KTH Computer Science
and Communication

Which type of visualisation technique, among the currently most employed standards, is best suited to visualize big and complex networks?

Theory and background
Date - 2014

VIKTOR GUMMESSON
vgum@kth.se

Master's Thesis in Computer Science
Royal Institute of Technology

Supervisor, KTH: Olov Engwall
Examiner: Olle Bälter
Project commissioned by: Scania
Supervisor at Scania: Magnus Kyllegård

Abstract

Referat

Contents

Chapter 1

Introduction

This chapter is intended to give an introduction to the subject of visualizing networks.

1.1 Background

To be able to visualize different networks is an important part in many fields, such as science and technology. For example, computer science that deals with complex networks of relationships between system components, displaying relations in a social network, molecular biology that study the interactions between various systems of cells, e.t.c.

There are different approaches to take when visualizing networks. The most traditional approach is to represent the network as some kind of graph, because when many structures in different scientific fields can be represented as anode-link graphs. Where nodes represents different components and are visualized with a shape and edges represents different components relations and are visualized by a connecting line between two nodes.

1.2 Arising problems with growing data

Though the traditional ways of visualizing graphs are pleasing and gives in intuitive way of looking at relations there arises problems when the networks that needs to be visualized is of a bigger size. The traditional ways may be sufficient when dealing with networks of small sizes of nodes and relations, but what happens when the networks become complex and has hundreds or thousands of nodes?

1.2.1 Edge and node crossing

When the node count becomes larger the area dedicated to layout these becomes smaller. This can contribute to that nodes starts to overlap etch other, making it hard to distinguish between a set of different nodes.

A similar problem arises concerning edges. Depending on the layout of the nodes a different amount of edges may overlapp, crossing eatch other. This may not be a problem if the number of crossings is low or the angle between two edges are high. But when this angle decreases and the number of crossings increases it becomes hard to distinguish between edges, to see which edge connects which node. If the relations are big enough the cluster of edges may become as just one big black area.

When dealing with layout techniques one strives to layout the nodes in a way minimize node- and edge crossings.

1.2.2 Labeling

Labeling nodes and edges in a network becomes more challenging as the network grows. In fact the optimal label placement of a graph has been shown to be NP-Complete[14]. One can see the task of labeling to be devided in to three differnt labeling tasks:

- Labeling area features (clusters).
- Labeling line features (edges).
- Labeling point features (nodes).

1.2.3 Situation awareness

Human and psychology factors play a role when visualizing a network, situation awareness is a term in this aspect. Endsley[4] defines situation awareness as:

> *Situation Awareness is the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status into the near future*

Situation awareness become an important to consider when choosing visualisation technique.

Figure 1.1 shows what can happend when trying to visualize big networks.

1.3 This thesis

This thesis revolves around the question:

Which type of visualisation technique is best suited to visualize big and complex networks?

With the corresponding hypothesis that:

One can conclude on a visualisation technique that is best suited to visualize big

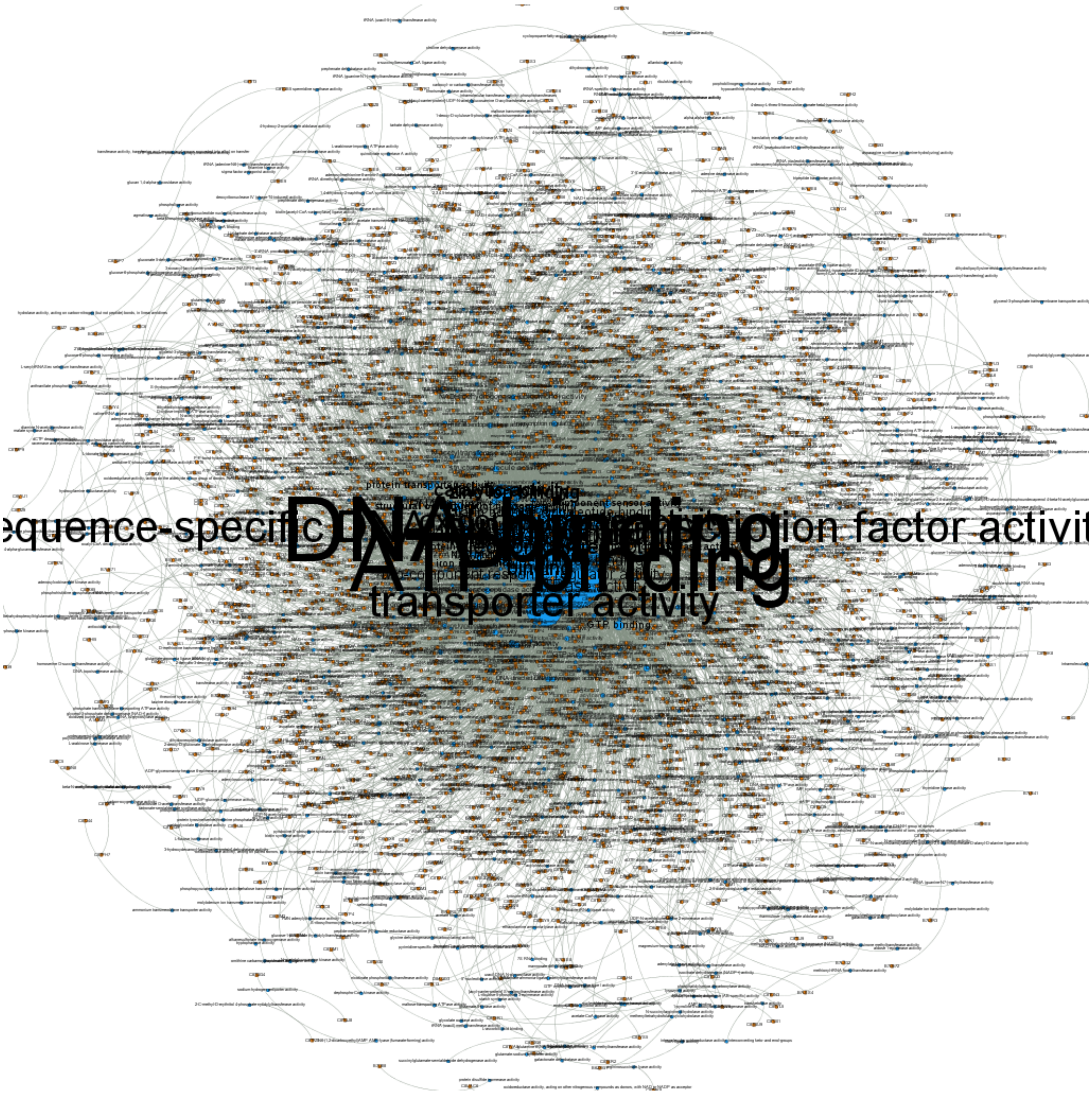


Figure 1.1.

and complex networks.

In chapter two different common visualisation techniques is described. Chapter three talks about tests done to evaluate a set of different techniques. Chapter five provides the results from chapter 3. Chapter six discusses the results.

Chapter 2

Visualization techniques and their theory

There exists a number of different approaches and techniques used to visualize big and complex networks. This chapter is intended to introduce some of these and explain how they work.

2.1 Common techniques

Though there exists a number of different approaches many of these is based on some fundamental technique or concept. Two major aspects is important to consider when trying to visualize a network. First is about the part that most probably relate to graph visualization, the actual layout algorithm that decides where each node is to be placed and how the edge routing is gonna go. Second is the aspect of how one is to navigate a graph when it have been generated. Navigation such as zooming and panning.

2.1.1 Force-Directed

Force-directed is a popular class for a type of algorithm for calculating layouts of graphs. They are constructed to strive towards generating graphs with node positions so that edges in the graph are of equal length and the layout displays as much symmetry as possible. One of the pros with these algorithms is that they are flexible, they do not relay on domain specific knowledge but instead only uses the information contained within the structure of the graph. Graphs produced by these algorithms tend to be aesthetically pleasing and exhibit symmetries[10]. Figure 2.1 shows an example of an graph drawn with a force-directed algorithm.

These algorithms are based on assigning forces between nodes and edges in a graph, simulating the motion of the edges and nodes or minimize their energy. One of the first force-directed algorithm dates back to 1963 with the algorithm of Tutte[25] and is based on barycentric representation[10]. Though the more com-

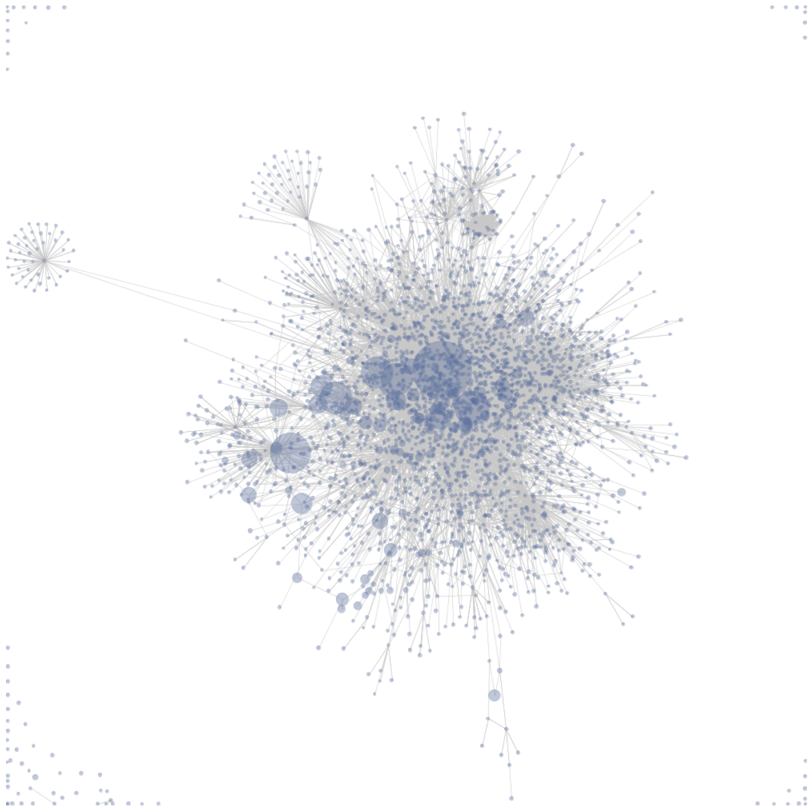


Figure 2.1. Visualization of links between pages on a wiki using a force-directed layout.

monly used algorithms such as Eades[3] and Fruchterman and Reingold[5] both relay on spring forces similar to those in Hooke's law. Here there are repulsive forces between all the nodes in a graph while in the same time attractive forces between nodes and there neighbours. Bellow is a summarization of the Eades aglorithm[10]:

To embed a graph we replace the vertices by steel rings and replace each edge with a spring to form a mechanical system. The vertices are placed in some initial layout and let go so that the spring forces on the rings move the system to a minimal energy state. Two practical adjustments are made to this idea: firstly, logarithmic strength springs are used; that is, the force exerted by a spring is:

$$\mathrm{c1} \times \log \frac{d}{c2}$$

where d is the length of the spring, and c1 and c2 are constants. Experience shows that Hookes Law (linear) springs are too strong when the vertices are far apart; the logarithmic force solves this problem. Note that the springs exert no force when d = c2. Secondly, we make non adjacent vertices repel each other. An inverse square *law force,*

$$\frac{c3}{d^2}$$

where c3 is constant and d is the distance between the vertices, is suitable. The mechanical system is simulated by the following algorithm.

```
algorithm SPRING(G:graph);
place vertices of G in random locations;
repeat M times
     calculate the force on each vertex;
     move the vertex c4 * (force on vertex)
draw graph on CRT or plotter.
```

Besides striving towards equal edge length and displaying symmetry one can argue that the graph layout also should strive to have an even vertex distribution for a more pleasing layout. The algorithm of Fruchterman and Reingold cover this by using a bit of a different physical model, seeing the vertices in a graph as atomic particles or as celestial bodies. Where the attractive forces are defined as[10]:

$f_a(d) = \frac{d^2}{k}$

Repuslive as:

$f_r(d) = \frac{-k^2}{d}$
Where d is the actual distance between two vertices and k is th optimal distance. K is defined as:

$k = C\sqrt{\frac{area}{numberofvertices}}$

Besides from this the algorithm also uses the notion of temperature as a refining step. This works so that when the algorithm improves the layout the adjustments becomes smaller from the last iteration of the algorithm. Follow in Figure 2.2 shows seudo code for the Fruchterman and Reingold algorithm.

As the graph size grows bigger, graphs with more than a few hundred vertices, a problem arises with the basic force-directed algorithms. The fact is that the used physical model has multiple local minima, and a graph produced with only a local minima can be much worse than would it be produced with the global minima. There have been developed algorithms to try and avoid local minima, such as the Hadany and Harel algorithm[6], which is based on a multi-level layout technique that works with graphs containing 15000 vertices.

In multi-level techniques the graph structure that is to be drawn is viewed in

```
area:= W * L; {W and L are the width and length of the frame}
G := (V, E); {the vertices are assigned random initial positions}
k := sqrt(area/|V|);
function f_a(x) := begin return x^2/k end;
function f_r(x) := begin return k^2/x end;
for i := 1 to iterations do begin
    {calculate repulsive forces}
    for v in V do begin
        {each vertex has two vectors: .pos and .disp
        v.disp := 0;
        for u in V do
            if (u ≠ v) then begin
            {δ is the difference vector between the positions of the two vertices}
                δ := v.pos − u.pos;
                v.disp := v.disp + (δ/|δ|) * f_r(|δ|)
            end
    end
    {calculate attractive forces}
    for e in E do begin
        {each edges is an ordered pair of vertices .vand.u}
        δ := e.v.pos − e.u.pos;
        e.v.disp := e.v.disp − (δ/|δ|) * f_a(|δ|);
        e.u.disp := e.u.disp + (δ/|δ|) * f_a(|δ|)
    end
    {limit max displacement to temperature t and prevent from displacement
outside frame}
    for v in V do begin
        v.pos := v.pos + (v.disp/|v.disp|) * min(v.disp, t);
        v.pos.x := min(W/2, max(−W/2, v.pos.x));
        v.pos.y := min(L/2, max(−L/2, v.pos.y))
    end
    {reduce the temperature as the layout approaches a better configuration}
    t := cool(t)
end
```

Figure 2.2. Fruchterman and Reingold algorithm

substructures where each substructure has less complexity that the whole. These substructures are then laid out in order from the most simple structure to the most complex one. For example bellow is Hadany and Haler's description of the multi-level method.

A natural strategy for drawing a graph nicely is to first consider an abstraction, disregarding some of the graphâs fine details. This abstraction is then drawn, yielding a ârooughâ layout in which only the general structure is revealed. Then the details are added and the layout is corrected. To employ such a strategy it is crucial that the abstraction retains essential features of the graph. Thus, one has to define the notion of coarse-scale representations of a graph, in which the combinatorial structure is significantly simplified but features important for visualization are well preserved. The drawing process will then âtravelâ between these representations, and introduce multi-scale corrections. Assuming we have already defined the multiple levels of coarsening, the general structure of our strategy is as follows:

1. *Perform fine-scale relocations of vertices that yield a locally organized configuration.*
2. *Perform coarse-scale relocations (through local relocations in the coarse representations), correcting global disorders not found in stage 1.*
3. *Perform fine-scale relocations that correct local disorders introduced by stage 2.*

2.1.2 Navigation through zooming

The way one zoom becomes a big part when navigating a graph, how one does this greatly affects the situational awareness. When navigating through a graph both global context and local detail are of importance. Global context is provide one to be able to orient oneself to be able to navigate through the graph. Though when zoomed out the local details are not on a high enough level to give any real information. So to get more detailed information one is needed to zoom in the graph to specific area, which is when a tunnel vision problem arises. Causing one to easier lose orientation and information of the overall dependencies when the context is lost.

FishEye view is a technique that address this problem of tunnel vision. One can compare the technique to a fisheye lens used by cameras for the creation of wide panoramic images. The techniques allows one to show high detail at focus while displaying less and less detail about information that are further away from focus, how much depending on how far away it lies. Extra effective this becomes if the data one works with has a clear structure so that one can cluster this data.

Next we show an implementation that was done from[18], there they assumes nodes is represented by squares and that no overlapping of nodes is present. It is based on data that is clustered so that a node contains a subset of different nodes. When zooming one zooms a node (cluster).

Notations:

- F_e - factor for growing nodes.
- F_s - factor for shrinking nodes.
- R_z - ratio of nodes to be zoomed with respect to their environment (length of parent node, L)
- r - ratio of nodes to be zoomed to the total length of all nodes.
- S_z - sum of length of all nodes to be zoomed.
- S_a - sum of lengths of all nodes.

Respectively variable is set as:

- $R_z = (1 - K_b) \times r + K_b$
- $F_s = \frac{(1-F\times\frac{S_z}{L})}{(1-\frac{S_z}{L})}$

Where Kb is a balance factor that controls the ratio between zoomed nodes the remaining nodes. A larger value on Kb results in a greater difference between size on zoomed nodes and remaining nodes.

Positioning:
For positions of nodes the x- and y-axis are divided up in segments by the boundaries of the nodes that are going to be zoomed. xi and xi' represents the positions before and after zoom. ls the length of a segment, and di the distance from xi to the left boundary of the segment containing xi. The xi are calculated by first sorting the segment list and then performing what shows in bellow for each node.

```
initialize xi' to the left boundary of parent node
for each segment to the left of xi
   if enlarging,  xi' = xi' + Fe ls
   else           xi' = xi' + Fs ls
for the segment containing xi
   if enlarging,  xi' = xi' + Fe di
   else           xi' = xi' + Fs di.
```

2.2 Two dimensional space

Next we will introduce methods used to visualize big networks in the two dimensional space. One benefit when using a layout in the two dimensional space is that one can get away from the problem of nodes overlapping each other.

2.2.1 BioFabric

BioFabric[13] is a method that uses a different approach to represent a graph than the traditional way. Where one represents nodes as a shape, like a circle or a rectangle, and edges as lines between nodes. Instead nodes are represented as one-dimensional horizontal lines and edges as one-dimensional vertical lines. These vertical lines starts at one of the horizontal lines (one specific node) and ends at another, representing a connection between these lines (nodes).

This different approach lets one get away from the problem of node and edge crossings. It guarantees no edge overlapping and no node overlapping.

The complexity that can arise whit many different methods when one handles updates of graphs is that the graph can alter its appearance heavily when only a few nodes are introduced. This problem exists in BioFabric as well, but because of adding one node is the same as adding one horizontal line and adding a edge equal to adding a vertical line this can help to not having such a big affect on the graphs appearance. Though how much it alters the graph is dependent on how many nodes are added and how many connections to other nodes are added.

As for how to layout the node and edges there are different approaches one can take. One basic approach is to do a breadth first traversal of the data to be displayed. Where neighbouring nodes are visited in the order determined by their degree, data structured by degree of nodes. Next follows an example of a way of assigning nodes and edges that uses this approach[13].

Node assignment:

1. Set row 1 as the next available row.
2. Find the highest degree node not yet processed, and assign it to the next available row. Make that row the current row; increment the next available row.
3. Take the node assigned to the current row and order its neighbors based upon their degree, highest degree first.
4. Traversing the neighbor nodes using that order, if the node has not yet been assigned, assign it to the next available row and increment the next available row.
5. Increment the current row. If a node has been assigned to that row, go to step 3. If not, go to step 2.

Edge assignment:

1. Set column 1 as the next available column. Make row 1 the current row c.
2. For current row c, get all the unassigned edges for the node in that row. Note that since we are not dealing with shadow links, all unassigned edges must connect to rows $\geq$ c.

3. For each row r $\geq$ c, create a set S of edges incident on c and r. Order these sets by increasing row number r, so that edges will be assigned in order of increasing length.

4. Iterating through the ordered list of sets, for each set S, order those edges in S based on lexicographic ordering of the link relation description, and assign them to the next available columns in this order; increment next available column appropriately. If there is a pair of directed edges with the same link relation description, downward links are assigned before upward links.

5. Increment the current row, and go to step 2.

Figure 2.3 is an example of a big network visualized with BioFabric using the basic approach.

One can also use approaches that try to group nodes based on similarity and difference between their connectivity. The way to represent similarity could be to use cosine similarity[23] or Jaccard similarity[24]. Figure 2.4 shows a network visualised using using similarity weights. Resulting in a less compact layout than the basic approach.

2.2.2 HivePlots

HivePlots is a visualisation algorithm that uses a number of radially oriented linear axes that has a coordinate system that is based on nodes properties. A networks nodes are layed out on these axes. Connecting nodes are shown with edges between them, visualized as curves between nodes. Figure 2.5 shows an example of a HivePlot.

Initially before the layout is made a number of structural parameters are calculated. Such as degree, flow, Page rank, clustering coefficient etc. Which parameters to use is up to the user to decide to fit with the network to be visualised. For example one would use the clustering coefficient to distinguish between hubs and clusters. Next these parameters are used to set up rules that are used to assign nodes to an axis and decide its coordinate. These rules are often boolean rules. Example of rules could be:

- Is the node a sink?
- Is the node a source?
- Clustering coefficient < 0.5?

If a HivePlot can be created with three axes this is preferred[11], laying the axis with a uniform radial distribution. This because with three axis you get a layout that is edge crossing free. In addition to this three axis makes it possible for each edge between each axis pair not to cross another axis. Though this is not restrained

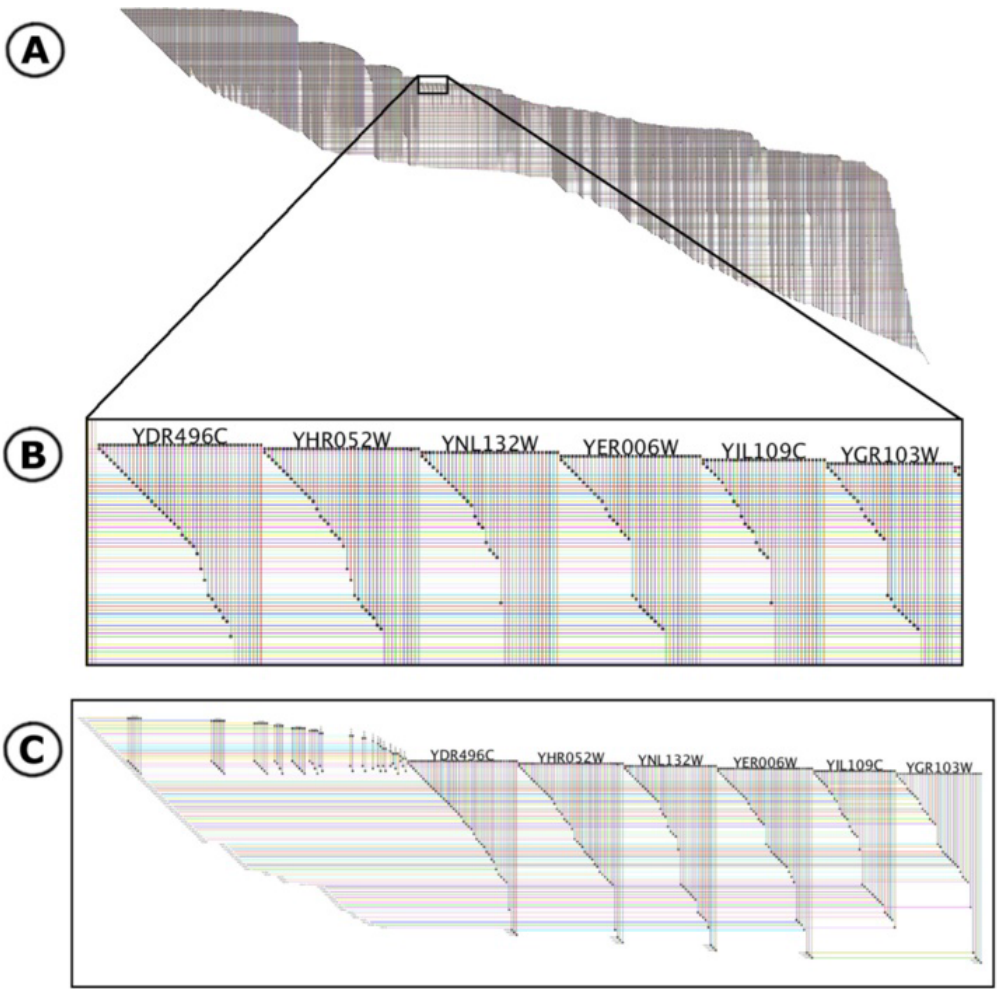


Figure 2.3. This is a depiction of the yeastHighQuality.sif data set [3-5] containing over 3000 nodes and 6,800 edges. The key feature of the BioFabric presentation is that nodes are depicted as horizontal lines, one per row; edges are presented as vertical lines, each arranged in a unique column. Note how the use of darker colors for rendering edges and lighter colors for rendering nodes insures that the former stand out despite the crossover. A) The view of the full network, laid out with the default algorithm. B) Detail of network shown boxed in network A, which highlights one advantage of the BioFabric presentation technique: similarities, and differences, in the connectivity of different nodes are immediately apparent. C) The six nodes and first neighbors depicted in a subset view, where all extra space has been squeezed out, creating a compact presentation that still retains all the relative positioning from the full view. Note how the full inventory of edges incident on the six nodes also includes those on the left originating from higher node rows.

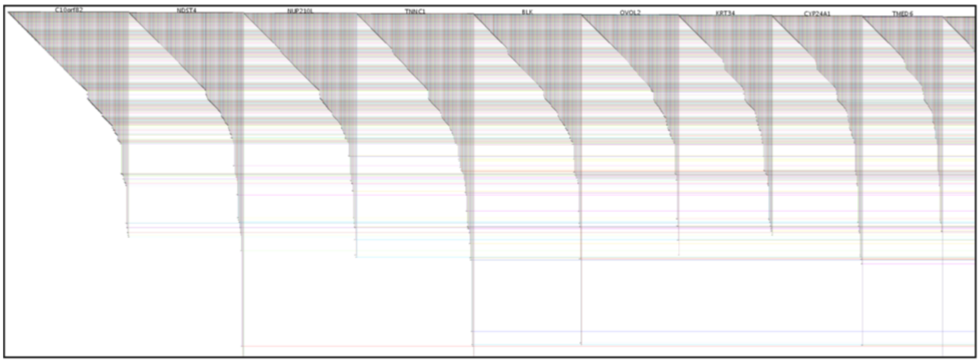


Figure 2.4. Layout that tries to place nodes with similar connectivity next to each other in the linear ordering of nodes.

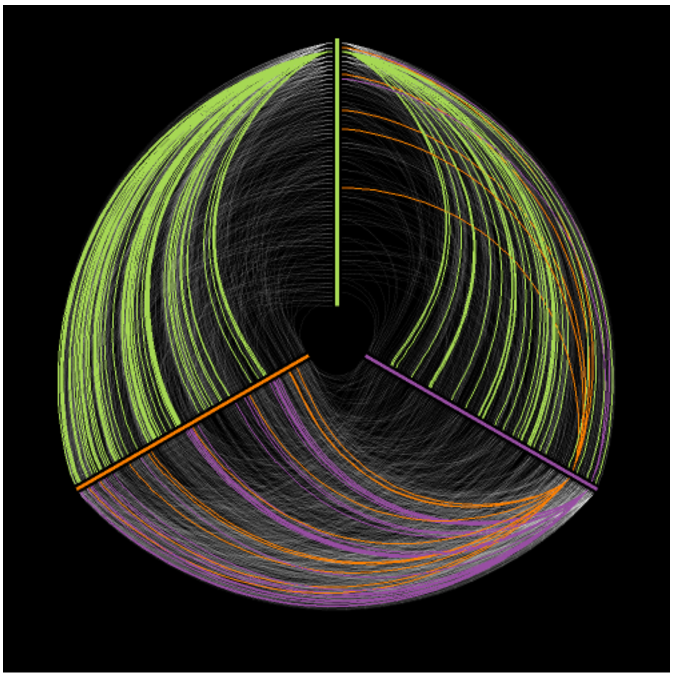


Figure 2.5. Example of a HivePlot containing 2500 vertices and 5900 edges.

to only three axis, it can be hard to partition nodes to axis so that nodes on axis are only connected to neighbouring axes.

2.2.3 TreeMap

TreeMap is a technique to present graphs in sequences of nested boxes[7]. TreeMap requires the data to be hierarchy structured as a tree. Figure 2.6 shows an example. The size of individual boxes becomes significant in a TreeMap layout, where the user specifies how they should grow. As an example, a TreeMap that has data that represents a file system hierarchy, the size of a box could be proportional to the size of the file it represents.

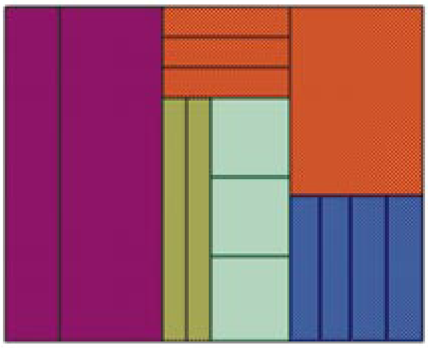


Figure 2.6. Tree-map: rectangles with color belong to the same level of the (tree) hierarchy. (Adapted from Johnson and Schneiderman [72]).

2.3 Three dimensional space

In the hope of acquire more space for the layout of a network one can take the approach to go from 2D to a 3D environment. In other words one can strive to get away from the euclidean space to another space that provides more space. An important aspect that follows going to a 3D view is that the system should be nagivatable. This because in a 3D view node and edge occlusions is bound to happen. Being able to change one view by navigating one can find a view of ones perspective that is without occlusions.

2.3.1 GerbilSphere

There have been studies on 2D vs 3D user interfaces that have shown that in many cases 2D exceeds 3D. Though the more space in 3D is still compelling. GerbilSphere is an inner sphere 2D system that tries to use the benefits from both a 2D approach as well as a 3D approach.

GerbilSphere works in a way that it lets the observer be able to places them self inside a sphere while projecting the network on the surface of the sphere. As part of the layout, GerbilSphere uses an extended version of the Fruchterman and Reingold force-directed algorithm to apply to the three dimensional space. Though this is not enough to work on the surface of a sphere. To apply the forces to the surface of a sphere, GerbilSphere uses a algorithm described by Kobeourov and Wampler[9]. Figure 2.7 shows the seudo code for their algorithm

For more technical information about the data structure and how there layout algorithm works see [19].

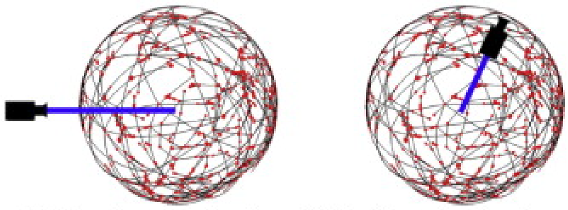
Zooming in GerbilSphere is viewed as having a world camera attached to one end of a tether and having the other end attached to the center of the sphere. Zooming in and out can be seen as moving the world camera along this tether. Figure 2.8 shows when zoomed out respectively zoomed in.

GerbilSphere implements a 2 1/2D interface, advocated by Ware[26]. When a user is positioned inside the sphere and zoom in, the part of the network when zoomed in will be visualized on a flat 2D surface, as seen in Figure 2.9. And also while zoomed in the nodes can appear as a 3D sphere. Still one can zoom out one can still have there point of interest in view and gain more global context of the network. Last one can zoome out enough to place the view outside the sphere. Seeing the network on a 3D sphere.

```
Input: Set of vertices and set of edges
Output: New vertex positions
k = sqrt((frameWidth * frameHeight)/|V|)
function f_a(x) = begin return x^2 / k end;
function f_r(x) = begin return k^2 / x end;
//Each vertex has two vectors: .pos and .disp
while temperature > 0 do
Repulsions      foreach Vertex v ∈ V do
                    tDisp = 0;
                    nearGridsList = getNearGridsList(v.myGridNum);
                    nearVertsList =
                    getNearVertsListFromGrids(nearGridsList);
                    foreach Vertex u ∈ nearVertsList do
                        vDisp = v.pos - fromSphereToTangentPlane(u.pos);
                        tDisp = tDisp + vDisp.norm() * f_r(vDisp.len());
                    end
                    v.disp = v.disp + tDisp;
                end
                //Edges have reference vectors fNode and tNode ∈ V
Attractions     foreach Edge e ∈ E do
                    diff = e.fNode.pos -
                    fromSphereToTangentPlane(e.tNode.pos);
                    e.fNode.disp = e.fNode.disp - diff.norm() * f_a(diff.len());
                    diff = e.tNode.pos -
                    fromSphereToTangentPlane(e.fNode.pos);
                    e.tNode.disp = e.tNode.disp - diff.norm() * f_a(diff.len());
                end
Updates         foreach Vertex v ∈ V do
                    v.pos = v.pos + v.disp.norm() * min(v.disp,temperature);
                    v.pos = fromTangentPlaneToSphere(v.pos);
                    v.disp = 0 ;
                end
                temperature = cool(temperature);
end
```

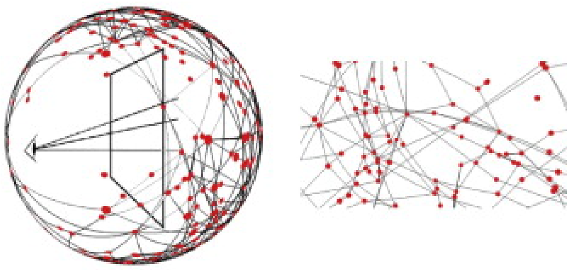
Algorithm 1: Spherical Volume Grid Based algorithm

Figure 2.7. Spherical volume grid based



(a) Negative tether length (b) Positive tether length

Figure 2.8. Spherical volume grid based



(a) InnerSphere Visualization (b) User's Perspective

Figure 2.9. Spherical volume grid based

2.3.2 Hyperbolic space

The hyperbolic space has the property that it has more room compared to the familiar euclidean space[17]. [27] states that the fifth postulate in the Euclidean plane geometry can be formulated as:

> through a given point, not on a given line, one and only one line can be drawn which does not

As in the hyperbolic plane geometry they introduce the Characteristic Postulate:

> Through a given point, not on a given line, more than one line can be drawn not intersecting the given line.

Moreover two lines that are parallel in the euclidean space are always the same distance apart. As in the hyperbolic space parallel lines are not equidistant. For instance two parallel lines in the hyperbolic space that do not intersect can be seperated by increasing distance the further away one moves from the origin. Figure 2.10 shows this compared to the euclidean geometry.

Normally to make use of the hyperbolic space, to use the extra room, one goes about to perform an layout algorithm in the hyperbolic plane or space and then display the results in the Euclidean plane or space. Some models to do this have been created. Best known are the Klein and the PoincarÃ© models[7].

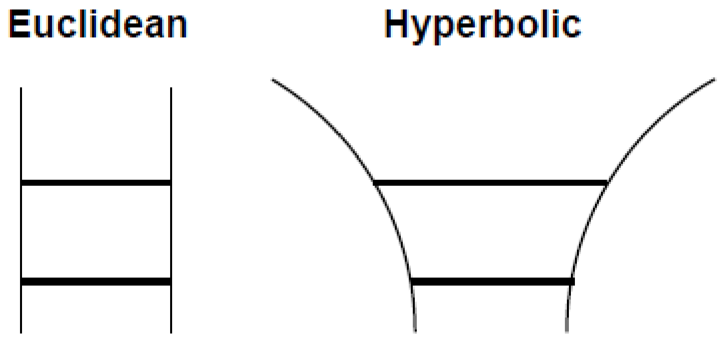


Figure 2.10. Parallel lines in euclidean space are always the same distance apart. In hyperbolic space the distance between two lines that never meet does indeed change. Here we show two geodesics which never meet but are not equidistant: the further they extend away from the origin, the more room there is between them.

When using hyperbolic layout of networks trees are of the used for the structure of the data. As cone trees, see Figure 2.11. Or as in [17] that lays out nodes on a hemisphere, see Figure 2.12. [17] visualise graphs in the 3 dimensional hyperbolic space placeing the trees inside a sphere. Exploiting the property that the amount of space covered by a sphere in the 3 dimensional hyperbolic space increases exponentially with respect to the radius of the sphere, rather than polynomially. Figure 2.13 shows an example layout in the 3 dimensional hyperbolic space from.

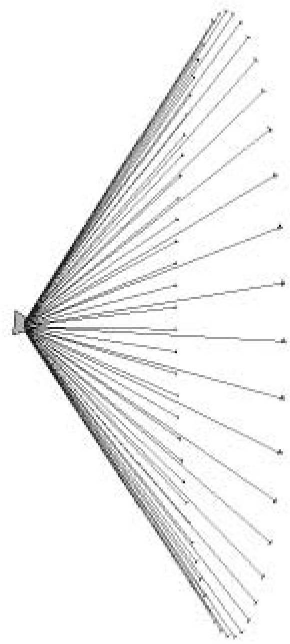


Figure 2.11. 54 nodes. Cone tree layout along the circumference of a circle

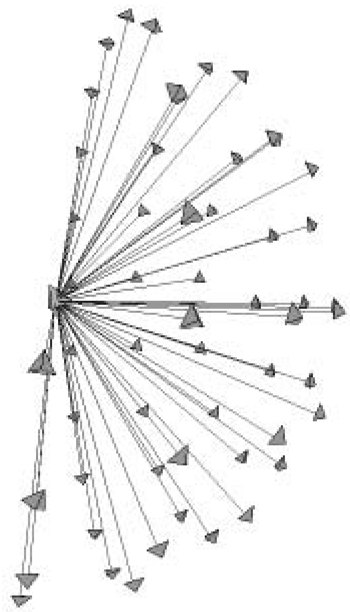


Figure 2.12. 54 nodes. H3 layouton the surface of the spherical cap

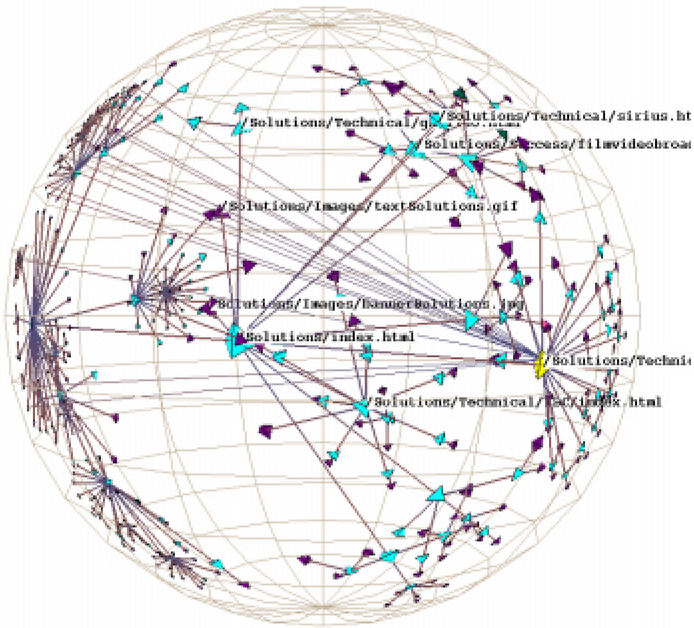


Figure 2.13. Link structure of aWeb site laid out in 3D hyperbolic space. The nodes represent documents,which are colored according to MIME type: HTML is cyan, images are purple, and so on.

Chapter 3

Method

3.1 Selection

How I've selected which methods to be investigated.

3.2 Implementation

Describe how I will implement a few visualization techniques and some prerequisites.

3.3 Tests

How tests been set up and purpose.

Chapter 4

Results

Show Implementation
Present results from performed tests.

Chapter 5

Discussion and Conclusions

Give an account of whether an optimal choice of visualisation technique exists. Go through the results and point to what's a good choice of visualization technique under what circumstances.

References

[1] T. Buring, J. Gerken, and H. Reiterer. Ipsep-cola: An incremental procedure for separation constraint layout of graphs. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):821–828, Sept 2006.

[2] Goodrich Duncan. Planar orthogonal and polyline drawing algorithms. In *Handbook of Graph Drawing and Visualization*, pages 223–246, October 2013.

[3] Peter Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[4] M.R. Endsley. Situation awareness global assessment technique (sagat). In *Aerospace and Electronics Conference, 1988. NAECON 1988., Proceedings of the IEEE 1988 National*, pages 789–795 vol.3, May 1988.

[5] T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Softw. - Pract. Exp.*, 21(11):1129–1164, 1991.

[6] R. Hadany and D. Harel. A muli-scale algorithm for drawing graphs nicely. *Discrete Applied Mathematics*, 113(1):3–21, 2001.

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

[8] U. Kang and C. Faloutsos. Beyond 'caveman communities': Hubs and spokes for graph compression and mining. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 300–309, Dec 2011.

[9] S.G. Kobourov and K. Wampler. Non-euclidean spring embedders. *IEE Transactions on Visualization and Computer Graphics*, 11, 2005.

[10] Stephen G. Kobourov. Force-directed drawing algorithms. In *Handbook of Graph Drawing and Visualization*, pages 383–408, October 2013.

[11] Martin Krzywinski, Inanc Birol, Steven JM Jones, and Marco A Marra. Hive plotsârational approach to visualizing networks. *Briefings in Bioinformatics*, 13(5):627–644, 2012.

[12] Zhiyuan Lin, Nan Cao, Hanghang Tong, Fei Wang, U. Kang, and Duen Horng Chau. Demonstrating interactive multi-resolution large graph exploration. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, pages 1097–1100, Dec 2013.

[13] William Longabaugh. Combing the hairball with biofabric: a new approach for visualization of large networks. *BMC Bioinformatics*, 13(â1):275, 2012.

[14] Joe Marks and Stuart Shieber. The computational complexity of cartographic label placement. Technical report, 1991.

[15] C. Mikkelsen, J. Johansson, and M. Cooper. Visualization of power system data on situation overview displays. In *Information Visualisation (IV), 2012 16th International Conference on*, pages 188–197, July 2012.

[16] Christopher Mueller, Benjamin Martin, and Andrew Lumsdaine. A comparison of vertex ordering algorithms for large graph visualization. *2007. APVIS '07. 2007 6th International Asia-Pacific Symposium on Visualization*, pages 141–148, 2007.

[17] T. Munzner. H3: laying out large directed graphs in 3d hyperbolic space. In *Information Visualization, 1997. Proceedings., IEEE Symposium on*, pages 2–10, Oct 1997.

[18] Doug Schaffer, Zhengping Zuo, Saul Greenberg, Lyn Bartram, John Dill, Shelli Dubs, and Mark Roseman. Navigating hierarchically clustered networks through fisheye and full-zoom methods. *ACM Trans. Comput.-Hum. Interact.*, 3(2):162–188, June 1996.

[19] David S. Shelley and Mehmet Hadi Gunes. Gerbilsphere: Inner sphere network visualization. *Computer Networks*, 56(3):1016 – 1028, 2012. (1) Complex Dynamic Networks (2) {P2P} Network Measurement.

[20] Jaroslav Snajberk, Lukas Holy, and Premek Brada. Aiva vs uml: Comparison of component application visualizations in a case study. *Information Visualisation (IV), 2012 16th International Conference on*, pages 54–61, 2012.

[21] G. Eick Stephen. Aspects of network visualization. *Computer Graphics and Applications, IEEE*, 16(2):69–72, 1996.

[22] D.R. Tesone and J.R. Goodall. Balancing interactive data management of massive data with situational awareness through smart aggregation. In *Visual Analytics Science and Technology, 2007. VAST 2007. IEEE Symposium on*, pages 67–74, Oct 2007.

[23] Wikipedia the free encyclopedia. Cosine similarity. http://en.wikipedia.org/wiki/Cosine_similarity. [Online; Accessed 28-November-2014].

[24] Wikipedia the free encyclopedia. Jaccard index. http://en.wikipedia.org/wiki/Jaccard_index. [Online; Accessed 28-November-2014].

[25] William T. Tutte. How to draw a graph. *Proc, London Math. Society*, 13(52):743–768, 1963.

[26] C. Ware. Designing with a 2 1
2 d attitude. *Information Design Journal*, 10, 2001.

[27] Harold E. Wolfe. Introduction to non-euclidean geometry. *Dryden Press*.

[28] Harold E. Wolfe. Introduction to non-euclidean geometry. 1945.