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

We show how to produce a level 4 project report using latex and pdflatex using the style file l4proj.cls

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) 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 Aim

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.

1.1.1 A subsection

The quick brown fox jumped over the lazy dog.

Chapter 2

The Fox and Dog

The quick brown fox jumped over the lazy dog.

2.1 The Fox Jumps Over

The quick brown fox jumped over the lazy dog. The quick brown fox jumped over Uroborus (Figure 2.1). The quick brown fox jumped over the lazy dog.

> The quick brown fox jumped over the lazy dog.
>
> The quick brown fox jumped over [**?**] the lazy dog. The quick brown fox jumped over the lazy dog.

2.2 The Lazy Dog

The quick brown fox jumped over the lazy dog.

Appendices

Appendix A

Running the Programs

An example of running from the command line is as follows:

```
> java MaxClique BBMC1 brock200_1.clq 14400
```

This will apply $BBMC$ with $style = 1$ to the first brock200 DIMACS instance allowing 14400 seconds of cpu time.

Appendix B

Generating Random Graphs

We generate Erdós-Rënyi random graphs $G(n, p)$ where n is the number of vertices and each edge is included in the graph with probability p independent from every other edge. It produces a random graph in DIMACS format with vertices numbered 1 to n inclusive. It can be run from the command line as follows to produce a clq file

```
> java RandomGraph 100 0.9 > 100-90-00.clq
```