University of Glasgow | School of Computing Science

Visualisation of Large Networks in a Browser

Ben Jackson

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

Level 4 Project — October 3, 2016

Abstract

The purpose of this research is to identify different techniques that existing software uses to visualise extremely large networks in a browser, and then compare them and review which methods are the most successful. This is good because ??. I did it by doing ??. I found out ??. Conclusion.

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name: _____ Signature: _____

Contents

Chapter 1

Introduction

Rendering large networks of data (tens or hundreds of thousands of nodes and edges) in a browser is a frequent problem encountered in visualisation software due to limits in browser performance and the ability to render the information in a meaningful way the user can make sense of it. This project aims to analyse many different software packages available and outline different approaches which can be taken to help minimize the clutter and performance required to display these large networks.

1.1 Context

Currently, the problem is that far too much data is passed to the client from the server, both requireing a lot of bandwidth and processing power, and once the data is finally rendered, the result is a mass of nodes of which no useless information can be taken from.

The ideal end result of the project would involve taking the data from the server that would normally be graphed, and instead analysing that data, and then passing a modified version of the data to the client. This could be in the form of an image, removing unnecessary nodes / edges, or node bundling / edge bundling. This would result in the information being far more useful to the client, with them being able to make useful decisions based on the network presented to them, as opposed to before where they were shown a huge mass of nodes that could not easily deciphered. The end result would also ideally reduce load times / CPU / RAM required.

useful?

Visualisation of large networks is challenging for a number of reasons, incuding the amount of processing power and memory required in order to produce the network or get an interactive visualisation of it, and the difficulty of displaying the network clearly so a user can get useful information from it. Limiting the research to visualising the data in a browser adds another level of complexity as browsers perform far more poorly than servers or average computers and there are far fewer choices of software available that will run in a browser.

1.2 Aims and Objectives

1.3 Achievements

Chapter 2

Background

2.1 Visualisation

2.2 Network Visualisation

2.3 Challenges of Browser Visualisation

2.4 Current Graphing Software

i should write somewhere about how difficult it was to find out about testing software systems as a developer.

Chapter 3

Experiment 1

3.1 Aim

((need to talk about JS and browser related stuff)) The goal of the first experiment was to do background research into several different types of graphing software, and as a result of the research, decide what paths the project could take. The software would be analysed for several characteristics, both related to general graphing of networks, and graphing of specificaly huge networks. A list of parameters were chosen to evaluate the software against as follows:

- Ease of download and installation of software
- Quality of documentation
- How easy it was to make a simple network using the software
- The ability to import and/or export a file into the system
- Render time for multiple sizes of graphs
- What features the software contained, including features are specific to displaying very large graphs
- Whether the software had the ability to change graphical options
- If multivariate graphs were supported

The software that was compared against the above parameters was all of the software found in the initial research of graphing software, which was:

- GUESS
- SNAP
- Gephi
- GraphViz
- Tulip
- D3.js
- InfoViz

3.2 Methodology

Using the list of paramaters above, a spreadsheet was made, in which every piece of software had a different sheet, and each sheet had all of the parameters each piece of software was to be compared against. The final list of parameters, split into as many sections as possible, all of which aimed to be as binary as possible or times, was:

- Date that the current version was created
- Ease of download (for most software this was a given but occassionally it was more difficult)
- Installation necessary
- Documentation
  - Easy to find
  - Easy to understand
  - Comprehensive
- Time until a hardcoded graph could be displayed
- A file could be loaded into the system
- If the file is altered, could it be exported to a file
- For different sizes, how long it took to render the graph
- If the software allowed for:
  - Zooming
  - Panning
  - Scrolling
- A graph could be dynamic - allowing data taken at multiple points over time to be viewed
- If the software included features for displaying huge graphs, such as:
  - Edge Bundling
  - Node bundling
  - Alternative layout options
- A graph could be multivariate
- There was the possibility of changing graphical settings for the graph
- A graph could be displayed in 3D
- Any other points of interest

Each piece of software was to be evaluated against this criteria, with the goal of both finding out which pieces of software were better, over many parameters, and also to become more familiar with graphing software used in industry.

3.3 Data

(need to incorperate that I expected APIs but I got full software packages which are fairly useless for visualising in a browser)

Upon beginning to run the experiment, it became clear that several pieces of software were not useful candidates for a number of reasons. The first piece of software, GUESS, turned out to be abandonware, last being developed in 2007 and never left beta, and hence wasn't tested thoroughly as a result of this, along with documentation being very poor and hard to find.

The next piece of software, SNAP, was not fit for purpose. It was very powerful but had the purpose solely of analysing and manipulating graphs, as opposed to also visualising them. As a result, many of the characteristics being tested were not relevant for SNAP as there was no visualisations and hence no zooming / panning / scrolling / huge graph visualisation support / render times (NEED TO CHANGE THIS), although it was still examined: reading it's documentation and going through all the sample code snippets, and understandin ghow they worked. The reason this was done was because, depending on the direction the project took, graph manipulation may well be used in order to bundle edges or nodes in order to increase performance of huge networks.

Next, Gephi was tested successfully. It was recent, easy to use, and documentation was thorough. After about half an hour I was comfortable with how it worked, able to make simple graphs, and to import graphs from many formats (CSV, GDF, QML, GraphML, net, etc.) and export to all of them, along with as a PNG, PDF or SVG. The time to render graphs of a few thousand nodes was less than half a second, less than a thousand nodes was instant. It supported zooming, panning and scrolling, and having data being dynamic was a possibility. There was also multiple heavily customisable layout options, many of which would suit huge networks more than others, although there was no additional support for huge networks, other than the system being very efficient (bundling was not supported, along with partial viewing of the data). Additionally, there was the ability to change graphical settings, have the data graphed in 3D, and have a multivariate graph.

Graphviz was difficult to test, being a collection of software as opposed to a single package. Most of the packages were not fit for purpose, often displaying data in charts, for example. The package I ended up testing, sfdp, was fairly disappointing. Although it fitted most of the criteria, it turned out to be very focused on graphing far smaller networks with little support for anything of a few thousand nodes or more. The documentation was, although by no means great, fine, and the software had the ability to import and export as a very large amount of file formats. It also supported zooming/panning/scrolling, but along with getting decreasing in speed a lot faster than other pieces of software (three thousand nodes took several seconds), the UI clearly was not build around supporting large networks, with information becoming very unclear quickly. ((IMAGE)). There was also no explicit support for visualising huge networks.

Tulip - TO DO

D3.js is a an API over full software which is what I hoped more of the software would be like. As a result it is far more flexible but requires far more setup to be done by the user. The documentation is good although complicated, and importing and exporting is easily possible via JSON, and other file formats with slightly greater difficulty. It supports zooming / panning / scrolling as expected and was fairly performant, displaying a few thousand nodes in about a second. Also, there are many ways to make it effective at showing huge networks, but this would require research into many different packages and potentially writing code to do it.

InfoViz was really pretty similar to D3, but less widely used. (write more here).

3.4 Results

In general the software was not what was expected and hence the experiment was not a huge success. Far more of the software was standalone packages than expected, and other issues were also come across (such as software being deprecated or for a different purpose).

3.5 Conclusion

Look into D3.js or graph manipulation and then D3 / InfoViz.

Chapter 4

Experiment 2

Chapter 5

Conclusion

Appendices

Appendix A

My first appendix

This is what was interesting.