order, and goes to the genealogical depth (of ~3) for each member.

Interesting, but no generational heading so difficult to read.

Especially going over several pages – no distinct vertical line showing what generation the entity in question is.

Not as intuitive as Ancestors booklet

Name index provided with associated page mentions.

Overall issues:

Recognises duplicates, but doesn’t omit the duplicates, just notifies the graph-creator that there are.

Also can’t put descendants and ancestors in the same booklet.

### Chordal diagrams

See research paper

### Genealogical chart

The thing that legacy 9 is most famed for

Can add extensive details to each node.

<See Legacy 9 sample graph – Descendants of Chad Brown>

Can add lots of details to every node, including pictures

* Can’t show descendants and ancestry in the same graph? – can (X-chart)

Can’t seem to acknowledge duplicates

Not immediately obvious what the colours mean (needs a legend)

Connections not intuitive (auto-put in by program)

The edges and nodes are NOT interactive (can expand and contract but not interactive)

If a mother and a father are within the same family tree, then the entity’s relationship will be mentioned twice in the graph – program will automatically write down the relationship as “Duplicated”, but it will not actually remove that relationship. This is especially the case with intergenerational incest. Or even sibling incest. Unintuitive. Also duplicates are not immediately obvious – it’s difficult to detect them. Also the descendants of duplicates is only shown in one version of the duplicate connection. E.g. see the number of times Zeus is referenced in the diagram. Will make it difficult to determine if it’s the ame entity repeated, or it’s different entities with the same name.

The connections are difficult to read. Marriages and siblings are on the same plane as the entity, which makes sense. However, the marriages with the entity are stacked. E.g. if Entity X has spouse Entity Y, and Entity X also has spouse Entity Z, then the graph shows that connection as Entity X=Entity Y=Entity Z, which is unintuitive. It makes the user believe that a connection between Eneity Y and Entity Z exists, which is incorrect. Need some way to separate the lines between spouses.

Single parent looks the same as cases of parthenogenesis – single line from entity only.

No ability to show disputed parents / lineage.

Spouses and co-parents have the same connection style (====)

Current display format makes finding errors in the diagram difficult. E.g. Ouranos married to Zeus?

Different colours added to diagram without prior consent or legend – what do they mean?

Doesn’t adapt for unusual connections e.g. Aphrodite and Uranos’ genitals (actually autochthony). The user can write notes about the specific connection and how unusual it is, however the notes are now shown in the node itself – what’s the point?

Additionally, if an entity has multiple spouses, the children line comes from the spouse node and not the connection between – interesting. Worth putting in my prototype to avoid the issues with curved connections? Just make the cases of parthenogenesis more obvious through the “unusual connection” identifier?

# Different layout algorithms

## Binary tree structure

**Pros:** commonly associated with genealogy, shows hierarchies and generations

**Cons:** Don’t deal with strange intergenerational connections. Certain that each entity will have more than one parent, and will possibly have more than 2 children. G tree is not valid either as does not deal with intergenerational.

## Cyclic graphs & Bidirectional Acyclic Graphs, Directed Acyclic Graphs,

Not possible. Person can’t be their own grandparent.

Needs to have both direct and undirected components. Don’t need to do operations on them yet so direction does not matter.

Maybe allow specification of directed or undirected connection? Hybrid?

## Force-directed layout for DAG

**Pros:**

Standard visualisation format for DAGs. Readily available for use and experimentation in different visualisation software, esp. D3.

**Cons:**

Nothing like genealogy graphs. Very few of them have formatting for genealogies. Need the hierarchical structure since otherwise it is unintuitive and doesn’t look like a genealogy chart (connections not immediately obvious)

Tried this, worked well as an early prototype but was clear that this was unintuitive. So started from scratch with a bespoke algorithm.

## Hierarchical layout for DAG

**Pros:**

Significantly closer to expectations for genealogy charts.

**Cons:**

Generally does not have support for more unusual relationships (e.g. intergenerational).

## Ancestor Fan

**Pros:**

Another well-known version of genealogy charts.

Could be interactable.

Immediately obvious who is the centre entity / point of focus.

**Cons:**

Would not be able to show intergenerational incest, and is difficult to read.

Would be hard to modify

Does not address duplicates well

# Different relationship types considered

Insert the screenshots of the digitized plans for showing relationships.

A screenshot of a cell phone

Description automatically generated

­­­­­­­­­­

# Early prototypes:

## Prototype 1

Prototype developed using GraphLib, DagreJS + D3.js and rendered as an interactable SVG format. Used to determine the pros and cons of the various data formats considered, used as inspiration for the bespoke graphing tools and graph formats.

Existing relationship data reformatted into GraphLib obejcts, and then converted to graph visualisation using the DagreJS + D3 graphing tool that creates a hierarchical graph layout. Shown on the page as an interactable SVG format

A picture containing text

Description automatically generated

Graph allows for automatic hierarchical formatting and placement. As you can see, all of the parents of Apollo’s children are in the same horizontal placement as Apollo. Also each node (thanks to GraphLib) can have “parent” attached to it to allow automatic detection of hierarchies, and the edges can “appear” as bi-directional if you want by omitting the arrow in the style tag. However the siblings of Apollo were put on the same level as his children, and there was no way to change this.

Is highly interactable, scrollable, zoomable and scalable so is very usable. By clicking on the nodes you can go to the entity’s page. However, the layout of the graph itself cannot be modified easily, and it also lacks the t-connections that genealogy charts are known for. As this is the case, it’s difficult to distinguish between siblings and mistresses and spouses, and even more confusing to add things like intergenerational incest. Both parents of an entity have to be inferred.

I added a superficial solution which is to add labels to each of the edges – makes the edge type more explicit (child vs sibling vs other parent), but this is not very user-friendly, especially if there are lots of different kinds of labels in the same location. For example, Zeus has a huge number of mistresses and offspring. This was impossible to read, even if the diagram was scrollable and interactable. The graph layout algorithm for D3 is also not customizable, so we can’t find alternative layout formats for these graphs.

A screenshot of a social media post

Description automatically generated

GraphLib also has removes duplicates by default – counter-productive for duplicate relationships such as incest. For example, Zeus and Hera are siblings but also husband and wife. So Hera node is created, Zeus node is created, and then the algorithm encounters “Hera is wife of Zeus” so the edge Zeus->Hera with label “wife” is added to the graphlib object. However, then the algorithm encounters “Hera is sibling of Zeus” and instead of creating a new edge between Zeus and Hera with the label “sibling”, it just overwrites the existing “wife” label. As such the resulting graph will only have one edge between Hera and Zeus – “sibling”.

Of course you could modify it so that “sibling” is appended to the existing “wife” label to create “sibling/wife”, but with this it’s not immediately obvious that there is an unusual connection there. We want to make it immediately obvious to the user that it’s unusual. As such we couldn’t get these unusual relationships to work in the prototype.

A close up of a mans face

Description automatically generated

We did create an easier prototype for unusual aspects in greek myths – disputed parents. These edges were different colours to signify that it’s not a usual connection, and the links were clickable, but only upon request.

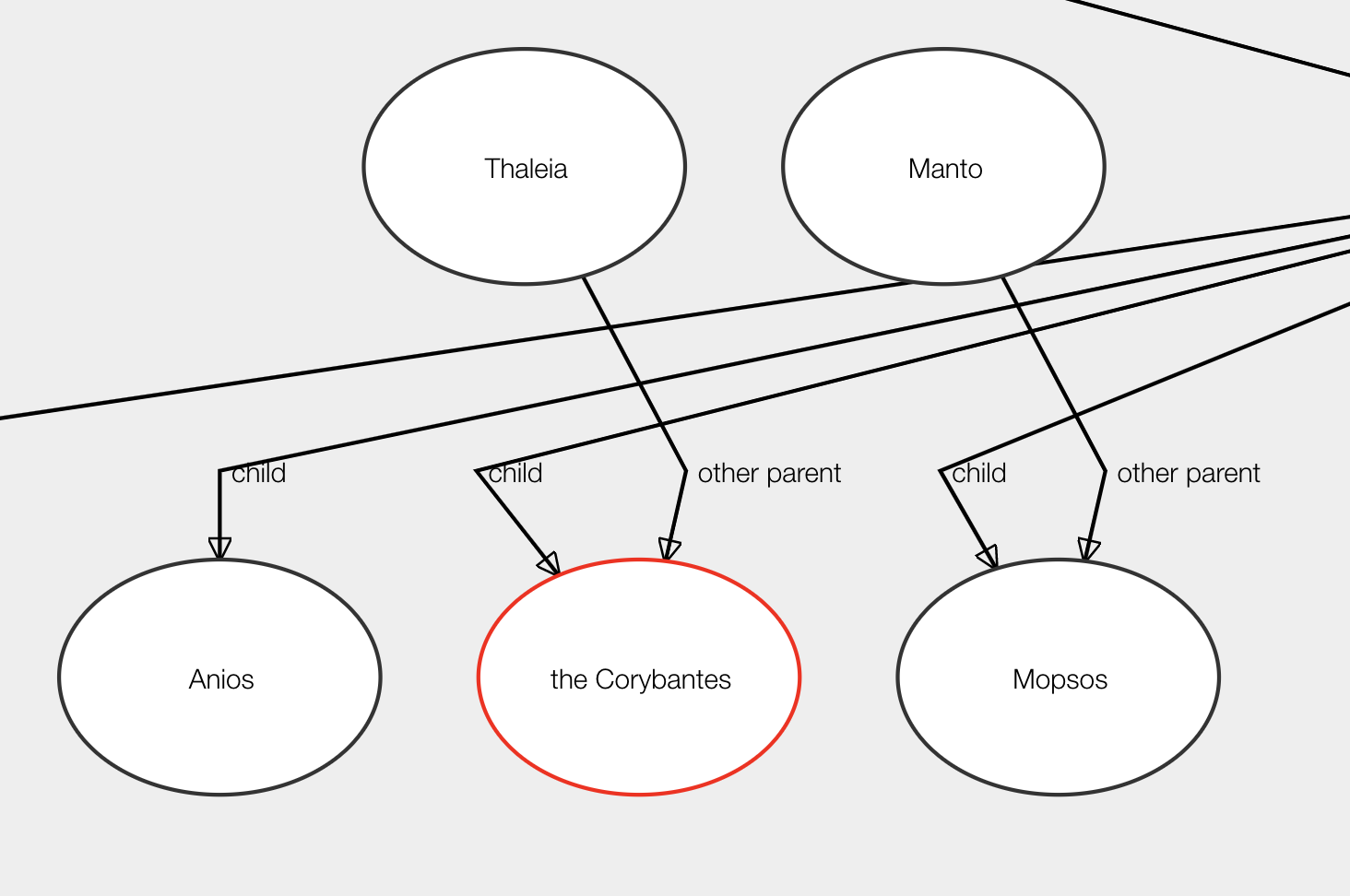
By clicking on the disputed edge, a new page would pop up detailing the unusual parts of this relationship, with links to the location in Scaife of the disputes, and as well as links to their entity pages.

A picture containing screenshot

Description automatically generated

There was also a brief addition of member nodes – signifying the differences between the collectives and the parent nodes.

Originally each of the collective pages contained a graph with the members of the collectives as \*children\* in dotted lines, however this was removed as it was deemed irrelevant to the genealogy by Greta.



The following collective graphs were removed. As you can see they were barely useful for showing collectives that were genealogical, such as Danaids: the daughters of Danaos. However they were not at all useful for other kinds of collectives, such as the Greek Contingents at Troy.

A close up of a device

Description automatically generated

^ Usefulish

A close up of a device

Description automatically generated^ Not at all useful

So decided to get rid of them.

Greta and Ben’s comments:

And then only differ from that traditional family tree where those are unusual relationships.

It is quite lkely that sticking to traditional family tree is the best approach, but we can't pre-empt these answers to the question (that's what a user study is for).

Is the current approach okay even for a first study?

Greta has trouble understanding the diagram

Document this stage of the project,

Prototype - as a result of this informal testing, and upon discussion with Ben and Greta, it was clear that we had to change it so it had more flexibility with layout of data, etc.

Create a new branch to work on a bespoke

Graphing layout algorithm

There are also no useful hierarchical options for D3JS

## Prototype 2

Commit log number 2532cdc397720d1259927629deb0d23eac9bad8d

A picture containing screenshot

Description automatically generated

Half-half, top half normal genealogy chart

Bottom half experimenting with large quantities of relationships. Clearly design not obvious

Either:

1. Group all related relationships together to make these clearer (not really helpful for distinct relationship types)
2. Try to stagger the genealogy diagram from the top part in the bottom.