



**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 B  lter  
Project commissioned by: Scania  
Supervisor at Scania: Magnus Kyll  rd



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background . . . . .	1
1.2	Arising problems with growing data . . . . .	1
1.2.1	Edge and node crossing . . . . .	1
1.2.2	Situation awareness . . . . .	2
1.3	This thesis . . . . .	2
<b>2</b>	<b>Visualization techniques and their theory</b>	<b>3</b>
2.1	Common techniques . . . . .	3
2.1.1	Force-Directed . . . . .	3
2.1.2	Navigation through zooming . . . . .	6
2.2	Two dimensional space . . . . .	7
2.2.1	BioFabric . . . . .	7
2.2.2	HivePlots . . . . .	7
2.2.3	TreeMap . . . . .	7
2.3	Three dimensional space . . . . .	7
2.3.1	BioSphere . . . . .	7
2.3.2	Hyperbolic space . . . . .	7
	<b>Appendices</b>	<b>7</b>
<b>A</b>	<b>RDF</b>	<b>9</b>



# 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, etc.

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 node-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 an 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 with 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 overlap, crossing each 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 to minimize node- and edge crossings.

### 1.2.2 Situation awareness

Human and psychology factors play a role when visualizing a network, situation awareness is a term in this aspect. Endsley[ref] 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 consideration when choosing visualisation technique.

## 1.3 This thesis

This thesis revolve 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 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 are based on some fundamental technique or concept. Two major aspects are 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 generated. Second is the aspect of how one is to navigate a graph when it has 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 rely 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[ref].

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[ref] and is based on barycentric representation[ref]. Though the more commonly used algorithms such as Eades[ref] and Fruchterman and Reingold[ref] both

relay on spring forces similar to those in Hooke's law[ref]. 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 aglo-rithm[ref]:



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 lay-out 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:

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

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  $c4 * (\text{force on vertex})$ 
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 Rein 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:

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

Repulsive as:

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



## 2.1. COMMON TECHNIQUES

Where  $d$  is the actual distance between two vertices and  $k$  is the optimal distance.  $K$  is defined as:

$$k = C \sqrt{\frac{\text{area}}{\text{number of vertices}}}$$



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 become smaller from the last iteration of the algorithm. Follow in Figure(2.1) the pseudo code for the Fruchterman and Reingold algorithm is shown[ref].

---

```

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 fa(x) := begin return x2/k end;
function fr(x) := begin return k2/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|) * fr(|δ|)
            end
        end
    end
    {calculate attractive forces}
    for e in E do begin
        {each edge is an ordered pair of vertices .v and .u}
        δ := e.v.pos - e.u.pos;
        e.v.disp := e.v.disp - (δ/|δ|) * fa(|δ|);
        e.u.disp := e.u.disp + (δ/|δ|) * fa(|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(2.1)

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[ref], 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 substructures where each substructure has less complexity than 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 graph's fine details. This abstraction is then drawn, yielding a rough 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 travel 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 zooms 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



## 2.2. TWO DIMENSIONAL SPACE

easier lose orientation and information of the overall dependencies when the context is lost.

FishEye view is a technique that **adres** 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 effecti  is becomes if the data one works with has a clear structure so that one can cluster this data.

The algorithm[ref] that is being introduced next assumes nodes represented by squares and that no overlapping of nodes is present.

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 enviroment (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.

Before zooming operation we initialise  $R_z$  and  $r$  to:

$$R_z = F_e \times \frac{S_z}{L} \quad 0 < R_z \leq 1$$
$$r = \frac{S_z}{S_a} \quad 0 < r \leq 1$$

## 2.2 Two dimensional space

### 2.2.1 BioFabric

### 2.2.2 HivePlots

### 2.2.3 TreeMap

## 2.3 Three dimensional space

### 2.3.1 BioSphere

### 2.3.2 Hyperbolic space



# Appendix A

## RDF

And here is a figure

**Figure A.1.** Several statements describing the same resource.

that we refer to here: A.1