



Terms and Conditions of Use of Digitised Theses from Trinity College Library Dublin

Copyright statement

All material supplied by Trinity College Library is protected by copyright (under the Copyright and Related Rights Act, 2000 as amended) and other relevant Intellectual Property Rights. By accessing and using a Digitised Thesis from Trinity College Library you acknowledge that all Intellectual Property Rights in any Works supplied are the sole and exclusive property of the copyright and/or other IPR holder. Specific copyright holders may not be explicitly identified. Use of materials from other sources within a thesis should not be construed as a claim over them.

A non-exclusive, non-transferable licence is hereby granted to those using or reproducing, in whole or in part, the material for valid purposes, providing the copyright owners are acknowledged using the normal conventions. Where specific permission to use material is required, this is identified and such permission must be sought from the copyright holder or agency cited.

Liability statement

By using a Digitised Thesis, I accept that Trinity College Dublin bears no legal responsibility for the accuracy, legality or comprehensiveness of materials contained within the thesis, and that Trinity College Dublin accepts no liability for indirect, consequential, or incidental, damages or losses arising from use of the thesis for whatever reason. Information located in a thesis may be subject to specific use constraints, details of which may not be explicitly described. It is the responsibility of potential and actual users to be aware of such constraints and to abide by them. By making use of material from a digitised thesis, you accept these copyright and disclaimer provisions. Where it is brought to the attention of Trinity College Library that there may be a breach of copyright or other restraint, it is the policy to withdraw or take down access to a thesis while the issue is being resolved.

Access Agreement

By using a Digitised Thesis from Trinity College Library you are bound by the following Terms & Conditions. Please read them carefully.

I have read and I understand the following statement: All material supplied via a Digitised Thesis from Trinity College Library is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of a thesis is not permitted, except that material may be duplicated by you for your research use or for educational purposes in electronic or print form providing the copyright owners are acknowledged using the normal conventions. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

Visualising Small World Graphs using Agglomerative Clustering around Nodes of Interest

Fintan McGee

August 23, 2013

A thesis submitted to the University of Dublin, Trinity College in candidacy for the degree of Doctor of Philosophy in Computer Science.



Thesis 10266

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work. I agree that Trinity College Library may lend or copy this thesis upon request.



Summary

The difficulty of visualising large graphs lies not just in processing power and display size but in the inherent visual complexity of a large data-set, as the noise and clutter from large numbers of nodes and an order of magnitude more of edges negatively impacts the comprehensibility of any visualisation. Small world graphs are a classification of graph that occurs frequently in models of real world networks such as computer systems and social networks. The overall objective of our research is to allow users to get a better comprehension of the relationships between data entities in the visualisation of real world systems.

The layout of a graph has a significant impact on its comprehensibility. Automated layouts may be used to cope with graphs containing a large numbers of nodes and edges. However, this may only provide a globally optimised layout, and may not necessarily focus on the nodes which might be of interest of the end user.

We introduce a novel approach for making large small world graphs more comprehensible by decomposing the graph into clusters, using an agglomerative clustering process based around user defined nodes of interest. We propose using clustering coefficient, a prominent feature of small world graphs that relates to local graph structure, as a heuristic to guide the agglomerative clustering process. We validate the effectiveness of our chosen heuristic experimentally against a large range of graphs and in comparison to other clustering heuristics.

We extend our clustering to generate a clustering hierarchy which reflects the clusters around the user's nodes of interest at multiple levels. We utilise this hierarchy to perform a multilevel layout providing users with a view of the graph, which reflects the relationships between the clusters defined by the user's nodes of interest. We also utilise our clustering hierarchy for *edge bundling*, a recently popular cluster reduction technique, in the graphs produced by our layout.

We provide an empirical user evaluation of edge bundling, the first of its kind to our knowledge. Our results show that while edge bundling negatively impacts understanding of low level node connectivity it does aid in the identification of higher level trends between clusters. We have also extended edge bundles into three dimensions for stereoscopic viewing and evaluated it empirically. Somewhat surprisingly, our results show that the stereoscopic viewing of edges with three dimensional depth offers no significant benefit to users.

Acknowledgements

First and foremost I'd like to thank my family for their unconditional support and encouragement throughout months and years of my PhD journey.

I believe the role of supervisor is very important and has a big influence on the path of a PhD. I was very fortunate to have an excellent supervisor, John Dingliana. Without his advice, support and his seemingly endless patience the submission of this PhD would not have been possible.

My colleagues in GV2 also played a vital role in this work. I am grateful to them all for their support, aid in developing experiments, experiment participation, proof reading, advice and general good company. Special thanks go to Ludovic Hoyet and Sophie Joerg for consistent moral support throughout and excellent advice and help in the submission of papers.

Contents

List of Figures	xii
Chapter 1 Introduction	1
1.1 Motivation	3
1.2 Key Concepts	4
1.3 Contribution	5
1.4 Scope	5
1.5 Related Publications	6
1.6 Thesis Layout	6
Chapter 2 Background and Related Work	9
2.1 Graphs	9
2.1.1 Graph Visualisations	10
2.1.2 Small World Graphs	10
2.1.3 Graph Centralities	15
2.1.4 Graph Edge Density	17
2.2 Graph Clustering	18
2.2.1 Clustering Overview	18
2.2.2 Clustering Approaches	19
2.2.3 Clustering Evaluation	21
2.3 Graph Layout	24
2.3.1 Force directed layouts	24
2.3.2 Fruchterman Reingold Layout	24
2.3.3 Multilevel Layouts	26
2.3.4 Hierarchy Based	28
2.3.5 Algebraic Approaches	29
2.3.6 Circular layouts	29
2.4 Graph Visualisation Evaluation	32
2.4.1 Evaluation Graphs	33
2.4.2 Graph Aesthetics	36
2.4.3 Evaluating User Performance	37

2.5	Edge Routing	41
2.5.1	Edge Bundling	41
2.6	Three Dimensional Stereoscopic Vision and Graphs	47
2.6.1	Stereoscopic Display of Graphs	47
2.6.2	Stereo rendering	49
2.6.3	Three dimensional layout of graphs	52
2.7	Implementation of Graph Rendering and Processing	52
2.7.1	Graphics Hardware	53
2.7.2	GPU Processing	53
Chapter 3 Agglomerative Clustering around Nodes of Interest		55
3.1	Motivation for Clustering	56
3.2	Related work	56
3.2.1	Clustering	57
3.2.2	Clustering Evaluation Metrics	57
3.2.3	Edge Density	58
3.3	Calculating Average Local Clustering Coefficient	58
3.4	Initial Investigation of Clustering Coefficient	59
3.4.1	Introduction	59
3.4.2	Initial Clustering Algorithm	60
3.4.3	Evaluation Approach	61
3.4.4	Results	62
3.4.5	Conclusions of our Initial Investigation	72
3.5	Maximising Clustering Coefficient Approach	73
3.5.1	Clustering Approach	73
3.5.2	Chosen Heuristics	73
3.5.3	Initial Cluster Set Up	75
3.5.4	Assignment of Nodes to Clusters	76
3.6	Clustering evaluation	77
3.6.1	Evaluation Graphs	77
3.6.2	Results and Analysis	78
3.6.3	Comparison with Edge Betweenness Centrality Clustering	86
3.6.4	Evaluation Conclusions	89
3.7	History of Infoviz Data-Set Example	90
3.7.1	Clustering Approach	93
3.7.2	Clustering Evaluation	93
3.8	Conclusions and Future Work	99

Chapter 4	Graph Layout	101
4.1	Motivation	102
4.2	Related Work	103
4.3	Circular Layout of Clusters	103
4.3.1	Initial Node Ordering	104
4.3.2	Cluster Rotation Implementation	106
4.3.3	Circular sifting	107
4.4	Layout and Hierarchy Generation	108
4.4.1	Generating a clustering hierarchy	108
4.4.2	Hierarchical Clustering Layout	111
4.4.3	Multilevel Cluster Layout	111
4.5	Results	113
4.5.1	Hierarchical Layout	114
4.5.2	Multilevel Layout	116
4.5.3	Hierarchical Edge Routing	116
4.6	Conclusions and Future Work	118
Chapter 5	Edge Routing	121
5.1	Edge Bundling Evaluation	122
5.1.1	Evaluation Motivation	122
5.1.2	Previous Experimental Approaches	123
5.1.3	Evaluation Hypotheses	124
5.1.4	Experiment Bundling Approach	125
5.1.5	Experiment Graphs	125
5.1.6	Experiment Methodology	128
5.1.7	Results	131
5.2	Stereoscopic Three Dimensional Edge Bundling	139
5.2.1	Motivation	139
5.2.2	Edge Routing in three Dimensions with stereoscopic viewing . . .	140
5.2.3	Defining Curve Depth	142
5.3	Three Dimensional Bundling Experimental Evaluation	143
5.3.1	Hypothesis	143
5.3.2	Choice of graphs and experiment factors	144
5.3.3	Initial Experimental Results	149
5.3.4	Follow On Path Tracing Experiment	155
5.3.5	Results	158
5.4	Conclusions	160
5.4.1	Edge Bundling	160
5.4.2	Three Dimensional Stereoscopic Edge Bundling	161

<i>CONTENTS</i>	xi
Chapter 6 Conclusions and Future Work	163
6.1 Conclusions	163
6.1.1 Graph Clustering	163
6.1.2 Graph Layout	164
6.1.3 Edge Routing	164
6.2 Future Work	165
6.2.1 Graph Clustering	165
6.2.2 Graph Layout	166
6.2.3 Edge Routing	167
Bibliography	169

List of Figures

1.1	Minard's flow map of Napoleon's Russian Campaign of 1812.	2
1.2	Simple artificial social network graph.	2
2.1	An undirected graph modelling the social connections of the karate club studied by Zachary[Zac77].	11
2.2	A simple graph showing the local clustering coefficient of each vertex. . . .	13
2.3	The shortest path between the two green nodes (10 and 4) is via the two yellow nodes (2 and 5)	13
2.4	A simple graph illustrating the betweenness centrality of each vertex. . . .	15
2.5	A simple graph illustrating the betweenness centrality of edges.	16
2.6	An example of the node sets used by Auber <i>et al.</i>	17
2.7	Example of clustered graphs with different MQ values (clusters are denoted by node colour)	23
2.8	Force Directed Layout of a graph containing 91 vertices.	25
2.9	FM3 multi-level layout example.	27
2.10	A simple circular layout of a 10 node graph	30
2.11	An example of a balloon tree layout	31
2.12	A clustered circular layout.	31
2.13	100 node ring lattice	35
2.14	A fully random procedurally generated graph	35
2.15	A small world graph generated by Watts and Strogatz' approach	36
2.16	Hierarchical edge bundling, software system example.	43
2.17	Migration example using geometric edge clustering	43
2.18	Edge Clustering example	45
2.19	Example image taken from [ZYC ⁺ 08], showing the effect of Zhou's Hierarchical edge bundling. The graphs are unbundled in the top row and bundled in the bottom).	45
2.20	A World War 2 stereoscope and Case.	47
2.21	Single camera projection.	50
2.22	Stereo camera projection.	51

2.23	The vergence angle of the eyes θ changes depending on the distance to the object being focused on.	52
3.1	Wikipedia graph with 91 vertices and 567 edges laid out using a simple force directed algorithm.	63
3.2	Graph from Figure 3.1 using our approach.	63
3.3	The modularity of graphs containing 60 nodes and increasing in density, using the described clustering approaches for building 4 clusters.	64
3.4	Modularity for building 5 clusters.	65
3.5	Modularity for building 6 clusters.	65
3.6	The modularity of graphs containing 60 nodes and 240 edges increasing in clustering coefficient, when clustered using 4 nodes of interest.	66
3.7	The modularity of graphs increasing in clustering coefficient, when clustered using 5 nodes of interest.	67
3.8	Sports club graph with 100 vertices and 803 edges laid out using a simple force directed algorithm.	67
3.9	Graph from figure 3.8 using our approach.	68
3.10	Layout of the Genealogy of Influence graph.	70
3.11	Clustered layout of the Genealogy of Influence graph.	71
3.12	The number of correctly clustered nodes.	72
3.13	A simple illustrative clustering example.	76
3.14	The average clustering coefficient of test graphs.	78
3.15	Evaluation of graphs with 200 Nodes and a density of 0.03 ($d_l = 3$), and an increasing level of randomness, denoted by p value.	79
3.16	Evaluation of graphs with 200 Nodes and a density of 0.07 ($d_l = 7$), and an increasing level of randomness, denoted by p value.	81
3.17	Evaluation of graphs with 200 Nodes and a density of 0.07 ($d_l = 7$), and an increasing level of randomness, denoted by p value.	82
3.18	Evaluation of a graphs with 200 Nodes and a density of 0.51 ($d_l = 51$), and an increasing level of randomness, denoted by p value.	83
3.19	Evaluation of a graph with 200 Nodes and a constant input rewiring probability $p = 0.1$, and an increasing density.	84
3.20	Evaluation of a graph with 200 Nodes and a constant input rewiring probability $p = 0.95$, and an increasing density.	86
3.21	Evaluation of AC3 vs. Edge betweenness for graphs of increasing randomness.	87
3.22	Evaluation of test graphs when clustered using Newman and Girvan's Edge Betweenness Centrality clustering (EBC) and our clustering coefficient heuristic (AC3) for comparison.	88

3.23	Number of clusters generated using Edge betweenness Centrality Clustering.	88
3.24	The infoviz data set laid out using FM3.	91
3.25	Infoviz edge betweenness centrality example	92
3.26	The infoviz data set split into 4 clusters using the AC3 approach.	94
3.27	The infoviz data set split into 4 clusters using the AC3 approach, keyword highlighted.	97
3.28	The infoviz data set split into 4 clusters using the modularity approach, keyword highlighted.	98
3.29	The infoviz data set split into 4 clusters using the BFS approach, keyword highlighted.	98
4.1	A 100 node procedural small world graph clustered around 4 nodes.	103
4.2	Node reordering and cluster rotation example	104
4.3	A simple example illustrating node reordering with 2 clusters.	105
4.4	Node reordering example	105
4.5	Three level deep hierarchical clustering example	107
4.6	Hierarchical Clustering, simple example	109
4.7	Hierarchy generation example, node assignment	110
4.8	Hierarchical force directed layout example	112
4.9	Multilevel layout processing illustration.	112
4.10	100 node small world graph, hierarchy approach.	114
4.11	400 node small world graph, hierarchy approach.	115
4.12	100 node small world graph, multilevel approach.	116
4.13	400 node small world graph, multilevel approach.	117
4.14	Hierarchically edge bundled versions of 100 node Small World Graph.	118
5.1	Edge experiment dense graph example	123
5.2	An example of a graph generated for our experiments, rendered with tightly bundled edges.	124
5.3	Illustration of the visual impact of different levels of bundling strength β	129
5.4	The impact of bundling on user accuracy and response time (in seconds)	132
5.5	The impact of node count on the effectiveness of bundling for the cluster connectivity experiment.	135
5.6	The impact of edge density on the effectiveness of bundling for the cluster connectivity experiment	136
5.7	Overall Results for the path tracing experiment.	137
5.8	Overall Results for the Cluster Connectivity experiment.	138
5.9	An example of the three dimensional bundles created by Lambert et al. [LBA10a]	139

5.10	Illustration of extension of bundling into 3D.	140
5.11	Shifting edge points and edge control points.	141
5.12	Different types of control point shift.	142
5.13	An illustration of the difference between the depth functions.	143
5.14	Illustration of the visual impact of rotating clusters and reordering nodes in clusters $\beta = 0.0$	145
5.15	Illustration of the impact of adding depth to straight line edges $\beta = 0.0$. .	146
5.16	Low density experiment graph example.	147
5.17	Unweighted means of the two significant single factors resulting from the Analysis Of VAriance for the path tracing experiment.	150
5.18	Unweighted means of the accuracy and user time taken for the depth fac- tors, resulting form the ANOVA for the path tracing experiment.	150
5.19	The depth types show significant results when analysed based on layout type.	153
5.20	Interaction effects from the cluster connectivity experiment.	155
5.21	An experiment random graph shown with and without edge shading. . . .	157
5.22	Interaction effects between bundling, layout and depth for the follow on path tracing experiment.	159
5.23	Interaction effect between bundling, depth and path length for the follow on path tracing experiment.	160

Chapter 1

Introduction

WE LIVE IN AN ERA WHERE MORE DATA IS PUBLICLY AVAILABLE THAN EVER BEFORE. The internet makes large volumes of data searchable, relatable and filterable. People model and enhance their real world social networks through websites such as Facebook and Linked-in, and generate further linked content through blogs and services such as twitter. Modern governments often release large volumes of data, ranging from internal emails to census results and social statistics. Technological, medical and scientific advances have allowed researches to generate huge volumes of information about the low-level workings of the universe and the basic genetic code of life. In 2010 Eric Schmidt, CEO of Google, claimed that every two days mankind was generating as much information as it had from the dawn of civilisation until 2002. While the accuracy of Schmidt's claim may be debatable, it is clear that more information is produced daily by mankind than ever has been prior to this in our history.

In his foreword to Ware's 2004 book [Waro4] on Information Visualisation, Card succinctly describes Information Visualisation as "*the use of interactive visual representations of abstract data to amplify cognition*". Information Visualisation has only emerged as a distinct field of academic research in the last two decades, but representing data with images to help with understanding is not such a recent idea. One of the most widely known visualisations is Minard's 1869 visualisation of Napoleon's campaign against Russia, which per Edward Tufte "may well be the best statistical graphic ever drawn" [Tufo1]. The visualisation, seen in figure 1.1 succinctly conveys information about the size of Napoleon's army, the position of the army, the direction of the army's movement and the weather condition over the temporal duration of the campaign in a single graphic.

Graph visualisation often represents data visualisation at its most abstract. In the simplest of terms, graphs can be considered as a mapping of relationships between entities. Graphs are often visualised as node-link diagrams, where nodes represent the entities and

1.1 Motivation

The difficulty of visualising large dense graph data sets lies not just in processing power and display size but also in the inherent visual complexity of a large data set. Visualisation of large data sets is an outstanding challenge in the field of visualisation in terms of comprehending the data as well as scaling algorithms for tasks such as layout and clustering [Che05, New04, Scho7] and many attempts have been made at addressing the visualisation of large graphs in terms of system scalability and comprehension [FvNo6, ETNG⁺08, ACJM03, Wil97]. Clutter is defined by Rosenholtz et al [RLMJ05] as “*the state in which excess items, or their representation or organization, lead to a degradation of performance at some task*”. Clutter resulting from thousands of nodes and an order of magnitude more of edges negatively impacts comprehensibility. Therefore the minimisation of clutter should be a concern of any graph visualisation. Graph analysis, clustering and visualisation are used to help give users a better understanding of topics such as software engineering [KLo8, BD07], social networks [Zac77, ACJM03] computer networks [Wil97], citation network analysis [ET07] and biological structures [ETNG⁺08].

Frequently visualisation applications are designed with a specific target domain or dataset in mind. Examples of such include computer program visualisation [KLo8], visual exploration of the Internet Movie Database (IMDB) [ACJM03] and visualisation of scientific citation networks [ET07]. Such a targeted development of an application is able to take advantage of characteristics of the data being visualised. Prior knowledge of the structure or characteristics of graph data allows for a targeted choice of cluster or layout algorithms that will be most suitable for the data.

Graphs with random edge distributions, where the distribution of edges among the vertices follows a Poisson distribution, are often generated procedurally and used to provide insight into graph theory. However, graphs modelling real world systems are not completely random and often contain an element of structure. We propose that this structure can be used by an agglomerative clustering algorithm to generate clusterings of graphs that aid in comprehension and layout, based on the user specification of nodes of interest. While traditional layout methods such as force directed layouts are very effective when it comes to laying out low-density graphs, real world graphs are frequently quite dense [Melo6]. However as graphs become more dense these layout approaches struggle in terms of aesthetics and comprehensibility, as well as in terms of algorithm execution time. The resulting layout can often contain groups of densely connected nodes in unreadable tight clusters with a large amount of edge overlaps. Also, while there is often a range of input parameters for force directed layouts, the user very often has little control over the final result. Different users may require different perspectives, and most layout algorithms do not provide this flexibility in terms of influencing the basic layout. We aim to provide the user with the ability to lay out a graph based around nodes of interest selected by them, to allow different

perspectives of the same data set to be generated.

The overall objective of our research is to allow users to get a better comprehension of the relationships between data entities in the visualisation of real world systems. In addition to our clustering and layout, we also evaluate edge routing techniques to show how these sort of graphs may be best visualised by a user to reduce the clutter caused by the edge density.

1.2 Key Concepts

Graph Visualisation is an extremely broad field covering many related topics. As part of this research we have engaged in many different aspects of the fields, such as clustering, layout, edge routing, graph generation, evaluation edge routing and the stereoscopic display of graphs. The purpose of graph layout algorithms is to allow for an easier understanding of the data by positioning nodes in such a way that the graph is more aesthetically pleasing to a user[HJ06]. As well as layout, the routing of edges also plays a large role in comprehensibility [WPCM02].

A *clustered graph* is a graph with recursive clustering structure over the vertices. Eades and Feng [EF97] give examples of two dimensional clustered graphs as well as describing an approach for visualising a graph with a multilevel clustering hierarchy in three dimensions. In their example, the clustering structure is an attribute of the graphs and vertices. However, in many cases, if a graph is to be clustered there may be no intrinsic attribute or parameter which describes the clustering hierarchy. Therefore, this structure may need to be determined by a clustering algorithm. There are different algorithmic approaches to clustering, some of which rely on the underlying structure of the graph, such as Newman and Girvan's top-down divisive clustering [NG04].

The analysis of various different types of networks has shown that many networks across different fields have similar characteristics and can be classified as *small world graphs* [WS98, CF09, ACJM03, vHW08a]. Small world networks are characterised by a high level of clustering and short path lengths. The term "small world" is based on the commonly known concept of there being six degrees of separation between any two people alive. It does not refer to the size of the graph, so many very large graphs can be considered small world graphs. Given the clustered structural nature of small world graphs, a suitable clustering algorithm may prove effective in dividing a large small world graph into more comprehensible clusters for visualisation.

Edges are one of the main sources of clutter in dense graphs. *Edge Bundling* [Holo6]

is a technique by which an attempt is made to reduce edge clutter by grouping edges together into “bundles” of curves. Though often cited as a clutter reduction technique, edge bundling’s claimed effectiveness has little basis in empirical evidence.

Three-Dimensional stereoscopic displays are becoming more widely available as commodity hardware. Research has shown [WMo8] that users can more easily comprehend large graphs when utilising three dimensional display techniques.

1.3 Contribution

Our goal of improving the comprehensibility of small world graphs has been broken down into multiple contributions.

- Our initial contribution is our novel approach to agglomeratively clustering small world graphs around nodes of interest. We propose average local clustering coefficient of a cluster as a heuristic to guide this agglomerative clustering.
- We demonstrate the effectiveness of our chosen heuristic by evaluating it against other metrics using a large range of graphs.
- We extend our clustering to generate a hierarchical clustering which reflects the relationships between the clusters created by our approach.
- We utilise our hierarchical clustering to perform a multilevel layout of the graph and aid in the routing of edges in the resulting layout. We also demonstrate an approach to reduce edge crossings in circularly laid out clustering hierarchies.
- We empirically evaluate “*Edge-Bundling*” a popular clutter reduction technique, which we utilise in our graph presentation.
- To support our evaluation we developed an approach to create procedurally generated graphs suitable for such experiments.
- We extend edge bundling into three dimensions and empirically evaluate the use of three dimensional stereoscopic depth to determine its effectiveness at reducing the impact of edge bundling on low level path tracing tasks.

1.4 Scope

The scope of this thesis covers many different aspects of graph presentation. The layout and agglomerative clustering of a graph are one form of presentation, the routing of the edges is

another. The common focus of each of these areas is how to best improve user performance. Our scope does not cover a full comparison of all clustering and layout approaches. We provide low level evaluations of the techniques described in this thesis. Such evaluations yield results which are not domain specific and can be generalised across many fields. High level domain specific evaluation experiments, using domain experts as participants, are beyond the scope of this thesis.

1.5 Related Publications

Some of the research described in this has previously been peer reviewed and published at international conferences. The following papers contain materials which were created as part of our research into this Phd. thesis.

1. [MD12a] **An Empirical Study on the Impact of Edge Bundling on User Comprehension of Graphs:**
Fintan McGee, John Dingliana
Advanced Visual Interfaces 2012 in cooperation with ACM-SIGCHI, Capri Island, Italy
2. [MD12b] **VISUALISING SMALLWORLD GRAPHS: Agglomerative clustering of Small World Graphs around nodes of interest:**
Fintan McGee, John Dingliana;
International Conference on Information Visualisation Theory and Applications 2012 (IVAPP 2012), Rome, Italy
3. [MD10] **An Evaluation of the use of Clustering Coefficient as a Heuristic for the Visualisation of Small World Graphs:**
Fintan McGee, John Dingliana;
Theory and Practice of Computer Graphics, UK 2010 (TPCG2010), Sheffield, UK

1.6 Thesis Layout

The rest of this thesis is laid out as follows:

- Chapter two provides a background on graph theory and describes the related work for this thesis. An overview of layout, clustering, graph evaluation, edge routing and three dimensional stereoscopic visualisation of graphs is also provided along with an examination of the state of the art in each.
- Chapter three covers graph clustering techniques. We present a new agglomerative clustering to allow users rearrange graphs around nodes of interest. We suggest a

heuristic, based on the structure of small world graphs and evaluate it against other heuristics across a wide range of graphs.

- Chapter four covers graph layout. We extend our clustering from chapter three to generate a clustering hierarchy. We utilise this clustering hierarchy for multilevel layouts of graphs, which reflect the users selected nodes of interest. We also utilise this hierarchy to route edges in the resulting graph.
- Chapter five examines the role of edge routing in hierarchically clustered graphs. We provide an empirical evaluation of edge bundling. We extend edge bundling into three dimensions for stereoscopic viewing of graphs and evaluate the impact it has on user performance.
- Chapter six describes our conclusions and directions for future work.

Chapter 2

Background and Related Work

GRAPH VISUALISATION IS A BROAD FIELD OF RESEARCH, covering many topics including computer graphics, mathematics, graph theory, art and human perception. In this chapter we present the basic concepts necessary to understand what follows, and we describe the related research that the chapters following this are built upon.

2.1 Graphs

When the term, “*Graph*” is used, many people immediately conjure up an image of a visual representation of a graph, forgetting that underlying this is a mathematical definition which exists completely independently from any visual interpretation. An undirected *graph* $G = (V, E)$ is defined by a set of vertices $v \in V = \{v_1, v_2 \dots v_n\}$ and a set of edges $e \in E$ connecting vertices $x \in V$ and $y \in V$ with $e(x, y) = e(y, x)$. If a graph is a *weighted graph* there is an associated numerical weight for each edge $w(e(x, y))$.

For a *directed graph* $e(x, y) \neq e(y, x)$. A directed graph can be transformed into an undirected graph by ignoring the edge direction. An unweighted graph can be considered to be a weighted graph where $w(e) = 1, \forall e \in E$. An edge in a directed graph is considered to have one vertex that is the edge source and one that is the edge target. If a vertex is the target, the edge is considered an *in-edge* to that vertex. If a vertex is the source, the edge is considered an *out-edge* edge to that vertex.

In practical terms the vertices of a graph model entities in the real world, and the edges model relationships between those entities. Social networks are an often encountered example, in which the vertices model people and the edges model a friendship or other social relationships between the people. A simple example of a social network can be seen in the work done by Zachary[Zac77], in which the members of a karate club were modelled as vertices for a graph, and their interactions were modelled as the edges. Zachary was able to process this social network to provided information about the future state of the group, which split into separate subgroups. Visualisations of this group can be seen in figure 2.1.

Directed graphs are frequently used to model systems where the direction of the relationship between vertices is important, such as a graph modelling predator and prey relationships in an ecosystem. This impacts how a graph can be traversed (i.e. moving from one vertex to another following the direction of the edges). However for tasks such as the laying out of vertices of a graph, it is often possible to ignore edge direction and still achieve a good result. For the remainder of this thesis, we assume all of the graphs that we are visualising are undirected graphs. However many of the techniques we use could also be applied to directed graphs.

2.1.1 Graph Visualisations

Most frequently graph visualisations take the form of node-link diagrams, consisting of nodes representing the vertices of the graphs and links between them depicting the edges, as can be seen in figure 2.1a. Matrix based visualisations offer an alternative approach (see figure 2.1b and for a recent large scale example see [ETNG⁺08]). Both approaches offer their own challenges. When using a node-link visualisation, the layout of the nodes and links has a significant impact on user comprehensibility [Pur97]. Analogous to this for matrix visualisation is vertex ordering. The node link form of visualisation is considered more intuitive than the alternative of matrix based layout [GFC04, FvN06]. Node-link diagrams also allow for a more flexible use of the display space, for example the use of hyperbolic or three dimensional space for layout [Mun98, WM08]. There are also visualisations which take a hybrid approach combining both matrix and node-link visualisation [HFM07]. This thesis focuses exclusively on the node-link style of visualisation. Usually when referring to a node link style of display, vertices are referred to as nodes, as for most purposes the terms node and vertex are inter-changeable.

2.1.2 Small World Graphs

Small world graphs are a category of graphs encountered frequently in models of real world systems. Milgram [Mil67] first identified the phenomenon in his work focused on social networks. The concept was more recently revived by Watts and Strogatz [WS98] and Watts [Wato3] and has been shown to hold true for a variety of networks, such as the relationships between actors and films [ACJM03] as well as computer systems [CF09] and citation networks [vHo4]. Small world networks are defined by two main characteristics. The first concerns the average of the shortest path lengths between each pair of vertices for the entire graph. The second characteristic is the average local clustering coefficient of the graph, which is defined as the average of the clustering coefficients for each vertex. To determine if a graph can be considered a small world graph, it is compared to a randomly generated graph with the same number of vertices and edges. A small world graph will have approx-

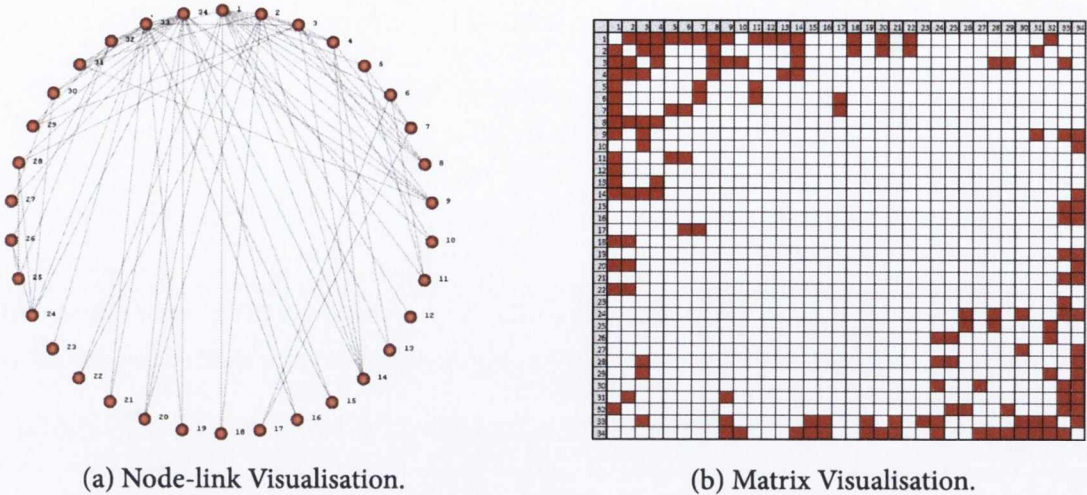


Figure 2.1: An undirected graph modelling the social connections of the karate club studied by Zachary[Zac77].

imately the same average path length, but a considerably higher (by orders of magnitude) average local clustering coefficient.

To define the average local clustering coefficient of a vertex, we first need to define the neighbourhood of a vertex.

Vertex Neighbourhood definition

The neighbourhood of a vertex v , denoted Γ_v is defined as the set of all vertices adjacent to v , not including v itself. We can extend this to a set of vertices defined by an induced subgraph $S = (V_s, E_s)$ (where $V_s \subset V$ and $E_s \subset E$, and $E_s \subset (v_i, v_j), \forall v_i, v_j \in V_s$). An induced subgraph is a subgraph where for every edge that exists between nodes in the subgraph at the parent level, there is a corresponding edge at the subgraph level. This results in Γ_S being defined as the set of vertices adjacent to all $v \in V_s$ but not including those vertices which are part of the subgraph. If $S = \Gamma_v$ then it follows $\Gamma_S = \Gamma(\Gamma_v) = \Gamma_v^2$. The size of the neighbourhood of a vertex is often referred to as the *degree* of a vertex. For a directed graph, there is both an *in-degree* and *out-degree* associated with each vertex. The in-degree is the the number of in-edges and the out degree is the number of out-edges.

Clustering Coefficient Definition

The *clustering coefficient* for a vertex, denoted by γ_v , is most commonly defined as the ratio of edges connecting the neighbours of a vertex to the maximum number of edges that could possibly connect the neighbours of the vertex [Wato3]. The clustering coefficient c

for a vertex v in an undirected graph is given by

$$\gamma_v = \frac{|E(\Gamma_v)|}{\binom{k_v}{2}}$$

where $|E(\Gamma_v)|$ is the magnitude of the set of edges connecting neighbours of the vertex, k is the neighbourhood size of the vertex, (i.e. $|\Gamma_v|$) and $\binom{k_v}{2}$ is maximum possible number of edges in Γ_v . From the above it can be seen that a vertex needs at least two neighbours to have a valid clustering coefficient value. For a directed graph the clustering coefficient is given by

$$\gamma_v = \frac{|E(\Gamma_v)|}{k(k-1)}$$

This is due to the fact that a directed graph can have double the amount of edges and $k(k-1) = \frac{1}{2}\binom{k_v}{2}$. The average local clustering coefficient for a graph, often referred to as the global clustering coefficient of the graph, is given by

$$\gamma_G = \frac{\sum_v \gamma_v}{|V|}$$

Figure 2.2 shows a simple graph and the local clustering coefficients associated with each node. If a node has a clustering coefficient of 1.0, its neighbourhood can be said to form a *clique*, a set of nodes where each node is adjacent to every other node in the set. As part of our clustering discussed in chapter 3 we use the concept of an average cluster clustering coefficient. The average clustering coefficient of a cluster, reflects the level of interconnectivity of nodes within the cluster. Therefore when calculating the clustering coefficient of nodes with a cluster, to generate the average cluster clustering coefficient, only neighbours within the same cluster are considered. A graph cluster with a high average cluster clustering coefficient, indicates that all of the nodes within the cluster have many interconnected neighbours within that cluster.

Average Shortest Path Length

The shortest path between two vertices (often referred to as the *geodesic distance*) is the smallest possible set of vertices it takes to traverse from one vertex to another. Often there may be more than one shortest path between a pair of nodes. The average shortest path length reflects the connectivity of a graph. As graphs become more dense the average shortest path length generally decreases, as there are more edges to traverse between vertices. Calculating the average shortest path length of a graph requires calculating the shortest path length between all possible pairs of nodes, which is a computationally intensive task. While Dijkstra's shortest path algorithm [Dij59] is suitable for finding the shortest paths from a single node, it is not efficient for calculating the shortest path for all nodes. Two of

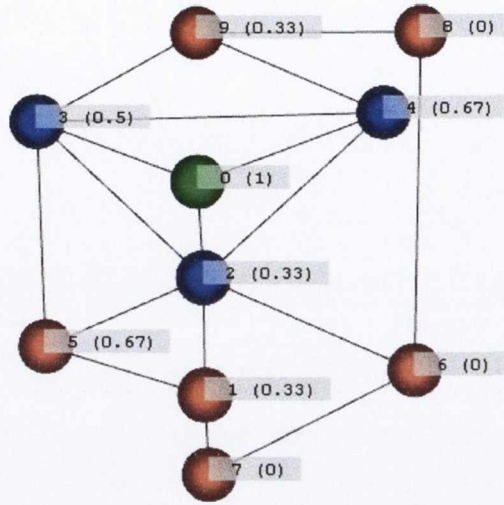


Figure 2.2: A simple graph showing the local clustering coefficient of each vertex (in brackets). Note that the green node has a coefficient of 1 as all of its neighbours (the blue nodes) are connected to each other.

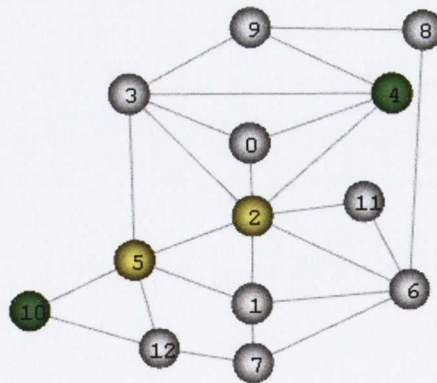


Figure 2.3: The shortest path between the two green nodes (10 and 4) is via the two yellow nodes (2 and 5)

the most commonly used algorithms for all pairs shortest path calculations are Johnson's [Joh77] algorithm, which is of complexity $\mathcal{O}(|V||E|\log|V|)$, and the Floyd-Warshall algorithm [Flo62], which is of complexity $\mathcal{O}(|V|^3)$. Due to the relative complexities Johnson's algorithm is preferred for less dense graphs, and Floyd-Warshall for more dense ones.

Small World Graph Specific Visualisation Approaches

As so many real world graphs fall within the domain of small world graphs, much research has been done on developing graph clustering and layout techniques specific to the small world model. Auber *et al.* [ACJM03] developed an application called *SWViz*, which provided multi-scale visualisation of small world networks. The author's observed that if networks display small world properties, their highly connected components also display small world properties. They utilised this observation to create a multi-scale visualisation. The highly connected components are determined by decomposing the network into strongly connected components by removing edges using the *edge clustering index* described in section 2.1.3.

McPherson *et al.* [MMO05] describe a system for discovering parametric clusters in social small world graphs. They describe an application that utilises Markov Clustering (described in section 2.2.2) to assign cluster identifiers to nodes. The system provides an initial tree based layout of the graph and allows users to resize and colour nodes based on attributes (such as node degree and clustering coefficient), as well as selected sub-graphs based on attributes. The system allows further clustering by combining node attributes such as the previously described cluster identifier, node degree, local clustering coefficient or any arbitrary value assigned to a node. These clusters are defined as part of a layout technique referred to as a *Self Organising Map* which projects from vector of input attributes onto a two dimensional grid. This layout is then further enhanced by a customised version of the Fruchterman Reingold layout (described in section 2.3.2), which allows for user input. McPherson *et al.* demonstrate this approach providing images of the result when clustering a social small world graph. The system can be used for any attributes, not necessarily local clustering coefficient, so it is not clear as to why it could not be used more generally than for specifically small world graphs.

Van Ham and Wattenberg [vHW08a] use edge betweenness centrality (described in section 2.1.3) as a basis for building a minimum spanning tree to aid with the layout of small world graphs (a minimum spanning tree is a subgraph that contains every vertex and the minimal set of edges which does not disconnect the parent graph). The clustered nature of small world graphs means that nodes which are unrelated to each other (with a large geodesic distance between them) will be positioned further away from each other in the spanning tree used as an input to layout. If the edges were distributed evenly or randomly there would be no benefit offered in the resulting layout. In van Ham and Wattenberg's

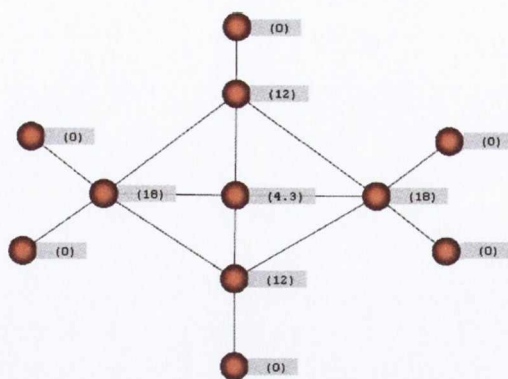


Figure 2.4: A simple graph illustrating the betweenness centrality of each vertex.

example nodes which are related to each other by an external classification do appear closer to each other in the final graph layout.

The preceding approaches demonstrate that the characteristics of a small world graph can be utilised as input into their clustering and layout. We utilise these characteristics for our clustering described in chapter 3. Our clustering is also used as an input to our layout described in chapter 4.

2.1.3 Graph Centralities

Centrality is a measure of importance of a vertex, or an edge in a graph [New10]. There are many different types of centrality measure, those described here are the most relevant subset. Centralities can also be used to guide algorithms for clustering by Newman and Girvan [NG04], or layout done by van Ham and Wattenberg [VHW08b].

Vertex Degree: This is one of the most straightforward centrality measures. For an undirected graph it is the number of edges connected to a vertex, or as stated above the size of a vertex's neighbourhood. For example in the a social network of friendships, the degree centrality rates those with more friends as more important.

Vertex Betweenness Centrality: Vertex betweenness centrality is a measure of how many shortest paths a vertex appears on. To derive vertex betweenness centrality for all vertices appears to require the complexity of the all pairs shortest paths algorithms mentioned previously. However, Brandes [Bra01] has developed an optimised approach which is $\mathcal{O}(|V||E|)$. Figure 2.4 shows a simple graph with the vertex betweenness of each node.

Edge Betweenness Centrality: Edge betweenness centrality is a measure of how many shortest paths a specific edge appears on. It can also be calculated in $\mathcal{O}(|V||E|)$ using

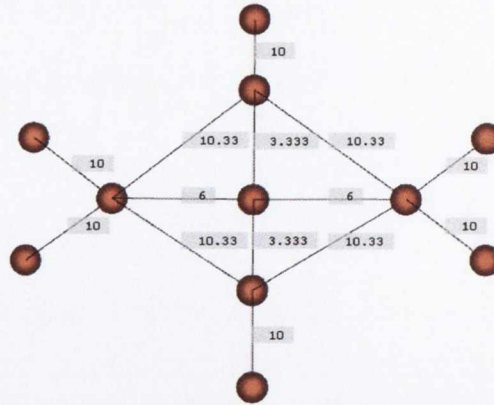


Figure 2.5: A simple graph illustrating the betweenness centrality of edges. The fractional values of some edges are a result of them appearing on multiple shortest paths of the same length.

an adapted version Brandes algorithm. Figure 2.5 illustrates edge centralities in a simple graph.

Local Clustering Coefficient: The local clustering coefficient of a vertex can also be considered a form of centrality. A vertex with a high clustering coefficient indicates that it is part of a strongly connected set of nodes, if the clustering coefficient is 1.0 the vertex is part of a clique. As commented by Newman, [New10], it is similar to vertex betweenness centrality in that it reflects the importance of a vertex based on its connections. However vertex betweenness centrality extends beyond the vertex's immediate neighbourhood and if a node has a high clustering coefficient, it most likely will have a relatively low vertex betweenness, as its neighbours will offer alternative shorter paths from more distant nodes.

Edge Clustering Index: The clustering coefficient of a node is often also referred to as the clustering index. Auber et al. [ACJM03] following on from Chiricota et al [CJM03] use a metric, originally defined by Alper[AK95], which generalises the previous definition for clustering coefficient for a vertex to apply to edges. This edge clustering index is used by Auber *et al.* to determine the strength of edges within the graph, and thus allows clusters to be determined by removing weak edges (those with a low clustering coefficient), similar to the way edge betweenness centrality is used by Newman and Girvan [NG04]. Their approach is as follows: Given an edge consisting of nodes u and v , the edge's neighbourhood is divided into 3 sets. $M(u)$ is the set of nodes that are neighbours of u but not v . $M(v)$ is the set of nodes that are neighbours of v but not u . $W(u, v)$ is the set of all nodes which are neighbours of both. Clearly these 3 sets are distinct, however they also may be connected by edges which do not contain either u or v (see figure 2.6).

Let $s(A, B)$ denote the strength of connectivity between two set of (distinct) nodes.

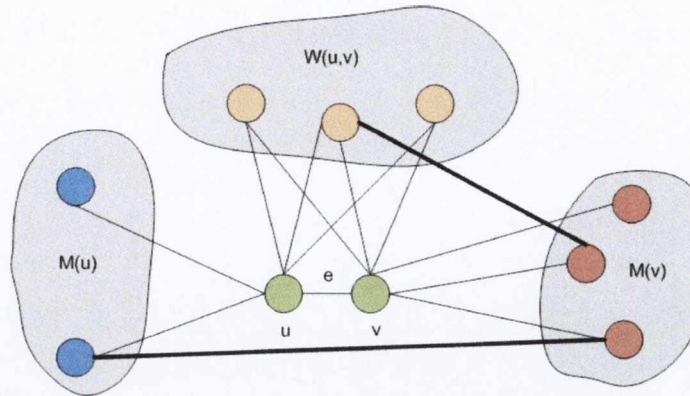


Figure 2.6: An example of the node sets used by Auber *et al.* [ACJM03] in calculating clustering index of an edge $e = (u,v)$

Let $r(A, B)$ be equal to the number of edges between two nodes in set A and the nodes in set B , then $s(A, B) = r(A, B)/|A| \cdot |B|$. This is in effect calculating the ratio of amount of connections between sets A and B and the maximum possible number of connections between the set A and B . Note that any edges that go between any 2 of the sets $M(u)$, $M(v)$ and $W(u, v)$ are part of a cycle of 4 edges that passes through (u, v) . A cycle is a path that begin and ends with the same vertex. 4 is the maximum path length of any cycle between the sets.

The definition of $W(u, v)$ means that there are as many cycles of length 3 as there are nodes in $W(u, v)$. The proportion of possible length 3 cycles is given by $|W(u, v)|/(|M(u)| + |M(v)| + |W(u, v)|)$. Summing the ratios calculated for each pair of connected sets, the ratio calculated for the set $W(u, v)$ with itself and the proportion of possible cycles of length 3, provides the edge clustering index γ_e .

$$\gamma_e = s(M(u), W(u, v)) + s(W(u, v), M(v)) + s(M(u), M(v)) \\ + s(W(u, v), W(u, v)) + |W(u, v)|/(|M(u)| + |M(v)| + |W(u, v)|)$$

2.1.4 Graph Edge Density

In their taxonomy of clutter reduction Ellis and Dix describe clutter as the result of “too much data on too small an area of the display”[ED07]. In a dense graph, edge congestion is the primary source of clutter. The links in a node link visualisation convey important information. However if they become too dense the graph becomes less comprehensible, resulting in nodes and other links becoming obscured. In terms of graph theory the density of a graph is usually considered to be the ratio of edges to the maximum possible number

of edges in the graph [CM83]. For an undirected graph this can be described as

$$d = \frac{|E|}{(|V|(|V| - 1)/2)} \quad (2.1)$$

A graph is then considered dense in mathematical terms if this ratio approaches 1.0, a graph with density 1.0 is called a *complete graph*. If a graph's density is close to 0.0 it is considered to be a *sparse graph*. However in practical real world examples of graph visualisation, which may contain huge numbers of nodes, a density approaching 1.0 is rarely seen. A complete graph with 1000 nodes would have 499,500 edges. Visualising a graph approaching this level of density using a standard node-link approach would not serve any useful purpose as the individual edges would be unreadable.

Another common measure of the density of a graph is the ratio of edges to nodes, referred to as the linear density

$$d_l = \frac{|E|}{|V|} \quad (2.2)$$

where $|E|$ denotes the number of edges in the graph. Most real-world graphs have a value of $d_l \leq 10$ [Melo6], which is still enough to cause a large amount of clutter. Melaçon *et al.* [Melo6] give an example of real world graphs which have even higher densities, such as web-crawl based graphs with $d_l = 25.57$. Given the frequency that dense graphs are encountered in the real world it is important to include edge density as part of any graph evaluation. It is clear that graph theoretic density scales the number of edges more dramatically for a change in vertex count, so for comparison of densities between graphs with different node counts linear density provides a clearer comparison.

2.2 Graph Clustering

2.2.1 Clustering Overview

Eades and Feng [EF97] describe clustered graphs as “graphs with recursive clustering structure over the vertices”. In their work they provide examples of two-dimensional clustered graphs and describe an approach for visualising graphs with a multilevel clustering hierarchy in three dimensions. In their examples, the clustering structure is an attribute of the graphs and vertices. However, in many cases if a graph is to be clustered there may be no intrinsic attribute or parameter which describes the clustering hierarchy. There are many different approaches to graph clustering (or partitioning as it is often referred to). Some methods use an algebraic approach, working on a mathematical representation of the graph [Chu97]. Other methods such as Edge Betweenness Centrality Clustering [NGo4] use a graph theoretic based approach, calculating graph theory characteristics of vertices or edges that are then used to partition the graph into clusters. Some clustering algorithms,

such as edge betweenness centrality clustering take a top down, or *divisive* approach splitting the graph into separate clusters. Others take a bottom-up or agglomerative approach, merging sets of nodes together to form clusters.

Many approaches generate a flat clustering of the graph, while others produce clustering hierarchies. Clustering hierarchies are clusterings where the clusters of a graph are themselves recursively clustered into sub-clusters. Graph clustering is a difficult problem that is NP complete [NGo4]. Algorithmically defined clusters may not match what an authority on the graph data believes is a good clustering. In their comparison of graph clustering algorithms for recovering software architecture module views, Bittencourt and Guerrero [BG09] comment that “fully automated clustering techniques alone cannot recover module views in a sensible way”. Schaeffer [Scho7] provides an in depth review of clustering methods and related topics.

2.2.2 Clustering Approaches

One of the most widely know forms of clustering is K-means clustering [HW79]. This is a very general clustering algorithm that is used for many purposes, not just graph clustering. In this approach the data points to be clustered (nodes in the case of a graph) are placed randomly in k clusters. The center of gravity of each cluster is calculated and each node is assigned to the nearest cluster based on a distance function between data points and a cluster’s center of gravity. The distance function is often, but not always, the euclidean distance. The process is repeated until the changes in clustering falls beneath a pre-determined threshold. The vectors used as an input to the distance metric may represent position in two or three dimensions, resulting in a geometric clustering. However K-means clustering may also be done with a vector of any level of dimensionality, representing other values than position in a graph space. For example, in Hopcraft *et al’s* [HKKS03] use of k-means clustering of a citation network, the data point vector used for the distance function represents the citations between papers, and has as many dimensions as there are citations in the paper.

Within graph visualisation, the aim of *geometric clustering* is to have vertices that are geometrically close to each other share a cluster and distant vertices appear in separate clusters. K-means clustering is an effective way to accomplish this. An example of such a clustering is given by Quigley and Eades’ FADE algorithm [QE01] in which a quad-tree is used alongside a modified force directed algorithm. The clustering provides different levels of abstraction at which a graph can be viewed.

Agglomerative Clustering

Agglomerative clustering is a bottom-up approach to clustering, merging nodes together iteratively to form clusters. Depending on the approach clusters can be merged together

or individual nodes can be added to clusters. When merging nodes and clusters together a similarity function is used to determine the suitability of the merge.

Hopcroft *et al.* [HKKS03] provide an agglomerative clustering of a co-citation network as part of their analysis on finding natural communities. They use a snapshot of a citation database or approximately 250,000 papers. The nodes in the extracted graph represent papers and the edges represent citation between them. The function used to determine which nodes should be agglomerated together is based on the product of the nodes' neighbourhood sizes, divided by the size of the intersection between the two neighbourhoods. The smaller this value, the closer the nodes are together and more suitable they are for merging.

Nodes can be merged together to form a flat clustering or a hierarchical clustering can be generated by repeatedly merging clusters as done by Hopcroft *et al.* This was also done by Newman [New04] using modularity, a metric utilised by Girvan and Newman in their previous work on edge betweenness centrality clustering [NG04], as a guiding heuristic for a greedy agglomerative clustering process. This agglomerative clustering produces a hierarchy of clusters. Modularity is then used as a metric to determine which level of the hierarchical clustering provides the best clustering. Modularity is described in more detail in section 2.2.3.

Algebraic Clustering

Algebraic methods work on algebraic representations of a graph. The most common algebraic form of a graph is an adjacency matrix. For an undirected unweighted graph $G = (V, E)$, the adjacency matrix is a square matrix with $|V|$ rows and columns. Given two nodes v_i and v_j , $i, j < |V|$, the value at entry (v_i, v_j) is equal to 1 if $(v_i, v_j) \in E$, otherwise it is 0. Algebraic methods work on this matrix and other algebraic matrices related to the graph such as the Laplacian matrix which is derived from the adjacency matrix and the degree matrix. The degree matrix of G is a $|V| \times |V|$ matrix where the diagonal entries (i, i) equal the degree of the i^{th} node of V . The Laplacian matrix is equal to the adjacency matrix minus the degree matrix. The analysis of these matrices and their characteristics, such as eigenvalues and eigenvectors, form the basis of the field of spectral graph theory [Chu97].

Spectral graph theory partitioning methods are used by Frishman and Tal [FT07] to cluster graphs as part of their GPU based layout. Algebraic techniques are also used for graph layout, for example the algebraic multigrid method (ACE) of Koren *et al* [KCH03]. Van Dongen's Markov Clustering (MCL) [vDoo] uses algebraic matrix representations of a graph as the transition matrix of the Markov chain used in his clustering approach. Clearly algebraic methods can provide many viable clustering approaches for graphs. However, our research focuses on graph theoretic approaches as these relate more closely to the visualisation of the graph on a display than the algebraic methods.

Edge Betweenness Centrality Clustering

Edge Betweenness Centrality Clustering is a divisive graph theoretic graph clustering method developed by Newman and Girvan [NG04]. *Edge betweenness centrality* is a measure of how important an edge is within a graph. It is determined by the number of shortest paths that an edge appears on out of all shortest paths for the graph as a whole. This algorithm is expensive, with a straight forward implementation being in $O(|E||V|^2)$, however Brandes [Bra01] proposes an alternative in $O(|E||V|)$. Similarly to Edge betweenness centrality, vertex betweenness centrality is defined as a measure of the number of shortest paths on which a vertex appears.

Newman and Girvan show that edge betweenness centrality can be used to partition a graph into clusters (or as they refer to them communities) based on the graph structure. Their approach consists of calculating the edge betweenness centrality for all edges, and removing the edge with the highest value. This is repeated until eventually the graph breaks into separate components and ultimately individual vertices. The partitioning at different stages of the algorithm is evaluated using modularity as a metric, and the iteration of the algorithm which produced the most modular components is used to assign vertices to clusters.

2.2.3 Clustering Evaluation

There are many different metrics used to evaluate clusterings. Boutin and Hascoët [BH04] discuss many other clustering evaluation approaches (referred to by them as clustering validation indices). They note that these evaluations are often difficult to interpret and compare. Evaluating the authoritativeness of a clustering is a difficult problem, not always readily solvable by a metric. Wu *et al.* [WHH05] use external clusterings in their evaluation of clustering algorithms for software systems to evaluate their chosen algorithms. They use the directory structure of the software system to create an authoritative clustering that reflect experts (i.e. the software developer). Clustering evaluation depends on the target application of the clustering. Bittencourt and Guerrero [BG09] and Wu *et al.* [WHH05] evaluate clustering algorithms in the domain of software analysis. Their evaluation metrics include distribution of cluster size (avoiding singleton clusters and clusters which consist of the majority of nodes), clustering stability (the clustering of a graph does not change much for a small change of the input graph) and authoritativeness (based on an external measure). These metrics are very useful for the the application domain of software evaluation, however they are not as suitable for our agglomerative clustering around nodes of interest which we describe in chapter 3. We describe next two of the metrics from the literature (which have also been used as heuristics to guide clustering), which are of relevance to our clustering evaluation in chapter 3.

Modularity

Newman and Girvan [NG04] define a measure of the quality of a division of a network graph, referred to as modularity. The measure is used to evaluate their community detection algorithm (which is essentially a top-down clustering algorithm). The measure has also been used in work by Newman [New04] as a heuristic value which is to be optimised, and hence guides the clustering rather than evaluate the quality of it. This metric is based upon the number of edges that start and end in the same cluster (referred to as communities in Newman and Girvan's paper). The modularity, Q , is calculated as

$$Q = \sum_i (e_i i - a_i^2)$$

Where $e_i i$ is the fraction of all edges that start and end in cluster i and a_i is the fraction of all edges that terminate in cluster i . A high level of modularity indicates a low number of inter-cluster edges. We believe that modularity provides a good metric, that translates across application fields.

Modularisation Quantity

Auber et al [ACJM03] and Chiricota et al.[CJM03] use a quality measure developed by Mancoridis et al [MMR⁺98] and utilised in Mancoridis et al's clustering tool "Bunch" [MMCG99]. This measure, denoted MQ (*Modularisation Quantity*) computes a value for any given partition of a graph. Chiricota et al. and Auber et al. use a slightly modified version of MQ that is defined only for undirected graphs as an evaluation measure. The MQ value is used by the Bunch tool as an function to be optimised to provide a good clustering (rather than evaluate one). Let A and B be two sets of disjoint nodes in a graph $G = (V, E)$, let s equal the ratio of edges between the two sets to the maximum possible number of edges between the two sets.

$$s(A, B) = \frac{e(A, B)}{|A| \cdot |B|}$$

Note that this ratio can be calculated for a set with itself. For a cluster A in an undirected Graph without self linking edges

$$s(A, A) = \frac{2(e(A, B))}{|A| \cdot (|A| - 1)}$$

If cluster A is a clique $s(A, A) = 1$. If none of the nodes in A are connected $s(A, A) = 0$. Given a partition (also referred to as a clustering) $C = (C_1, C_2, \dots, C_p)$ that divides the graph $G = (V, E)$ into p partitions the MQ score for that partition is given by:

$$MQ(C; G) = \frac{\sum_{i=1}^p s(C_i, C_i)}{p} - \frac{\sum_{i=1}^{p-1} \sum_{j=i+1}^p s(C_i, C_j)}{p(p-1)/2}$$

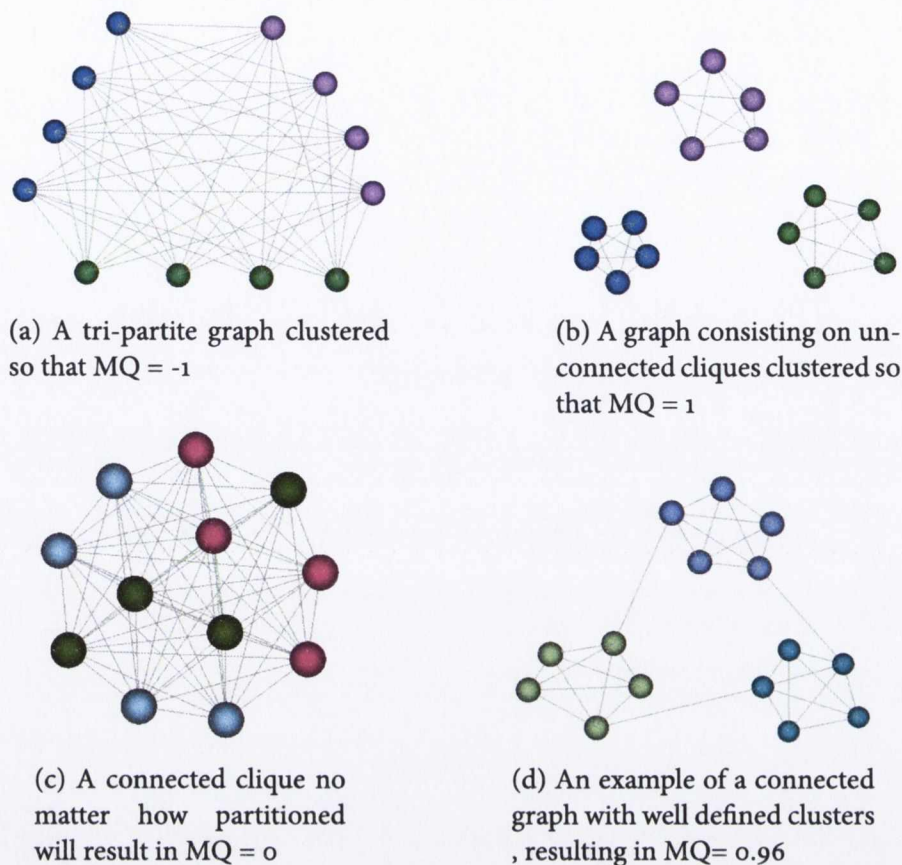


Figure 2.7: Example of clustered graphs with different MQ values (clusters are denoted by node colour)

Essentially this is a measure of the difference between the s ratio of intra-cluster edges denoted by $s(C_i, C_i)$ and the s ratio of inter-cluster edges, denoted by $s(C_i, C_j)$. The minimum value of MQ is -1 , representing a K -partite graph, where no nodes in a given cluster are connected to each other, but are connected to every other node in the graph. The maximum value is 1 , representing a non-connected graph where each cluster is a *clique* that is not connected to any other cluster.

Difference Between Modularity and Modularisation Quantity

The MQ metric differs to Newman and Girvan's modularity measure. Modularity compares the fraction of all edges that are intra-cluster edges to fraction of all edges that are inter-cluster edges. MQ is a measure of the difference between the average ratio of actual intra-cluster edges to the maximum amount of intra-cluster edges possible and the average ratio of the amount of inter-cluster edges to the maximum amount of inter-cluster edges possible. This means that modularity depends purely on the number of edges (which is bounded to the number of nodes) and MQ depends on the number of edges and the number of nodes directly (as the maximum possible number of edges between two clusters is a

function of the number of vertices).

2.3 Graph Layout

There are many different approaches to graph layout, each with the same aim of producing an image that is in some way aesthetically pleasing to a user and improving the users ability at some task. The different approaches encompass many different representations of a graph. Force directed layouts work by modelling a graph as a connected physical system. Algebraic approaches work directly on the adjacency matrix representation of a graph. Many layout approaches lay out an entire graph at once, while multi-level approaches create higher level representations of a graph and lay these out, using them as a basis for the positioning of the final graph nodes.

2.3.1 Force directed layouts

One of the most common types of layout is *force directed layout*. The early force directed approach by Eades[Ead84] was based on modelling an undirected graph as a system of springs. This was further enhanced by Kamada and Kawai [KK89] by addition of calculating an ideal layout between vertices which are not connected, and formulating the layout problem as an energy optimisation problem. Gansner et al [GKN05] have followed on from this, replacing Kamada and Kawai's local Newton-Raphson minimization of the energy function with a global approach called majorization from the field of Multi-Dimensional Scaling (an approach used for layout by Harel and Koren). Fruchterman and Reingold [FR91] developed a physics based algorithm which models attractive and repulsive forces between vertices as well as using the concept of a global energy value to limit the movement of nodes during layout. GEM[FLM95] is another force directed algorithm for undirected graphs where the vertices of the graph are modelled as charges repelling each other and the edges are modelled as springs. There are more recent versions of forced directed layout which employ a multilevel approach, such as GRIP[GK01], the Fast Multi-Scale method of Harel and Koren, [HK01], and the Fast Multi-pole Multi-Level Method of Hachul and Jünger [HJ05].

2.3.2 Fruchterman Reingold Layout

Force directed layout algorithms work by modelling a graph as a system of attractive and repulsive forces between vertices. The positions of the vertices are updated based on these forces, until stability is reached. Stability is not guaranteed so some external bounds are placed on the size of the forces. One of the most common force directed algorithms is

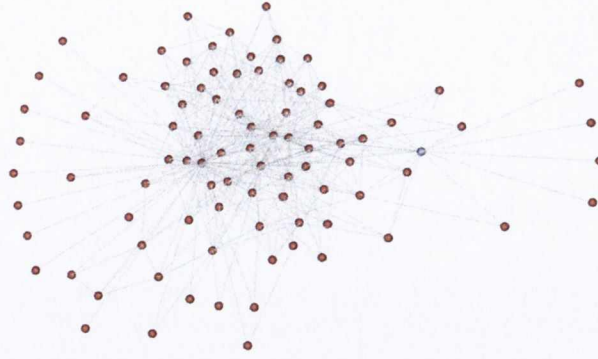


Figure 2.8: Force Directed Layout of a graph containing 91 vertices and 567 edges. Each node is a unit distance across. The ideal distance K has been set to 15. The grid variant version has not been used, so repulsive forces are applied to all nodes regardless of distance between them

the Fruchterman-Reingold force directed algorithm [FR91]. This algorithm works on the basis of having an ideal distance between connected vertices. This ideal distance, usually denoted k is used in the derivation of the attractive and repulsive force between vertices. These attractive and repulsive forces cancel each other out when two connected vertices are the ideal distance apart. The ideal distance can be considered like the length of a relaxed spring between two connected nodes. If the nodes move closer than the ideal distance the spring pushes them apart. If the nodes move further away from each other than the ideal distance the spring pulls them together. The attractive forces, f_a and repulsive forces f_r are defined as follows:

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

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

where d is the distance between a pair of vertices and k is the ideal distance between a pair of connected nodes. The forces acting on each individual vertex are calculated as follows. The total repulsive force for an individual vertex is calculated by the summation of the forces between that vertex and every other vertex in the graph. The total attractive force is calculated by the summation of the attractive forces between vertices and every vertex it is connected to. The final force for a vertex is the sum of the attractive and repulsive forces, and it is this final force which is used to displace the vertex.

$$F_i = \sum_{e_{ij} \in E_i} f_a(d(v_i v_j)) - \sum_{i \neq j, v_j \in V} f_r(d(v_i v_j))$$

Where E_i is the set of all edges connect to the vertex i and V is the set of all vertices in the graph. The algorithm for calculating the forces for a single vertex can be seen in algorithm listing 1.

These calculations are repeated for each vertex over many iterations. An upper bound

Algorithm 1 Algorithm for calculating Fruchterman-Reingold forces acting on a single node

```

v ∈ V
for all u ∈ V do
  if u ≠ w then
     $\delta := v.position - u.position$ 
     $v.displacement := v.displacement + (\delta/|\delta|) * f_r(|\delta|)$ 
  end if
end for
for all u ∈ V do
  if {u, v} ∈ E then
     $\delta := v.position - u.position$ 
     $v.displacement := v.displacement - (\delta/|\delta|) * f_a(|\delta|)$ 
  end if
end for

```

on the magnitude of displacement, referred to as the temperature, is set and decreased at each iteration, resulting in increasingly smaller adjustments in position until the graph is in a stable state, usually determined by when a minimal displacement between iterations is reached. This algorithm is used to lay out undirected graphs; however a directed graph can also be laid out using this technique simply by ignoring the directionality of edges and limiting the number of edges between a pair of vertices to one. One issue with force directed algorithms is the algorithmic complexity of the approach. The calculations of the repulsive forces requires $O(|V|^2)$ operations and the attractive forces requires $O(|E|)$ operations resulting in a per iteration complexity of

$$O(|V|^2 + |E|)$$

per iteration. Given that an instance of the layout algorithm may execute several hundred iterations, performance can be a significant issue, particularly for large sized graphs. Optimisations such as the Grid Variant Algorithm suggested by Fruchterman and Reingold, or some of the multilevel approaches reduce complexity of the repulsive forces to $O(|V| + |E|)$ for most practical use cases.

2.3.3 Multilevel Layouts

Multilevel algorithms are an approach which aim to improve the layout of basic force directed algorithm by accelerating the algorithm and giving a global quality to the placement. The concept was introduced by Walshaw [Walo1] and independently also by Harel and Koren [HK01], who refer to it as multi-scale layout. A key part of multilevel algorithms is the coarsening phase. A coarse version of a graph is simply an abstracted graph of the original, where multiple nodes in the original graph are represented by a single node in the

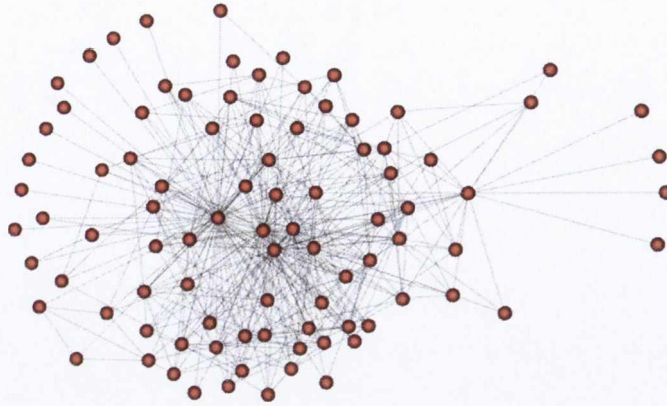


Figure 2.9: Layout of the graph from figure 2.8 using Hachul and Jünger’s FM3 multi-level layout algorithm with a input inter-node distance of 15 (equivalent to a k value of 15). Each node in the image has a radius of 1. The implementation used is the Open Graph Drawing Framework [TDoG13] version of the FM3 algorithm

coarse version. A multilevel layout being performed on a graph $G = (V, E)$, produces a hierarchy of coarse graphs. The graph with the finest level of detail G_0 is the original graph. G_1 is produced by running a coarsening algorithm on G_0 . The hierarchy is generated by repeated coarsening the graph G_i to form G_{i+1} until the minimally sized coarse graph is achieved. The approach to coarsening of a graph is a distinguishing factor between many different multilevel approaches.

Walshaw utilises an approach known as matching to combine pairs of nodes in order to generate a coarse version of a graph. The matching is done by generating a set of graph edges known as a maximally independent edge set. This is a subset of all edges in the graph with the property that that no 2 edges in the set share a common vertex, (i.e. no two edges are adjacent), it is maximal when no more edges can be added to the set without breaking this property. All the pairs of nodes defined by the edges in that set are collapsed to form a single node in the coarse graph. Therefore, a node at each level of the coarsening hierarchy represents two nodes at the level below, except for the bottom level which is the original graph.

GRIP[GK01] generates a coarsened version of a graph G_i from graph (G_{i-1}) by applying a maximal independent set filtration. A maximal independent set filtration is a subset of vertices such that $V \supset V_1, V_1 \supset V_2 \dots V_{k-1} \subset V_k \subset \emptyset$. V_i is a maximal subset of V_{i-1} if the graph distance between each of its elements is at least $2^{i-1} + 1$, i.e no vertices in the subset contain a common edge, and no more vertices can be added without introducing one.

In the coarsening phase of Hachul and Jünger’s Potential Field Based Multi-level Algorithm (often referred to as FM3) [HJ05], vertices are partitioned into what the authors refer to as solar systems, characterising each vertex as sun planet or moon. Each solar system is collapsed to the sun node in the next tier of the coarsening hierarchy.

Frishman and Tal[FT07] use an algebraic technique called spectral partitioning to partition the graph in to clusters of nodes which can be represented as single nodes in the coarser versions of the graph. This is a top down approach to multilevel layouts as opposed to the bottom up approach of maximal independent set filtration. The coarsening of the graph using spectral partitioning requires post-processing to avoid small disconnected clusters, a problem not encountered in the bottom up approaches such as Walshaw's use of vertex matching.

Once the coarsening phase is complete the layout phase applies a layout to each graph in the N hierarchy, progressing from the most coarse level G_{N-1} to the finest G_0 . The choice of layout algorithm, differs between multilevel approaches, but they all use some variant of the force directed model. Part of the advantage of multilevel approaches is that the placement of vertices in a more coarse version of a graph provides a good initial placement for the layout of the next less coarse graph. The most straightforward strategy is that the nodes in graph G_i are initially placed at the position corresponding to their representative node in the more coarse graph G_{i+1} . This is the approach used by Walshaw, but other approaches use different methods. For example Hachul and Jünger's method uses their solar system structure in graph G_i to derive a position for vertices in G_{i-1} .

When laying out a coarse graph as one of the levels of the multilevel layout, care has to be taken so that a layout of the graph G_i does not completely disrupt the layout of the previous more coarse graphs at levels G_{i-1} and above. Walshaw does this by weighting the relaxed spring distance k of the Fruchterman-Reingold algorithm based on the level used by the previous levels coarse graph.

An example of results of a Hachul and Jünger's multi-level layout can be seen in figure 2.9. The results are similar to the basic Fruchterman-Reingold algorithm, seen in figure 2.8. This is to be expected as both are force directed algorithms, the difference is that FM3 offers faster performance and lower algorithmic complexity, particularly for much larger graphs.

A more comprehensive list and evaluation of multi-level algorithms can be found in Bartel *et al.*'s evaluation of several multilevel algorithms[BGKM11] as well as Hachul Jünger's comparison of fast algorithms for drawing large general graphs [HJo6]. Bartel *et al.* also describe many different approaches to graph coarsening and initial node placement in the different levels of graph.

2.3.4 Hierarchy Based

Frequently if a graph has an associated hierarchical clustering (i.e. it is a compound graph), it can be laid out using a hierarchical geometric approach such as a tree layout, a cone tree, a balloon tree or a tree map. These are graphs where the hierarchical nature of the graph clustering is embedded in the geometry of the layout. Tree-maps developed by Johnson and

Shneiderman [JS91] display data hierarchies (not necessarily graphs with adjacency relationships outside of the hierarchy) where items in the hierarchy are displayed as subregions of their parent items in the hierarchy. Sugiyama's layout [STT81] is an early hierarchical layout, under which child nodes are positioned in layers beneath their parents in such a way as to reduce crossings. Cone Trees [RMC91] are three dimensional displays of node hierarchies where each node is laid out such that it is at the apex of a cone, and all of its children in the hierarchy are positioned around the circumference of the base of the cone. Balloon trees such as that used by Holten [Holo6] are essentially a projection of a cone tree layout onto a 2D plane [CK95]. Each low-level cluster is essentially a circular graph. An example of a balloon layout can be seen in figure 2.11. Herman et al cover a variety of tree bases layout in their survey of graph visualisation and navigation techniques (2000) [HMM00].

2.3.5 Algebraic Approaches

Force directed algorithms are not the only approach to graph layout, there are also algebra based algorithms for drawing graphs, such as ACE (Algebraic multi grid Computation of Eigenvectors) [KCH03] which uses an algebraic multi-grid optimisation approach as well as High Dimensional Embedding (HDE) [HK02]. HDE creates a drawing (in a conceptual sense, this is just a positioning of the nodes) in m dimensions (m is defined as an input) and projects it down to a two or three dimensional drawing, for visualisation. The m dimensional drawing is created by selecting m vertices from the graph as pivot nodes which form the basis of the *axes*. The position of each node in the graph along an axis is based on its graph theoretic distance from the pivot node corresponding to the axis. Once the nodes are positioned in the m dimensions, the drawing is projected down to two (or possibly 3) dimensions for visualisation using PCA (Principle Component Analysis), a technique by which multi-dimensional data is reduced to fewer dimensions. One key advantage of HDE is the speed of the algorithm as it has a time complexity of $O(m \cdot |E| + m^2 \cdot |V|)$. Given that m is independent of graph size, complexity only increases linearly with vertex and node count.

2.3.6 Circular layouts

Circular layouts are a restrictive but simple approach to geometrically laying out a graph. The nodes are evenly spaced around the circumference of the circle with the edges passing thorough the interior of the circle, see figure 2.10 for a simple example. As with all graph layouts it is desirable to reduce the number of edge crossing [Pur97]. Obviously the number of crossings in a graph is dependent on the ordering of the nodes around the circle edges, an NP-hard problem to solve [MKNT87]. Many different approaches and heuristics exist to produce better circular layouts. Six and Tollis [ST99] order their vertices so that the

number of edges drawn close to the edge of the circle is maximised. While this does reduce the number of crossings, it is not clear that it will make the graph more legible, as for larger dense circular drawings, many edges close to the edge of the graph will not only cross, but do so with a very acute crossing angle which per Weidong et al. [WSHEo8] makes them more difficult to read. It is possible to maximise the crossing angles in the circle layout using an approach such as that suggested by Nguyen et al [NEHH11], but such a technique would not improve matters much as it does not change the sort order of the nodes in the circle, and can also result in a slightly misleading visual clustering of the nodes in the circle.

Baur and Brandes [BB05] developed an approach to reducing crossing within circular layouts, which consists of an intelligent initial placement of nodes, followed by a circular sifting approach, which rotates the position of a node around the circle circumference, and assigning its final position as the one which resulted in the least number of crossings. Ganser and Koren [GK07] have developed techniques which lower the edge density within a circular layout. They use a three pronged approach, consisting of ordering nodes in such away that edge lengths are reduced, adapting edge bundling for use within a circular layout, and routing edges external to the circle using

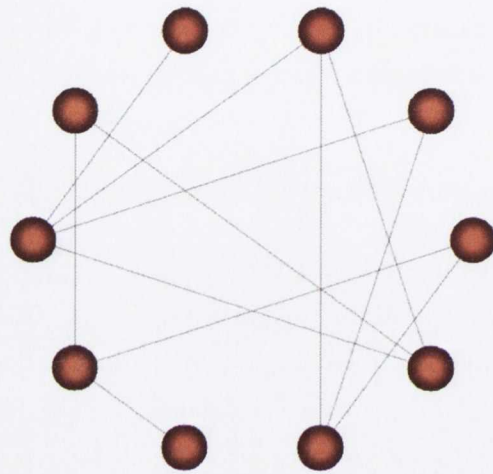


Figure 2.10: A simple circular layout of a 10 node graph

curves. Circular graphs can also display clustered hierarchies, as can be seen in figure 2.12 where the clustering hierarchy is conveyed by the positioning of nodes and the grouping of edges.

It is possible that a graph consists of multiple circular layouts. For example different clusters within a graph could be laid out as circular sub-graphs, as done in a balloon tree layout, or as with approaches such as Topolayout[AMA07] circle layouts can feature as one of multiple approaches utilised in graph layout. When multiple circle layouts are used within a graph, inter-circle edges and their crossings should also be considered. Crossing can be reduced by rotation of circles as well as ordering of the constituent nodes within the circle. Many approaches model physical torque, with the edges to other circles applying a rotational force on the source circle. Essentially an energy function is minimised to provide a good rotation to a circle. Examples of this include the GEM layout of Frick *et al.* [FLM95], as well as Symeonidis and Tollis [ST04], who use a polar coordinate based form of force directed layout.

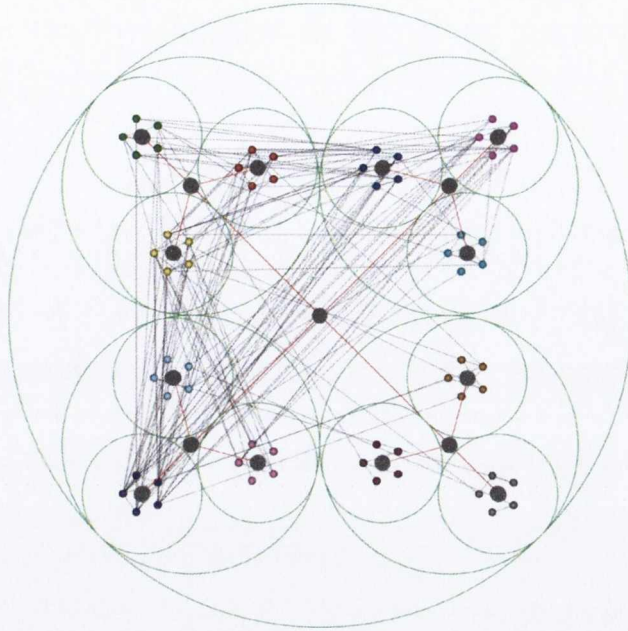


Figure 2.11: An example of a balloon tree layout of a 60 node graph. The hierarchy nodes are coloured black, connected by the red lines. The hierarchy levels are also circled in green to clarify the hierarchy structure .

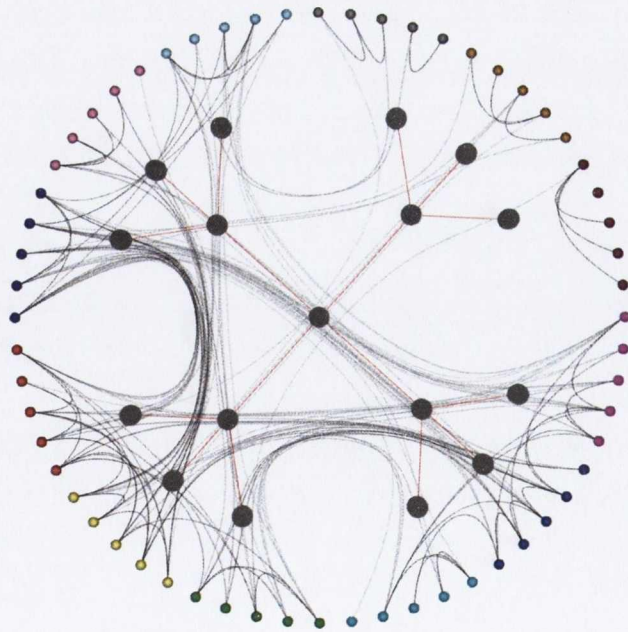


Figure 2.12: The same graph as in figure 2.11 under a clustered circular layout. The grey hierarchy nodes are shown for illustrative purposes.

Other approaches

Other approaches have included using space-filling curves as a framework for vertices [MMo8] and genetic algorithms [BBoo] and topology based layout [AMAo7]. Space filling curves [MMo8] position nodes along a curve designed to take up the full graph display space. The order of the nodes along the curve is decided by an ordering function, which is dependent on a good clustering. The primary advantage of this approach is the speed of layout, which makes it much faster than the force directed approaches, particularly for exceptional large dense graphs. Barreto and Barbosa's [BBoo] genetic layout approach uses graph aesthetics such as edge crossings and vertex distances to select the input layouts used to create successive generations or graph layout. Archambault et al's Topolayout [AMAo7] is an innovative approach for the layout of graphs. It decomposes a graph using topological features, to form a hierarchy. Topological features are graph structures such as cliques (a set of nodes that are fully connected to each other), connected components, and trees. Topolayout uses other layout algorithms, such as a basic circular layout and High Dimensional Embedding to layout the various topological features at lower levels of the hierarchy, depending on which layout is more suitable for the feature. The authors also increase comprehensibility by applying edge crossing reduction techniques to the identified features in their graph.

2.4 Graph Visualisation Evaluation

Evaluation is a challenge in the field of visualisation, be it scientific visualisation, information visualisation or graph visualisation. Frequently, graph visualisation paper authors, particularly if the topic is the visual presentation of a graph, have to rely solely on displaying images of their technique. For example Holten [Holo6, HWo9] and Cui [CHH⁺o8] rely on the visual presentation of their edge bundling techniques as part of their evaluation. Such an approach is necessary, particularly for a technique as visual as edge bundling, but it is limited in the number of cases that can be shown within the space of a paper. Chen [Cheo5] describes the lack of "*intrinsic quality measures*" as one of the unsolved visualisation problems of information visualisation.

The notion of a quality measure that is intrinsic to information visualisation, and hence not dependent on the application or subject matter or any external reference point, is important as it allows a consistent evaluation of quality between an evaluation based on user assessment and one based on an evaluation metric. In graph visualisation such intrinsic measures, often referred to as aesthetics, do exist. In addition to intrinsic metrics of a visualisation, user performance is important as the ultimate goal of visualisation is to aid human understanding. User performance can only be measured by empirical experiment, although it can be shown to correlate with some metrics. For example Purchase's work [Pur97, Pur98, WPCMo2] investigating graph aesthetics does so using user experi-

ments. Empirical evaluation can be done through low level abstract tasks such as indicating the distance between two vertices in a graph are connected by two or 3 hops [WM05], or through higher level domain specific tasks such as considering where a new web page should be added to in a website's directory structure [RCMCoo].

When evaluating the real word effectiveness of a clustering or layout technique it can be difficult to subjectively quantify its effectiveness without a high level task. This section is concerned with describing how intrinsic measures are used and evaluated as well as general evaluation techniques.

2.4.1 Evaluation Graphs

When evaluating a technique related to the presentation of a graph, e.g. evaluating a graph layout or a visual effect such as the use of colour, it is obviously necessary to chose a graph (or graphs) that will be the basis of the experimental evaluation. The choice of graph has a very significant role in the experiment. Graphs vary in size structure and density depending on what they are modelling. The choice of graphs should also be suitable for the visualisation technique being evaluated. Preferably, it should allow the evaluation of the efficacy of the technique under a range of experimental conditions.

In her 2004 paper on the challenges of information visualisation (not specifically graph Visualisation), Plaisant [Plao4] suggests the creation of repositories of data and tasks as the next step in providing a solution to the problem of information visualisation evaluation. In the conclusions of their 2011 state of the art survey on the visual analysis of large graphs von Landesberger *et al.* [vLKS⁺11] comment in their conclusions on the need for more taxonomies for aspects of visualisation such as tasks and measures for quality, as well as benchmarks for comparing techniques and *“although several taxonomies and sample data sets exist, a more broader scope of theory and data aspects is needed”*. While more and more data sets have become available, it is not always clear which data set is the most suitable for testing a specific data visualisation approach. A wide range of task specific data sets would help standardise graph evaluations across different techniques.

In their evaluation of large graph layout algorithms Hachul and Jünger [HJo6] used 11 graphs from real-world graph sets. These real-world graphs consisted of a subset of the AT&T graph Library [AT12], a subset of Walshaw's graph collection [Wal12], and a single social network graph of 2113 people.

Using a real-world graph to test a specific visualisation technique is limiting as there is no flexibility in graph parameters. Unless the visualisation technique is only targeted at a very specific data set, it may not be enough to target it at such a limited range of data. In order to evaluate graph visualisation it is often useful to procedurally generate graphs, as the characteristics of the graph can be defined beforehand in such a way that the technique is tested under a variety of conditions.

There exist many procedural approaches to generating graphs, and these have been used in the past to test graph layout algorithms. Hachul and Jünger [HJ06], in addition to the real world graphs set, created many graphs using simple procedural approaches. These result in graphs which display regular patterns which are apparent by visual inspection (depending on layout of course). These patterns are reflected by the names given by the graph authors: the snowflake graph, the flower graph and the Sierpinski graphs (based on Sierpinski triangles).

Apart from loading real world graphs from external libraries, real world graph data can be generated by parsing data sets such as program source code, website links or data-base relations. In chapter 3 we present some examples of graphs data created by parsing data from Wikipedia. It is very useful to be able to randomly generate large sets of small world graphs to analyse the effectiveness of a particular algorithm on graphs with a wide range of properties (such as size, edge density, graph clustering coefficient, level of randomness). When procedurally generating graphs characteristics can be determined as an input to an algorithm which can still generate graphs with some level of randomness. This can allow for an algorithm to be tested against a wide range of graphs with different parameters and also to be tested against multiple graphs with the same parameters.

Random Graph Generation

One of the earliest common methods of generating graphs is the Erdős-Rényi model [ER59] of random graphs, also known as the Poisson Random Graph. While this is one of the best known random graph models and has provided many insights into the field of network graph theory, it does not accurately reflect the structure of many real world graphs in terms of edge distribution. It may not also provide graphs suitable for the evaluations of different visualisation approaches, e.g. if an edge routing evaluation depends on different levels of edge connectivity between clusters. As mentioned by Lancichinetti *et al.* [LFR08], cluster size edge distribution vary in real world graphs. In section 5.1 we will see an example of graphs generated for the specific task of evaluating edge routing in a compound graph utilising a distribution of edges not found in a simple Poisson Random Graph.

In some cases, such as evaluating clustering algorithms, it may be desirable to know in advance what the optimum clustering of a specific graph is. However graph partitioning is an NP complete problem. So for a completely randomly generated graph finding the best possible clustering requires analysing all possible permutations of clustering for that graph. As graph get larger this becomes more and more impractical. Moussiades and Vakali [MV09] propose an approach for generating random graphs where optimal clustering is known. This is useful when a naturally occurring clustering or community structure (as demonstrated in Zachary's karate club example) is desired from the input graph. However, if extra constraints are required, such as a specific characteristics, e.g. a high average

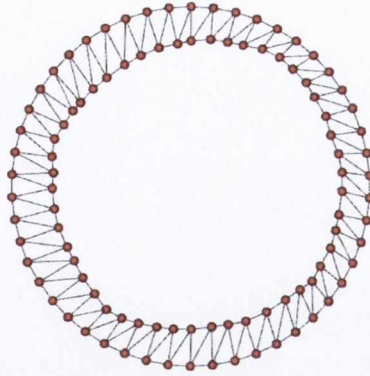


Figure 2.13: A ring lattice of 100 nodes each with a degree of 4, the starting point of the Watts and Strogatz' small world generation algorithm

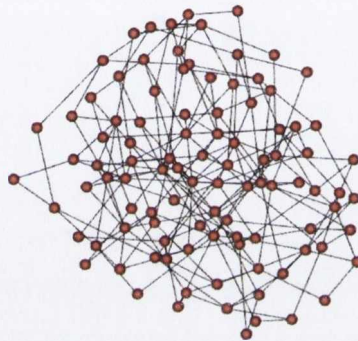


Figure 2.14: A random graph equivalent to that in figure 2.13, generated using Watts and Strogatz' small world generation algorithm with an input probability of 1.0 and laid out using a force directed algorithm. The graph has an average path length of = 3.37636 and an average local clustering coefficient of 0.012

local clustering coefficient, or some node based constraints, e.g. limiting node degree, such an approach may not be possible.

Random Small World Graph Generation

The small world graphs described in section 2.1 demonstrate characteristics of real world graphs. It is possible to procedurally generate small world graphs. We use Watts and Strogatz' approach for creating small world graphs [WS98] for use in the evaluation of our agglomerative clustering in chapter 3. In generating these graphs we have control over the graph size, edge density and level of randomness, allowing us to create graphs of various degrees of "small world-ness".

The approach starts with a ring lattice graph of n nodes where every node has k links connecting it to its neighbours (i.e. every node has a degree k , resulting in $|E| = k|V|$). This can be considered a fully structured graph, where a completely random graph could be considered full unstructured. Such a lattice with 100 nodes and a degree of 4 is displayed

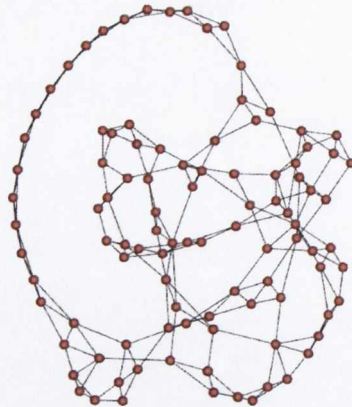


Figure 2.15: A small world graph equivalent to that in figure 2.13, generated using Watts and Strogatz' small world generation algorithm with an input probability of 0.1 and laid out using a force directed algorithm. The graph has an average path length of = 5.53879 and an average local clustering coefficient of 0.396

in figure 2.13. We consider a specific vertex and consider the edges connecting it to one of its neighbours in a clockwise sense. Each edge is rewired to a randomly selected neighbour, with a probability of p . Duplicate edges and loops are forbidden. Each vertex in the graph is processed in a clockwise order. The input value of p has a large impact on the resulting graph. For a value of $p = 1.0$ the result is a randomly wired graph, as seen in figure 2.14. For low values of p (e.g. $p = 0.1$) the result is a small world graph as displayed in figure 2.15.

The combination of the real world-like characteristics of these procedurally generated graphs, as well as the ability to control the edge density, graph size and level of structure makes these graphs very useful for performing experiments on the efficacy of clustering algorithm. In addition to this the non-deterministic aspect of their generation allows for a variety of graphs to be generated for the same input parameters.

2.4.2 Graph Aesthetics

One approach to evaluate a graph layout is based on the aesthetics of the resulting layout. Aesthetics in this context refers to measurable attributes which reflect the quality of a resulting layout.

Reduction of edge crossing has long been identified as a desirable graph layout aesthetic for 2 dimensional graph layouts. Crossing reductions has been used as an approach to heuristically improve algorithms for some time. For example Sugiyama [STT81] states that "*The greatest difficulty in tracing paths is line crossings*" and he uses crossing reduction as a step to improve the layout of hierarchies.

Many aesthetics have been postulated, often as part of the goal of a particular graph drawing algorithm. These aesthetics have included graph symmetry [Ead84], minimising the number of edge crossings [STT81], minimising the number of bends in edges [Tam87],

and path continuity [WPCMo2]. The efficacy of various aesthetics has been evaluated in user experiments and some have shown to be more important than others. Purchase's work [Pur97] has demonstrated that edge crossings are by far the most important aesthetic, impacting both user response time and accuracy, with symmetry and edge bends being of lesser importance but having significant results. Later work with Ware *et al.* [WPCMo2], showed that continuity of paths (i.e. keeping path between more distant nodes that traverse multiple edges as straight as possible) is also an important factor.

For many graphs (those which are not planar) crossings cannot be omitted altogether, Huang and Eades [WSHEo8] demonstrated that maximising the crossing angle so that intersecting edges are perpendicular reduces the negative effects of edge crossing. Huang and Huang [HH10] have followed up on this work and demonstrated that 38% of variance in performance is attributable to crossing angle and the rest is attributable to the crossing number of the graph (in cases where there was a performance variance). So where further crossing minimisation is not possible, maximising crossing angles can be used as a secondary aesthetic.

More recent work by Purchase *et al.* [PPP12] has indicated that when users manually create graphs, they use edge crossing reduction as an aesthetics and also align nodes and edges to an underlying grid.

2.4.3 Evaluating User Performance

To fully evaluate the effectiveness of any graph algorithm or suggested aesthetics, its benefit to the end user needs to be measured. Different tasks to evaluate user performance, and implicitly graph comprehensibility, have been suggested.

For experiments considering the impact of graph aesthetics [Pur97] as well as the impact of various graph layout algorithms [Pur98], Purchase utilises a path tracing task (specifying the shortest path between two nodes), as well as questions concerning the graph structure, such as "how many nodes must be removed to disconnect two highlighted nodes?" These tasks are designed to reflect the experiment participants understanding of the relational nature of the graph being displayed.

Ware *et al.* [WPCMo2] use a path tracking task in their evaluation of the aesthetics of graph edges. This includes features such as edge crossings, path directness and average geometric edge length. For each displayed graph the user is asked to determine the shortest path distance between two nodes and is given an option of three, four or five to choose from as their answer. The time to answer and error rate is recorded for analysis.

In Ware and Mitchell's [WMo8] experiments on the comprehensibility of 3D graph visualisation, they use a path tracing task where the paths distances were limited to either two or three hops. In Huang *et al.*'s [WSHEo8] evaluation of the effect of crossing angles of a graph, the task is again path tracing with paths of length from four to seven hops

long. In Huang *et al.*'s study of graph evaluation [HEHo8], the authors used path tracing as part of an eye-tracking study, as well as questionnaires which were used to evaluate user's perception of node importance and node grouping.

These above experiments involve low level relational tasks which are not domain specific. For a broader system evaluation, experiments are based on high level, interpretive tasks. For example, in Ridsen *et al.*'s [RCMC00] study on ease of use of 2D and 3D visualisation of web content, they used a search task related to directory management of a website with all of the participants of the experiment having technical experience. Assessing performance at high level tasks which require domain knowledge is difficult. In Purchase *et al.*'s study on comprehension of UML diagrams [PMCC01], the study was limited by the fact that university students, given a tutorial on UML diagrams, were used as subjects rather than experienced software engineers.

High level tasks are useful for evaluating visualisations designed for specific purposes. However it is difficult to generalise any results across general node-link diagrams as the high level task itself, or the visualisation application domain, may have a strong influence on the graph characteristics. For example the directory structure for Ridsen *et al.*'s experiment was a tree structure, so it is not clear if their results will generalise to other types of graph. Given that domain knowledge also plays a significant role, it may be better to use a low level task, such as path tracing, that can be generalised to a higher level task. Table 2.1 provides a summary of tasks used in previous experiments.

Graph Properties in Previous Experiments

Tables 2.1 shows the size of graphs in previous works. While the largest graph was used by Ridsen *et al.*, the experiment was performed on a single graph, used for a high level task, as opposed to a series of graphs or a large size with varying different properties. Therefore it is difficult to see how the scale characteristic contributes to the result.

Clearly the choice of graphs for a study reflects the goals of the study. Ware and Mitchell [WMo8] utilise 4 different graph sizes, with the same varying edge distribution between nodes for each size, reusing the same graphs under different viewing conditions for their experiment. This reflected the aims of the experiment, which concerned the impact of stereoscopic three dimensional viewing as graph size increased. Other evaluations of graph aesthetics such as [Pur98, Pur97] use only a single graph in their evaluation. In these experiments Purchase evaluated very specific effects of graph layout, and introducing a larger range of graphs may have introduced confounding factors. Determining the range of graphs to be used as inputs is a difficult issue. If a wide variety of graphs are used, there is an increased risk of introducing factors external to those under investigation, which may impact results. If only a small number, or single graph is used, results may not generalise to graphs with different properties to the ones used for an experiment.

Study Authors	Year	Task	Goal
Purchase	1998	Path tracing, Node removal, Edge removal	How long is the shortest path between two given nodes? How many nodes must be removed to disconnect two highlighted nodes? How many edges must be removed to disconnect two highlighted nodes?
Purchase	1997	Path tracing, Node removal, Edge removal	How long is the shortest path between two given nodes? How many nodes must be removed to disconnect two highlighted nodes? How many edges must be removed to disconnect two highlighted nodes?
Ware and Purchase	2002	Path tracing	How many hops between highlighted nodes?
Mitchell and Ware	2008	Path tracing	How many hops between highlighted nodes?
Ware and Franck	1996	Path tracing	Does a path exist between nodes?
Risden et al.	2000	Node Insertion	Using the graph to aid in the addition of new nodes to a hierarchy
Huang et al.	2008	Path tracing	Determining the number of links in path, measuring the time for a correct answer, ignoring incorrect ones

Table 2.1: User tasks in previous graph based empirical studies

Study Authors	Year	Node Count	Edge to Node Ratio	Layout
Purchase	1998	16	1.69	Multiple type of layout
Purchase	1997	16	1.69	Graph layout was derived from the aesthetics being evaluated
Ware et al.	2002	42	1.0 - 5.0	Force Directed With Simulated Annealing
Mitchell and Ware	2008	33, 100, 333, and 1000	Max degree of 5	A modified version of Force Directed Fruchterman Reingold layout
Ware and Franck	1996	21-291	1.333	Random Layout
Risden et al	2000	1200	unspecified	Hierarchy combined with list layout for the purpose of 2D and 3D comparison
Huang et al	2008	N/A Full graphs were not used, only small graph-like stimuli	N/A	N/A

Table 2.2: Graph sizes in previous graph based empirical studies

2.5 Edge Routing

Keeping edge crossing to a minimum is important for two dimensional graphs. However for a graph of any considerable size and density, there will be a significant number of edge crossings, and maximising edge crossing angles may be ineffective due to constraints of the graph layout or the sheer number of edges. Therefore, a huge number of edge crossings is often unavoidable in a very large dense graph. Example subjects of such graphs are complex computer programs and file systems [Holo6, BDo7], graphs representing air traffic [CHH⁺08, HW09] (using a map and geometrically fixed nodes). Other approaches are necessary to reduce the amount of clutter introduced by a larger volume of edges and clarify the paths taken by edges. Edge bundling is a recently popular technique by which edges are grouped together and drawn using curves which share a common path from their source to destination, if they have a common geometric destination or a common conceptual basis. It is often used in dense graphs where straight edges become indistinct due to the clutter cause by the number of edges as clutter reduction techniques such as the reduction of edge crossings are impractical or ineffective.

2.5.1 Edge Bundling

The edges which contribute to a grouping of edges, referred to as a bundle, may be determined by graph structure, such as a hierarchy [Holo6, BDo7], or the geography of nodes [CHH⁺08, BDo7, HW09]. Once the edges for a specific bundle have been defined, they are drawn using curves known as splines. In a spline the individual curve points are calculated using a polynomial function which interpolates values based on an input set of control points. There are many different type of splines, however the most commonly utilised splines for edge bundling are Bézier curves and B-splines. Other types of curves such as β -splines have be tested [Holo6] but never adopted. Bézier curves are significantly easier to calculate than B splines. The calculations consist of simply multiplying a geometry matrix, defining the control points, by a Bézier basis matrix, which defines the polynomial function. This allows long curves to be created using multiple Bézier segments with overlapping control points. B-splines require a more significant amount of calculation, as a different basis has to be worked out for different line segments using the Cox De Boor recursive algorithm. However B-splines result in a much higher level of control of the resulting curve and this allows a much more flexible routing. B-splines are a better approach, as long as graph size does not result in their computational complexity being an issue. Different approaches to bundling are characterised by how edges are chosen to be bundled together, which spline type is used to draw the edges as curves, and how the control points are defined for the curves.

Hierarchical Edge bundling

Developed by Holten, the hierarchical edge bundling algorithm [Hol06] is one of the most effective forms of edge bundling. It is, however, limited to compound graphs that are laid out in the form of a hierarchy. In the basic form, the nodes of the graph are positioned using a hierarchical layout, and hierarchy node positions are utilised as control points for drawing the splines. Piecewise splines are used to draw the edges between nodes. These edges are referred to as adjacency edges to discern them from the edges of the hierarchy (which are not drawn). The set of control points for the edges is the set of nodes on the shortest path between the source and target nodes in the hierarchy. B-Splines are chosen as the spline representation. Beta splines and Bézier curves were also considered, however they were found lacking. Bézier curves did not provide the desired level of control and beta splines required extra process to achieve results which were achievable with regular B-Splines and an external straightening parameter. This straightening parameter is used as two splines with similar sets of control points will overlap, however by adjusting the straightening parameter the splines are drawn at slightly different positions. The straightening parameter adjusts the control points of each individual spline that makes up a bundle based on their position in the curve and the relative position of the initial and end control points. The initial and end control points are the positions of the source and destination nodes of the edge. For a curve using N control points, beginning at P_0 and ending at P_{N-1} , the straightened position P'_i of control point P_i is given by:

$$P'_i = \beta \cdot P_i + (1 - \beta) \left(P_0 + \frac{i}{N-1} (P_{N-1} - P_0) \right)$$

β is the straightening parameter. It lies in the range $[0, 1.0]$, with $\beta = 0.0$ resulting in straight lines between curves and $\beta = 1.0$ resulting in tightly bundled overlapping curves. Holten also uses alpha blending of the edges to help convey the density of bundles, and to help pick out bundles where large numbers of edges overlap. The edges of the visualised graph are directed edges with the direction indicated by a changing colour gradient which may also have an impact on the readability, as a differing change in gradient between edges may help distinguish them. The results of Holten's edge bundling approach using two different layouts can be seen in figure 2.16.

In later work, Holten and Van Wijk [HVW08] utilised hierarchical edge bundles as a technique to aid in the visual comparison of hierarchically organised data. The edge bundles visually emphasize the splits, joins, and relocations of sub-hierarchies between data sets being compared.

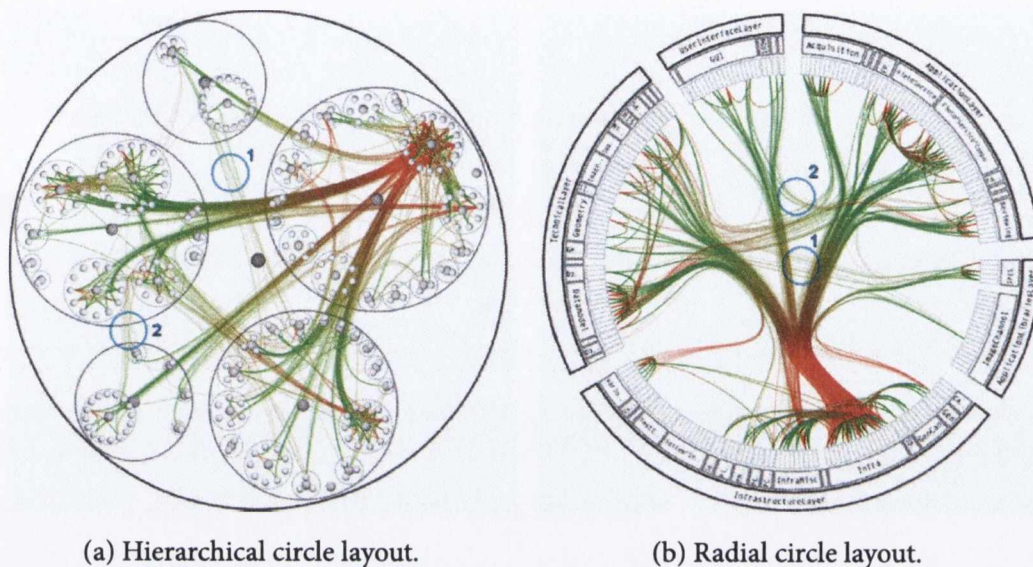


Figure 2.16: Images taken from [Holo6] visualising a graph showing the structure of a software system, using hierarchical edge bundling

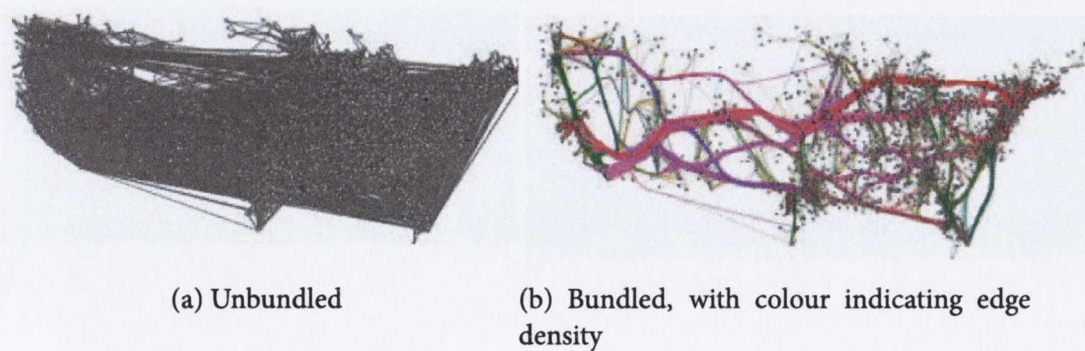


Figure 2.17: Images taken from [CHH⁺08], visualising a graph showing migration between states in the U.S., using Cui and Zhou's geometric edge clustering

Geometric Edge Clustering

Cui *et al.* have developed a purely geometric approach to bundling edges [CHH⁺08], following on from previous work by Qu *et al.* [QZW07]. This approach involves generating a control mesh for the graph. The control mesh is generated by a *Delaunay Triangulation*, a technique by which geometric space is divided into a set of triangles, based on a set of input points. The resulting triangles have the property that no point in the inputs set lies within the circum-circle of any of the triangles generated.

A set of input points can be selected by the user or determined automatically. These points are not nodes within the graph but points which are to form the vertices of the Delaunay triangulation (edges to be included can be specified too). In the case where the mesh is determined automatically the graph is subdivided into a grid and for each cell in the grid the number of graph nodes and links passing through are calculated. The authors then store the direction of each edge in a feature vector and perform a Kernel Density Estimation. This gives them a probability curve of direction for each grid square. If there is a strong probability of all curves going in a specific direction then this is the primary direction assigned to the square. Grid cells are then merged with cells containing a similar primary direction (i.e. the difference between orientations is within some threshold angular distance). Cells are merged into a larger region until the difference between all primary directions is beyond some threshold specified as an input.

Once the mesh is generated actual clustering of the edges begins. For each edge of the mesh, one or more control points are generated. The input to the smoothing is the points at which the graph edges intersect the mesh triangle edges. The authors use K-means clustering of these intersection points to determine the actual control points to be used (if only one control point is desired this is in effect the average position). This results in noticeably tight bundles (see figure 2.18). However, this can be partially rectified by using a higher resolution of grid and adjusting the threshold angle for merging grid squares. Due to the kernel density estimate and averaging (and K-means clustering) some edges might be periodically be bundled in the wrong direction resulting in difficult to follow meandering links in the final visualisations. To overcome this problem, the authors perform local edge smoothing. The smoothing is local as it only considers alternate paths for the edge within nearby triangles of the mesh. The authors determine which edges need to be smoothed by examining a metric which is a combination of the bundled edges angular difference for the original straight edge and Euclidean distance difference for the straight edge. The weighting of the two quality attributes is decided by the user. Using this metric, poor quality links are identified and an alternative is sought by searching the mesh triangles that the edge passes through, as well as some of the neighbouring mesh triangles.

It can be seen from the above that this approach is algorithmically more complex than hierarchical edge bundling, requiring extra smoothing and a less intuitive generation of con-

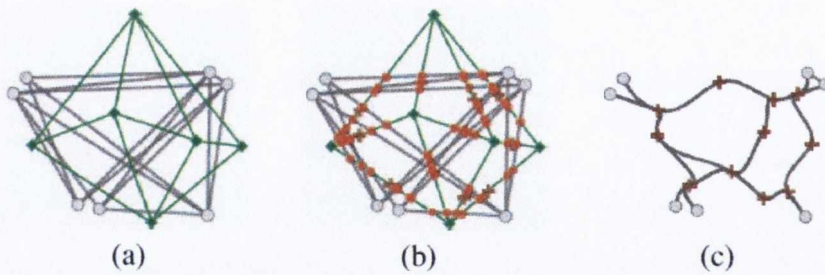


Figure 2.18: Example given by Cui[CHH⁺08] of Edge clustering by control points: (a) a graph with a control mesh, (b) the intersections and the control points and (c) the merged graph.

control points. The results of Cui and Zhou’s approach can be seen in figure 2.17, on a graph representing migration patterns between states in the US. This type of graph, where node positions are fixed absolutely (as they represent states on a map), benefits the most from Cui’s approach as the clustering hierarchy necessary for hierarchical edge bundling is not present.

Other Geometric approaches

Zhou *et al.* [ZYC⁺08] build on Cui’s approach allowing for a hierarchical bundling of edges (see figure 2.5.1 for an example) using an energy based approach to determine the control points for the bundled hierarchy edges. The resulting images bundle edges tightly and care has to be taken about the reading of the graph, as the bundled edges could be misinterpreted as a visualisation of a hyper edge in a hyper graph. The authors admit that sometimes the edge direction is not always entirely clear. This approach differs to [Holo6] and [CHH⁺08] as when edges share a common path they can overlap entirely for some of that path.

Lambert *et al* [LBA10b] provide an approach that is slightly similar to Cui *et al.*. It includes a similar spatial subdivision, but instead of Cui *et al.*’s grid it uses a hybrid quad-tree / Voronoi diagram approach. The grid is also used to route edges. The edges of the grid act in a similar manner to the Delaunay triangulation performed by Cui *et al.*, however the strength of these edges is determined by calculating a shortest path algorithm on the original graph and calculating how many edges from the original graph cross each grid edge. The authors have also extended their edge bundling

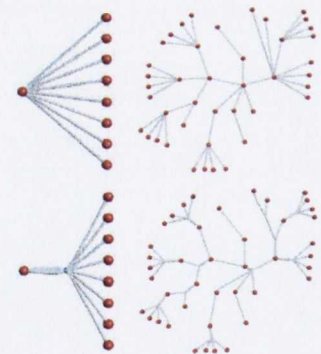


Figure 2.19: Example image taken from [ZYC⁺08], showing the effect of Zhou’s Hierarchical edge bundling. The graphs are unbundled in the top row and bundled in the bottom).

into 3D [LBA10a], displaying 3D edge bundles running across a spherical geographic map of earth. The authors also use visual techniques such as bump mapping to aid in the visual display of the edges.

Force Directed Edge Bundling

Holten and van Wijk [HW09] also developed an edge bundling approach which has no need for a hierarchy, as edges are routed using a force directed algorithm. In this approach, edges are subdivided into points that interact with each other in a manner similar to a force directed layout. This approach leads to very high levels of bundling so edge compatibility measures are used, to determine which edges should be bundled together. Edges are bundled together only if they are compatible in orientation and in length. The resulting edges are then smoothed using a Gaussian kernel to adjust the position of the internal points of the edges. The strength of the bundling can be adjusted in manner similar to the bundles in [Holo6]. The resulting colour of the edges is determined from a gradient scale related to the number of edges intersecting a specific pixel.

Other Occurrences Gansner and Koren [GK07] use edge bundling to reduce clutter as part of their improved circular layout. Edges are bundled tightly, merging to a single line, leaving the connectivity of node pairs to be inferred by the user based on the order of nodes at either end of the bundle. The approach used for winding roads [LBA10b] has also been expanded to generate 3D edge bundles for use with spherical geographical layouts [LBA10a]. 3D Edge bundling also features as a component of Balzer and Deussen's level of detail visualisation of clustered graphs [BD07]. Edges are grouped together based on inter cluster connectivity and divided into segments which are routed together algorithmically. Bézier curves are utilised to smooth the resulting edge segments, but are not utilised as part of their routing. Pupyrev *et al* [PNBH12] also use an approach similar to the mesh based approach described previously, creating a grid graph for edge routing. This routing graph is used to route edges so that they do not obscure nodes, as opposed to Cui *et al.* who use the mesh to generate control points. Bundles are drawn using Bézier curves, which are evenly spaced apart (rather than using a straightening parameter), and ordered in such a way that edged crossing are reduced. Luo *et al.* [LLCM12] also utilise a geometry based approach using spatial partitioning to bundle edges in order to reduce edge ambiguity. Their approach is interactive and on demand. Bundling only takes place in areas of a graph where a user desires it, to reduce edge ambiguity and improve readability.

Bundling Interactivity

Approaches such as Cui's geometric edge bundling and Holten's hierarchical edge bundles produce an edge routing which is global for the whole graph. The edge bundling is in-

teractive in the sense that parameters can be altered and the edge bundling redone. Other techniques such as Edgelens [WCG03] distort edges in a local fashion which can be considered related to bundling. Edge lens distorts edge around a focal point (the lens) using edge curvature to allow the user to see any obfuscated data, such as nodes, or to separate closely routed edges. Edge lens derives from edge plucking [WCo7], which is a technique which allows users to interact with edges directly, moving them using the cursor (like plucking a guitar string) in order to clarify any ambiguity about their path. Riche *et al.* describe an interactive local bundling approach referred to as link magnets [RDLC12] as part of their overview of interactive edge routing approaches. This approach requires user interaction to position “magnets” in the graph which distort edges to allow the user to see graph aspects more clearly.

2.6 Three Dimensional Stereoscopic Vision and Graphs

Stereoscopic viewing of objects has been known about since the Victorian era. It was identified by Charles Wheatstone in 1838 [Whe38], who also developed the Wheatstone stereoscope (and coined the term stereoscope). A stereoscope is a device which, when used to view two images side-by-side, provides the illusion of depth. The images must correspond to the left and right eye view of a scene. During World War two, as part of operation Crossbow, allied reconnaissance planes took multiple photographs of the landscape of Nazi long range missile sites in Europe.

These were then viewed using a stereoscope, of the design shown in figure 2.20. The use of stereoscopic 3D allowed the allied analysts to measure the height of new structures at the site.

2.6.1 Stereoscopic Display of Graphs

Much research has been done in visualising graph structures using three dimensional displays [WF96, WMo8, SM93, HHL10]. Displaying a graph in stereoscopic 3D usually requires specialised hardware. Stereoscopic vision depends on delivering a different image to each eye. Many approaches require the user to wear glasses, while a screen displays images for both the user’s left and right eye.



Figure 2.20: A World War 2 stereoscope and Case.

Stereoscopic display approaches

Early promising results on visualising network graphs in three dimensions was produced by Ware and Franck [WF96] using active shutter glasses. Active shutter 3D glasses require a display which alternates between rendering the left eye view and right eye view for each frame. The active shutters of the glasses are synchronised with the display frequency so that each eye only sees the frame that is targeted at that eye. Passive glasses systems are usually projector based and require overlapping polarised projections of each eye image onto the display. The passive glasses act as filters so that each eye only sees the image intended for that eye. Most contemporary three dimensions cinema displays utilise passive 3D, which many home television and computer three dimensional displays utilise active 3D. Previous generations of shutter glasses were often bulky and quite uncomfortable, however more recent consumer oriented products have improved the form factor and reduced the size of shutter glasses. It is also possible to display a three dimensional stereoscopic image without the use of glasses. Auto-stereoscopic displays [Del05, HHL10] are a more recent technology at consumer level, however they very often require a large number of images to be rendered for each frame, compared to the usual two for most other stereoscopic visualisation techniques. This can significantly damage interactivity as a high level of scene complexity results in a greatly reduced frame rate.

Anaglyph 3D is an alternative to using the complex hardware required for shutter glasses. This approach uses red/cyan or similarly coloured glasses on a regular display but with specially altered renderings. Anaglyph stereo has been used for three dimensional experiments, for example van Shooten *et al.* [vSvDZS⁺10]. However the distracting nature of the coloured lenses as well as the loss of colour as an information channel makes anaglyph stereo a last resort and only if no other means of stereoscopic vision is possible.

In [WMo8], Ware and Mitchell used an adaptation of a Wheatstone mirror stereoscope to allow for higher resolution stereoscopic display. This device, rather than requiring glasses, requires the user to look into an apparatus where two mirrors reflect images from a pair of high resolution displays, one for each eye. While allowing for high resolution image that reduces artefacts such as image ghosting, which affect other approaches, this apparatus is not practical for everyday user interaction with a display.

Head Tracked displays often go hand in hand with stereoscopic displays and add an extra level of immersion for users [WAB93, WF96, HHL10]. When head tracking is used the images displayed are updated depending on the position and orientation of the users head. Effectively the virtual camera rendering a scene is controlled by the users head motions. This adds motion parallax which adds to the perception of three dimensionality in addition to the stereoscopic effect.

One final option is a head mounted display, which is essentially a large pair of glasses

with a separate display for each eye as well as a means of head tracking. These devices generally require complete immersion in a visualisation by a user and limit interactivity with an external stimuli and hence are quite impractical for many real world visualisations.

Stereoscopic and motion depth cues

Both stereoscopic and motion cues provide depth information to the user. However the level of contribution of each, as well as the impact of combining both cues, is not consistent across all previous work comparing the impact of stereoscopic and motion cues on user performance.

Sollenberger and Milgram's experiments [SM93] showed that both motion cues and stereoscopic display improve user performance at a tree based path tracing task. The motion cues were based on rotation of the graph structure back and forth, and their task was focused on selecting the correct root node for a tree containing a highlighted leaf node. The authors state that the motion cues improve user performance more than the stereoscopic cues. Ware *et al.* [WAB93] conducted a similar experiment, except the rotation was the result of head tracking, coupled with the stereoscopic display, and found similar results.

In Ware and Franck's 1996 paper [WF96] users performed head tracked path tracing tasks on randomly laid out three dimensional graphs. Once again motion is better, but stereo also has a strong effect. In later work, from Ware and Mitchell which re-visited the topic of stereoscopic display of graph visualisations [WM05, WM08], the authors used a spring embedder layout as opposed to a random layout, as well as a much higher resolution display than previous experiments. They discovered a larger positive effect from use of 3D depth cues (both motion and stereoscopic) than previously noted. In particular, that the viewer could comprehend larger graphs more easily. There was no significant difference between motion and stereo cues for novice users (14 of which were used in the experiment). In their work on path tracing tasks using depth with multi-view (i.e. auto-stereoscopic) displays Hassaine *et al's* [HHL10] results show that when comparing motion cues, in this case motion parallax as a result of head tracking, combined with stereoscopic cues, stereoscopic depth cues play a larger role in user understanding of the graph. The authors postulate that this may be because motion parallax only has an additive effect if there is a significant amount of occlusion in the 3D graph rendering. They also note that previous work [BPG00, AGB96] has shown that the benefit of motion parallax and stereopsis depends greatly on context and can be influenced by the task, as well as by the experimental procedure.

2.6.2 Stereo rendering

Given a common dual image based approach for stereoscopic display a visualisation application must take the rendering hardware and display hardware into consideration.

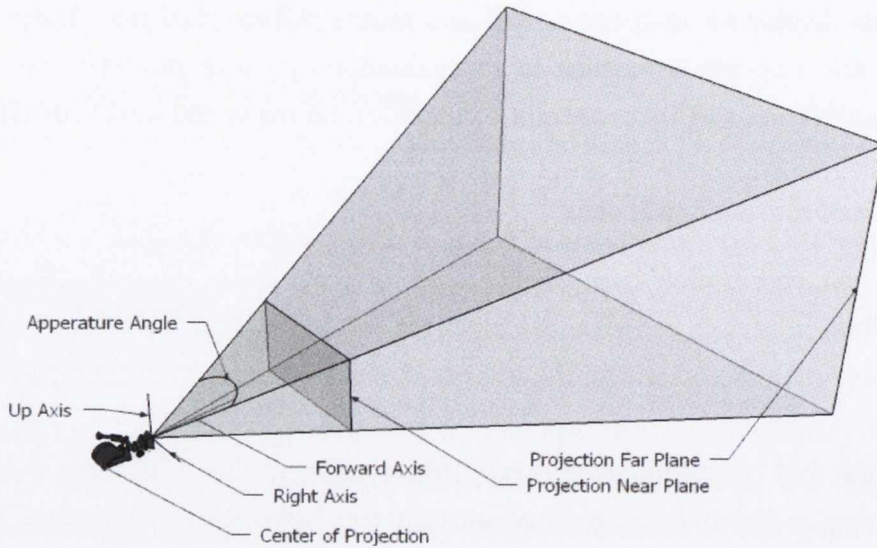


Figure 2.21: A simple single camera set up, the image projected onto the near projection plane is that which is seen by the user.

The OpenGL graphics API provides extensions that allow applications to render to separate display buffers for the left eye and the right eye. In order for an application to render an image to be viewed on a stereoscopic display the scene is rendered from the perspective of the left eye and rendered to the left display buffer and then from the perspective of the right eye to the right display buffer.

Each display buffer requires a different scene projection to reflect the fact the image is targeted to a specific eye. The standard approach to rendering a scene in a visualisation application is to model a camera with a position and set of orientation vectors in the rendering space. A viewing frustum is also defined. This is used to specify the projection matrix which maps from the three dimensional model of a scene (which may just be a simple graph visualisation) to the two dimensional projection plane as seen in figure 2.21. The parameters of this frustum can be specified as the position and size of near and far clipping planes. Alternatively, these parameters can be defined like those of a camera are, representing a field of view (also know as aperture) and an aspect ratio, with a maximum and minimum viewing distance. The aspect ratio is the proportion of the height of a projection plane to the width of the projection plane. In a simple single camera setup the projection plane can often be thought of as the near clipping plane of the viewing frustum. As there is only a single camera position, the view transformation being applied to the scene needs to only consider the camera position and orientation

To render a scene using a stereo camera it is necessary to know the position of each eye (which can be determined from a value for the eye separation) as well as build a separate viewing frustum for each. Because these viewing frustums have separate origins, but the same near and far planes, they are not the symmetric frustums as seen in the single camera

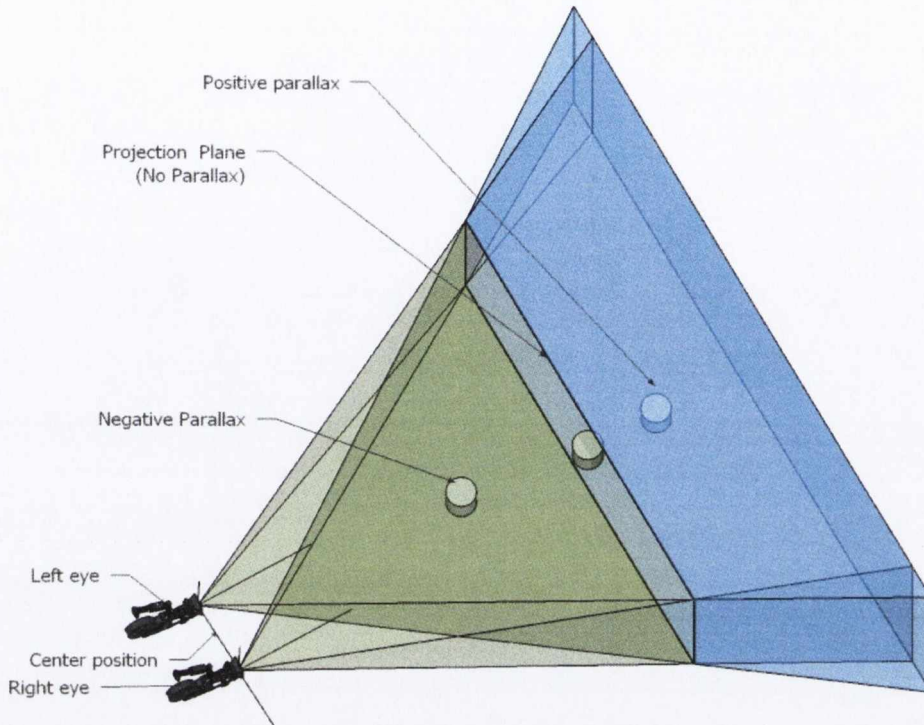


Figure 2.22: A stereo camera set up. The eye positions are offset along the right vectors of the camera by the eye separation value. The overlapping frustums are not symmetrical like the frustum used for the single camera eye setup

case (as can be seen in figure 2.22), we also need to define a focal length parameter, which is the distance from the camera position to the projection plane. This directly impacts the 3d effect of stereoscopic viewing. Objects which are closer to the camera than the projection plane will appear to pop out in front of the display (negative parallax). Objects which are further away will recede into it (positive parallax). The matrix which models the viewing transformation applied to the rendered scene by the camera also needs to be updated between left eye and right eye rendering passes. The camera orientation is the same for each eye, however the position of the camera for the left eye rendering and the right eye rendering depends on the eye separation.

Eye Strain

The stereoscopic parameters described above can have a significant effect on a users ability to perceive the graph as a 3D object. If there is a mismatch in values, such as the eye separation being too large relative to the focal length, there may be excess parallax (positive or negative) as an object becomes more distant from the focal length. As a result, the disparity between the left eye and right eye images is too large and the viewers brain is unable to combine them into a 3D object, so the stereoscopic effect is lost.

Even if the viewing setup still allows stereoscopic vision, in many cases the setup can

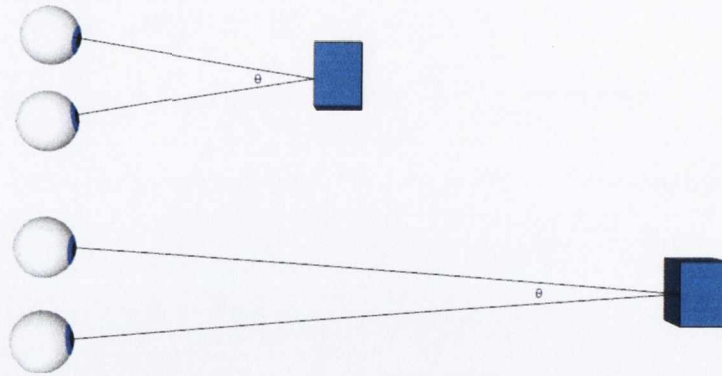


Figure 2.23: The vergence angle of the eyes θ changes depending on the distance to the object being focused on.

tax the human visual system so heavily that physical discomfort can occur. One of the main causes of this strain is related to a mismatch between the focal length of the eyes and what is referred to as the vergence. Vergence describes the convergence of the eye orientation when we look at an object. A more distant object will result in a narrower vergence angle as can be seen in image 2.23. When a user looks at an image on a stereoscopic display all objects are at the same focal length, regardless of how deep they appear to be. The vergence angle of an object combined with the disparity between the left eye and right eye images does provide enough information for a user to perceive the objects in 3D. However the the lack of correct focus information combined with the vergence may be the cause of eye-strain that is frequently associated with stereoscopic viewing [WRMW95].

2.6.3 Three dimensional layout of graphs

Many graph layout algorithms can be easily extended from the usual two dimensional layout to three dimensions. For example Ware and Mitchell used an 3D dimensional spring layout for their experiments[WM08]. Extending a force directed layout into three dimensions is a simple case of extending the force vectors and positions of the nodes into three dimensions. However three dimensional layout of graphs causes significant issues of occlusion, depending on the viewing angle of the graph.

2.7 Implementation of Graph Rendering and Processing

As described in section 2.1 graphs are most often visualised as node link diagrams. When rendering a node link diagram, there are many low level technical implementation aspects to be considered in order to execute the techniques described in this thesis to render the associated graphs to a display. We have developed a graph visualisation application to demonstrate and test our approaches. All images in this thesis, unless credited otherwise

are generated from this application.

Our implementation uses the C++ programming language. This was chosen due to the high performance offered, in terms of speed, as well as the wide choice of available libraries to support graph analysis and rendering. To aid in our processing of graph data we have utilised the BOOST graph library, which offer graph data structures and algorithms. BOOST is a cross-platform open source set of libraries that provides a wide range of functionality for C++ application development [Lib12]. For rendering our graphs we use OpenGL, a cross platform 3D graphics library.

2.7.1 Graphics Hardware

Hardware accelerated graphics are a common feature of modern commodity PCs. Modern graphics cards use a Graphics Processing Unit (GPU) with many cores capable of processing data in parallel, ideal for accelerating raster based rendering. Modern graphics hardware includes specialised memory, separate from main computer memory, that stores graphical data, allowing it to be processed by GPUs and displayed quickly. Within OpenGL a 3D object or model is specified as a set of vertices (representing points in 3D). These are then processed by the Graphics GPU to produce fragments of a (pixel based) raster image. These fragments are then processed further, combined into pixels and placed in an area of memory called a display buffer. The display buffer data is sent to a display device such as a monitor or projector for viewing. OpenGL utilises Vertex Buffer Objects (VBOs), which are structures in graphics memory which store the vertices of a 3D object to be drawn. Storing graph data in VBOs allows large interactive graphs to be rendered more quickly than if the data is passed from main application memory.

2.7.2 GPU Processing

Modern graphics hardware, though originally designed for enhancing 3D rendering performance, is also capable of being used for other computational tasks. It is extremely beneficial when calculations are discrete and capable of being done in parallel, such as the calculations of forces acting on an individual node for a force directed layout as seen in Frishman and Tal [FT07]. Programmable GPUs allow the user to create small pieces of code called shader programs which execute on graphics primitives such as the vertices of a 3D model and the pixel fragments of a raster display (and more recently geometric primitives).

In earlier attempts to utilise GPUs to aid in graph layout the GPU was utilised using an approach known as General Purpose GPU programming. The performance optimisations offered by the GPU are focused on graphics operations and as a result the programming model is structured for that specific field. Effectively GPGPU is an approach by which an algorithm is disguised as a graphics rendering pass, during which the programmable

shaders process the input graph data. Rather than outputting a pixel colour to the screen buffer, the output information is read back to the CPU and interpreted by the calling program. This programming model limits the range of problems that can be ported to the GPU.

Since Frishman and Tal's work [FT07], GPGPU has been replaced as an approach to programming graphics hardware. CUDA (Compute Unified Device Architecture) is an programming model realised by nVidia to specifically allow access to programmers to graphics hardware for non graphics purposes. It is a C based programming language that, while reflecting the underlying multi-core architecture of the GPU, is free of many of the restrictions of GPGPU programming.

Harish and Narayanan [HN07] showed the benefit of CUDA for accelerating graph algorithms. The authors showed significant benefits using basic CUDA implementations, for random general graphs Breadth First Search was 20 to 50 times faster than a corresponding implementation on the GPU. A Single Source Shortest Paths algorithm ran 70 times faster on the GPU than on the CPU. However such a dramatic performance was not seen for graphs which were scale-free. This meant that some vertices had considerably more edges than others, which impacted the performance. However the GPU approach still outperformed the CPU approach. For an example real world graph of low degree, the CPU actually outperformed the GPU. This was as on a low degree graph these algorithms are not easily parallelisable as the graph is almost linear, so the benefit offered by GPU parallelisation is lost. Luo et al [LWH10] have produced an effective Breadth First Search algorithm. In terms of layout Godial *et al.* [GHGH09] have demonstrated a CUDA implementation of Hachul and Jünger's Fast multi-pole multi-level Method (FM3)[HJ05] which performs at least 20 times faster than the CPU based version and is 30% faster than Frishman and Tal's GPGPU approach.

CUDA has also been used to optimise other aspects of graph visualisation than clustering and layout. For example in Ersoy *et al.*'s Skeleton-Based edge bundling [EHP⁺11], the algorithm for calculating a skeleton structure for graph edges ran 100 times faster using a GPU based CUDA solution, when compared to a CPU implementation.

Chapter 3

Agglomerative Clustering around Nodes of Interest

AS THE AMOUNT OF INFORMATION TO BE VISUALISED by a graph becomes larger or more dense, the graph becomes more difficult for a user to comprehend. Use of a clustering structure on top of the classical node-link model can help provide information, as nodes that are clustered together have an implicit relationship. A graph may not have an intrinsic data structure and any clustering provided by a generic clustering algorithm may not align with a user's task.

In this chapter we present an approach for agglomeratively clustering graphs based on user input. As part of our approach a user can specify nodes of interest, which form the basis of the clusters. We build clusters around these nodes using a heuristic which we have chosen based on the structure of the often encountered Small World Graphs described in section 2.1.2.

We chose clustering coefficient, described in section 2.1.2, as a heuristic. We build clusters agglomerative by adding nodes to clusters based on their impact on the resulting average clustering coefficient of the cluster. If a cluster has a high average cluster clustering coefficient, it indicates that all of the nodes within the cluster have many interconnected neighbours within that cluster.

Chapter structure: This chapter is structured as follows:

- In sections 3.1 and 3.2 we describe our motivation and the related work for this chapter and in section 3.3 we describe how we calculate clustering coefficients.
- Section 3.4 describes our initial investigations into using clustering coefficient as a heuristic to guide the agglomerative clustering around nodes of interest.
- Section 3.5 describes an approach which maximises the average cluster clustering coefficient of clusters.

- In section 3.6 we provide a detailed evaluation of our approach across a wide range of graphs, comparing it to other heuristics. We also compare our approach to a well known top-down clustering algorithm.
- In section 3.7 we apply our approach to a benchmark data set and examine the results against the existing classification of the nodes.
- In section 3.8 we describe our conclusions and potential avenues for future work.

3.1 Motivation for Clustering

Our motivation is to make graphs more comprehensible and we use graph clustering to support this aim. We are focusing on small world graphs specifically. This is due to the presence of groups of highly connected nodes, the strong likelihood of cluster structures within the graph, as well as the common occurrence of small world properties in real world networks. If a user is investigating nodes of specific interest to them, reorganising the layout of the graph based on the nodes of interest may aid in their analysis. For example a user may want to view a graph describing a large program focusing on specific classes, or a biologist may wish to view a predator-prey network focusing on certain animals.

The purpose of our clustering approach is to aid in the layout by clustering nodes around the user's nodes of interest. The clustering assigns nodes in such a way that they are clustered around nodes that they are more conceptually related to, based on graph structure. If grouping a node with one node set over another results in a higher heuristic score for that cluster, we can infer that the node conceptually belongs more to it. In less dense graphs a clustering may be obvious as there will be few links between clusters. However, for more dense graphs, useful clusterings may not be so obvious. The density of edges can make the graph more difficult to read and the relationships between nodes may be obscured. A node may also have strong relationships with several other nodes, and allowing the user to rearrange a graph based on nodes of interest allows the user to see clearly the relationships that are most pertinent.

3.2 Related work

The characteristics, origins and relevance of Small World Graph are described in detail in section 2.1.2. The background and state of the art of clustering is described in section 2.2.1 and evaluation techniques for clusterings are described in section 2.2.3. A brief summary of cluster and evaluation is provided here to provide context for this chapter.

3.2.1 Clustering

There are many different approaches to graph clustering (or partitioning as it is often referred to). Some methods use an algebraic approach, working on a mathematical representation of the graph, [FT07, vDoo]. Other methods such as Edge Betweenness Centrality Clustering [NG04] use a graph theoretic based approach, calculating graph theory characteristics of vertices or edges that are then used to partition the graph into clusters. Some clustering algorithms, such as edge betweenness centrality clustering take a top down, or *divisive* approach splitting the graph into separate clusters. Others take a bottom-up or agglomerative approach, merging sets of nodes together to form clusters [New04, HKKS03, QE01].

3.2.2 Clustering Evaluation Metrics

Newman and Girvan [NG04] define a measure of the quality of a division of a network graph, referred to as modularity. The measure is used to evaluate their community detection algorithm (which is essentially a top-down clustering algorithm). The measure has also been used in work by Newman [New04] as a heuristic value for agglomeratively building clusters. This metric is based upon the number of edges that start and end in the same cluster (referred to as communities in Newman and Girvan's paper). Auber et al [ACJM03] and Chiricota et al. [CJM03] use a quality measure developed by [MMR⁺98] and utilised in their clustering tool "Bunch". This measure, denoted *MQ* (*Modularisation Quantity*) computes a value for any given partition of a graph. Chiricota et al. and Auber et al. use a slightly modified version of MQ that is defined only for undirected graphs as an evaluation measure. The MQ value is used by the Bunch tool as a function to be optimised to provide a good clustering, rather than as a metric to evaluate one. Both modularity and MQ score are described in detail in section 2.2.3. Boutin and Hascoët [BH04] discuss many other clustering evaluation approaches (referred to by them as clustering validation indices) and they note that these evaluations are often difficult to interpret and compare. Bittencourt and Guerrero [BG09] and Wu *et al.* [WHH05] evaluate several clustering metrics in the context of software system analysis.

In their work on the layout of small world graphs [vHW08a], van Ham and Wattenberg utilised a social network based on the the influences between prominent historical figures, the "genealogy of influence" [Lov10]. In this network, individuals are connected if one of them was an influence on the work of another. For example Socrates influenced Plato, therefore there is an edge between the nodes representing each of these individuals in the network. The purpose behind the use of such a network is that the profession of each individual provides an extrinsic clustering of nodes, and such a clustering can be used to evaluate a layout (or in our case a clustering).

3.2.3 Edge Density

In section 2.1.4 we discussed graph density and differentiated between graph theoretic edge density and linear density. As mentioned previously, many real world graphs have a linear density value of $d_l \leq 10$. However some examples such as web-crawl graphs have even higher densities, such as web-crawl graphs with $d_l = 25.57$ [Melo6]. Increasing edge density alters the structure of a graph, and impacts the behaviour of an agglomerative clustering algorithm. The more dense a graph is, the larger the number of neighbour nodes that are available for agglomeration into a cluster. Clearly graph density needs to be considered as part of an evaluation of an agglomerative clustering algorithm. It is clear that graph theoretic density scales the number of edges more dramatically for a change in vertex count, so for comparison of densities between graphs with different node counts linear density provides a clear comparison.

Purchase [Pur97] has demonstrated how the crossing of edges is the graph aesthetic which affects most human understanding of the graph. Unfortunately, in large dense graphs, edge crossings are unavoidable. We hope that by clustering the graph intelligently, strongly related nodes will appear closer to each other within the same cluster. This will reduce long edges and the likelihood of edge crossings.

3.3 Calculating Average Local Clustering Coefficient

Calculating the clustering coefficient of a single node is straight forward and is described in section 2.1.2. A simple algorithm for calculating the clustering coefficient of an individual node within a cluster is shown in algorithm 2. The average clustering coefficient of a graph, sometimes referred to as the global clustering coefficient, is the sum of all node clustering coefficients divided by the number of nodes. The average clustering coefficient of a cluster, reflects the level of inter-connectivity of nodes within the cluster. Therefore when calculating the clustering coefficient of nodes within a cluster, to generate the average cluster clustering coefficient, only neighbours within the same cluster are considered.

In our approach, we set the clustering coefficient of a node to zero if it has less than two neighbours. Sometimes, as done by Schank and Wagner, clustering coefficient is only considered for nodes with more than two neighbours. This results in the global clustering coefficient being the sum of all node clustering coefficients, divided by the number of nodes which have two or more neighbours.

The calculation of the clustering coefficient for a large set of nodes can be a time consuming task. The time taken to calculate the clustering coefficient of a node does not just depend on the size of its neighbourhood, but also on the size of the neighbourhood of each node in the original node's neighbourhood. Therefore calculating the clustering coefficient for each node in a graph depends not only on the number of vertices $|V|$ and the number

of edges $|E|$ but also the distribution of edges in the graph, (see [New10]). Fortunately the clustering coefficient for each node in a graph can be calculated in parallel.

We have implemented the calculation of the clustering coefficient of each node within a graph or cluster on the GPU. The result is averaged by the CPU to determine the average clustering coefficient for the nodes within a cluster. The input data required for this algorithm are the nodes within the cluster, and the edge list for the graph. For extremely large graph an approximation may be preferable, Schank and Wagner[SW05] describe an approach for approximating clustering coefficient quickly. However for building clusters agglomeratively using clustering coefficient we will calculate exact values.

Algorithm 2 Algorithm for calculating clustering coefficient of a node within a cluster

```

 $v \in V_c$ 
for all  $u \in V_c$  do
  if  $\{u, v\} \in E_c$  then
     $v.neighbourhoodSize := v.neighbourhoodSize + 1$ 
    for all  $w \in V_c$  do
      if  $u \neq w \wedge \{u, w\} \in E_c \wedge \{w, v\} \in E_c \vee \{v, w\} \in E_c$  then
         $v.neighbourhoodEdgeCount := v.neighbourhoodEdgeCount + 1$ 
        {As  $w$  is also a neighbour of  $v$ }
      end if
    end for
  end if
end for
if  $v.neighbourhoodSize > 1$  then
   $v.clusteringCoefficient := v.neighbourhoodEdgeCount / v.neighbourhoodSize * (v.neighbourhoodSize - 1)$ 
else
   $v.clusteringCoefficient = 0$ 
end if

```

3.4 Initial Investigation of Clustering Coefficient

3.4.1 Introduction

In this section we describe our initial investigation into the use of average cluster clustering coefficient as a heuristic. In order to determine its potential we evaluated it against both procedurally generated and real-world graphs. Our approach consists of agglomeratively merging nodes, one at a time, into the cluster which results in the highest heuristic score for the node being clustered.

3.4.2 Initial Clustering Algorithm

To determine which cluster a node conceptually belongs to, the average clustering coefficient of a cluster is used as a heuristic. The clustering process is as follows:

1. Specify the nodes of interest used as a basis for clusters.
2. Add one neighbour node to each node of interest to form a basic cluster.
3. Build a list of remaining nodes in the graph, sorted by distance from a node of interest and node neighbourhood size.
4. Add each node to the cluster, that has the highest resulting average cluster clustering coefficient if the new node is included in the calculation
5. Assign the single neighbour nodes to the clusters of their neighbours.

Each cluster initially only contains a single node of interest, selected by the user. For each node of interest a single neighbour is added as the calculation of a clustering coefficient requires a node to have more than one neighbour. The node added to the cluster is the neighbour of the node of interest, with the largest neighbourhood size. This results in each cluster containing two connected nodes.

A node's neighbourhood is the set of nodes which it is directly connected to. The graph distance between a node and its neighbours is exactly one. The set of all nodes of the graph that have a neighbourhood size large than one and have not already been assigned to clusters is then stored in an ordered node list, built by traversing the graph from each of the nodes of interest using a breadth first search. Nodes are stored, primarily, in order of their increasing graph distances from a node of interest and secondarily by the size of the node's neighbourhood from largest to smallest. Nodes that are connected to only one other node (i.e. it has a neighbourhood size of one), are not added to the list. The reasoning behind this is that a node which only has one neighbour is guaranteed to have a negative impact on the local clustering coefficient value of a cluster. Given that the node can only ever be added to the cluster that it is connected to, it is added to the cluster that its only neighbour is assigned to once all other nodes are assigned.

The motivation for ordering the list secondarily by neighbourhood size is to allow nodes of a lower neighbourhood size to be added to a node where as many as possible of their neighbours have already been added. If nodes of a large neighbourhood size already are processed first then any node added is more likely to find several of its neighbour nodes already assigned to the cluster. Furthermore, this ordering results in more balanced cluster sizes, as it will prevent the clusters which are initially based on more highly connected nodes from taking all the nodes with a small neighbourhood size. A more balanced

clustering is more likely to result in a more symmetrical graph, which per Purchase's experiments [Pur97] is a graph aesthetic which affects human understanding (although nowhere near as strongly as the number of edge crossings).

In the next stage of the algorithm, the ordered list is iterated through adding each node temporarily to a cluster. When a node is added to a cluster the average clustering coefficient of the cluster is recalculated to determine the impact of adding the node to the cluster. The node is then permanently added to the cluster which has the highest resulting coefficient. Finally, once all other nodes have been assigned to a cluster, the single neighbour nodes are then assigned to the cluster of their neighbour.

3.4.3 Evaluation Approach

In order to evaluate the effectiveness of the clustering we compare our algorithm to variations where cluster coefficient impact was not taken into account. For the "Round robin" clustering approach, nodes are initially sorted in the same manner as before, but assigned to each cluster in a sequential fashion. Nodes are only assigned to clusters which they are connected to. A more thoroughly random approach was also taken, by assigning nodes to a cluster chosen entirely at random, from a list of all clusters that neighbours of the node have already been assigned to. Furthermore we have also evaluated using the change of the clustering coefficient (which we refer to as the clustering coefficient delta). In this approach nodes are assigned to the least negatively impacted cluster, instead of assigning the node to the cluster with the highest resulting cluster clustering coefficient.

Evaluation Graphs

We evaluated our algorithm using a wide variety of graphs. For an evaluation using real world data, we generated a set of four graphs, based on connectivity between Wikipedia articles. We evaluate these graphs using Newman and Girvan's modularity metric.

We also randomly generated small world graphs which are clustered using our approach and as well as the round robin and random approaches. We used Watts and Strogatz' approach for creating small world graphs [WS98] as described in section 2.4.1. These randomly generated graphs contained 60 nodes and vary in edges density. We also generated a second set of 20 graphs of consistent size and density, with an increasing probability of rewiring. This results in a set of graphs with a decreasing clustering coefficient. These procedurally generated graphs were evaluated based on the modularity of the resulting clustering.

In order to analyse how effective the use of the average local clustering coefficient of a cluster is in building conceptually related clusters, we created an artificial social network data set modelling activities at a sports club. Our model contained 100 nodes each representing a member of the club. Each member is assigned a level of interest in six activities,

between 0 and 1.0. The sum of a member's interest across all activities is equal to 1. In order to generate the graph we calculated the Euclidean distance between each member's 6 levels of interest. If the Euclidean distance was less than a threshold value of 0.5 we assume, due to the common amount of activities, that the members are socially connected. Therefore we added an edge to the graph connecting the nodes representing the members. The resulting graph contains 803 edges. Using this graph to evaluate our clustering we can see exactly the ratings for each node for each activity and hence determine if they have been placed in a conceptually correct cluster. The node of interest selected for each cluster is a person who undertakes only one activity with the maximum level of interest. This means that each cluster member should have some level of interest in the activity of the node of interest.

We also use our algorithm to cluster and layout a real-world social network data set. We chose the influence data set [Lov10] as used by van Ham and Wattenberg[vHW08a]. This data set contains prominent figures in the field of art, science and entertainment and relates them using "influenced by" relationship. The generated graph contains 1929 nodes and 4364 edges.

Evaluation Metric

We used Newman and Girvan's modularity metric [NG04], which is described in section 2.2.3. Modularity depends solely on the relationships between nodes. Where contextual meta-data about the node clustering is available, we use this to determine if the node conceptually fits in with the cluster it is assigned to.

3.4.4 Results

Wikipedia Data Set

The clustering coefficient based, round robin and random algorithms were each run on the Wikipedia test graphs, where the four nodes with the highest degree were selected for clustering. The random clustering was run three times for each graph and averaged, as it resulted in a different clustering each time. The resulting modularity of each graph is displayed in Table 3.1. The use of clustering coefficient as a metric produces a significantly higher level of modularity than a round robin or random assignment of nodes to clusters.

Randomly Generated Small World Graphs

For the randomly generated small world graphs, we compare the change in modularity over graphs of increasing density of edges relative to nodes with the four different approaches. We use the clustering coefficient approach, round robin assignment and a random assignment as well as the delta of the clustering coefficient in determining the assignment of

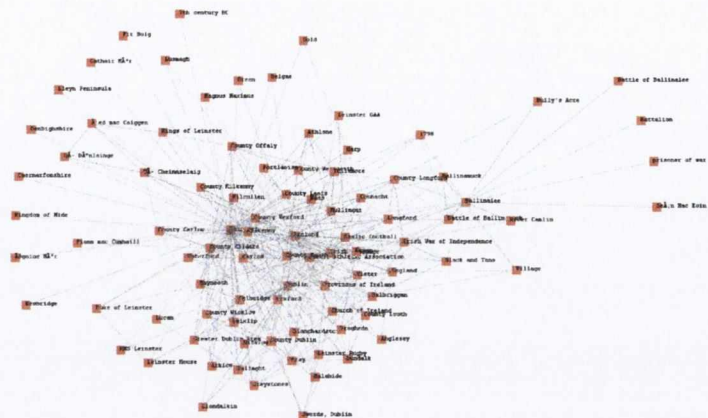


Figure 3.1: Wikipedia graph with 91 vertices and 567 edges laid out using a simple force directed algorithm.

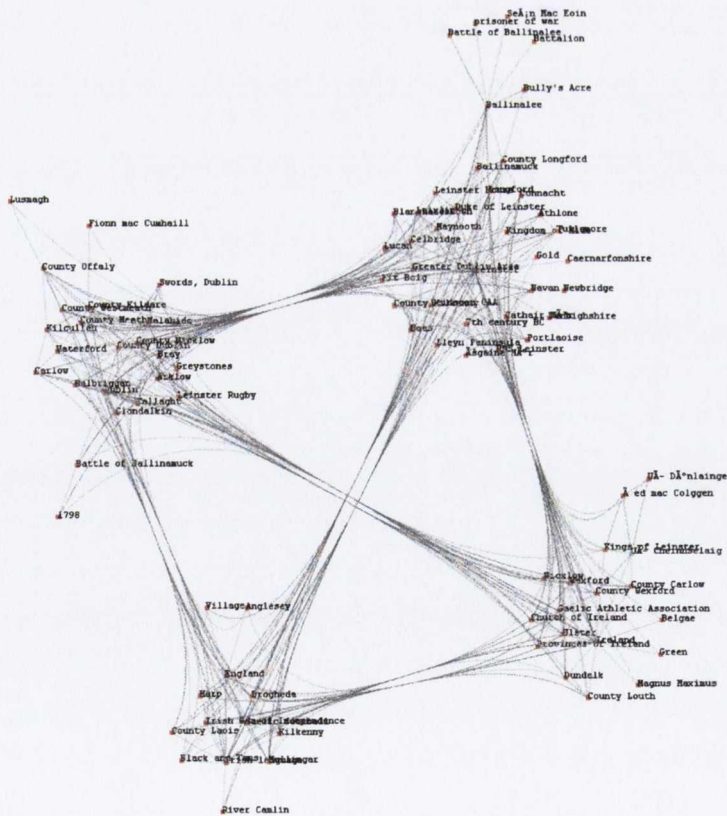


Figure 3.2: Graph from Figure 3.1 using our approach.

$ V $	$ E $	Clustering Coefficient	Round Robin	Random Average
91	567	0.1279	0.049	0.561
358	3729	0.0931	0.038	0.0424
506	3962	0.1545	0.0692	0.0645
1000	28534	0.0251	0.0038	0.005

Table 3.1: Modularity values for Wikipedia based graph using different approaches to clustering.

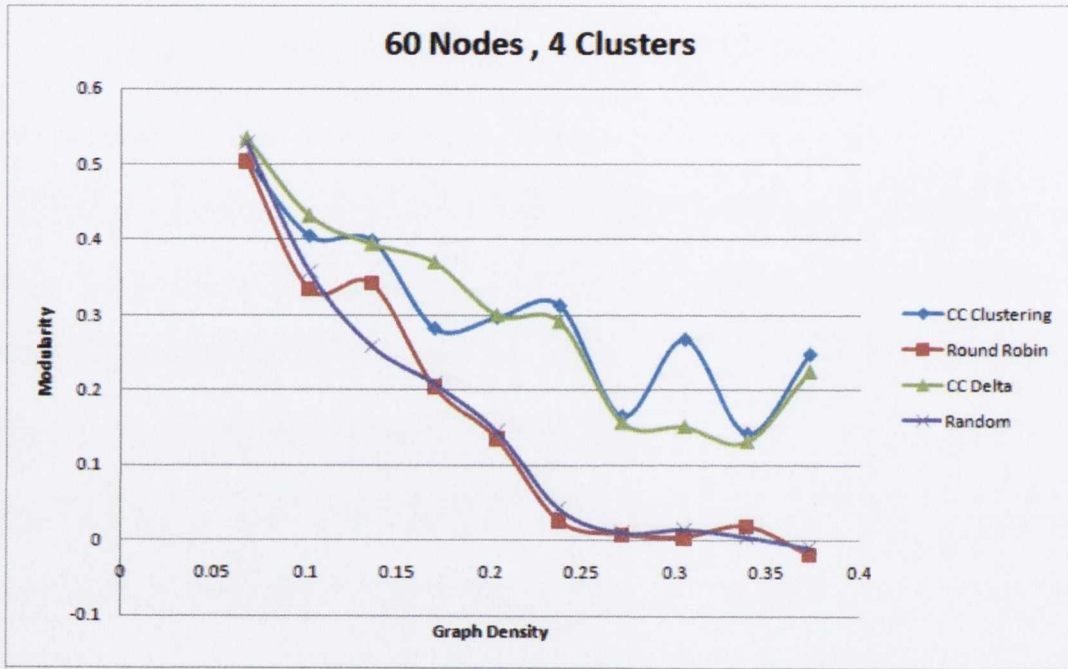


Figure 3.3: The modularity of graphs containing 60 nodes and increasing in density, using the described clustering approaches for building 4 clusters.

nodes to clusters. In each case the use of clustering coefficient as a heuristic showed an improvement. Figure 3.3 shows the improvement in using clustering coefficient over the round robin and random method for creating 4 cluster for a graph with 60 nodes and edge density increasing from 120 to 660 edges. On average use of the maximum average clustering as opposed to the delta of the average clustering coefficient resulted in a more modular clustering but the difference is not as distinct as with the other approaches. Similar results can be seen for forming 5 or 6 clusters (see Figures 3.4 and 3.5).

We also analysed the effect of altering clustering coefficient for a graph of a given size and density. Global clustering coefficient is altered implicitly by changing the rewiring probability when generating the graph. As can be seen from figure 3.6 the impact of changing the global clustering coefficient is not consistent, but use of the clustering coefficient as a heuristic still improves modularity even as the graph becomes less small world like.

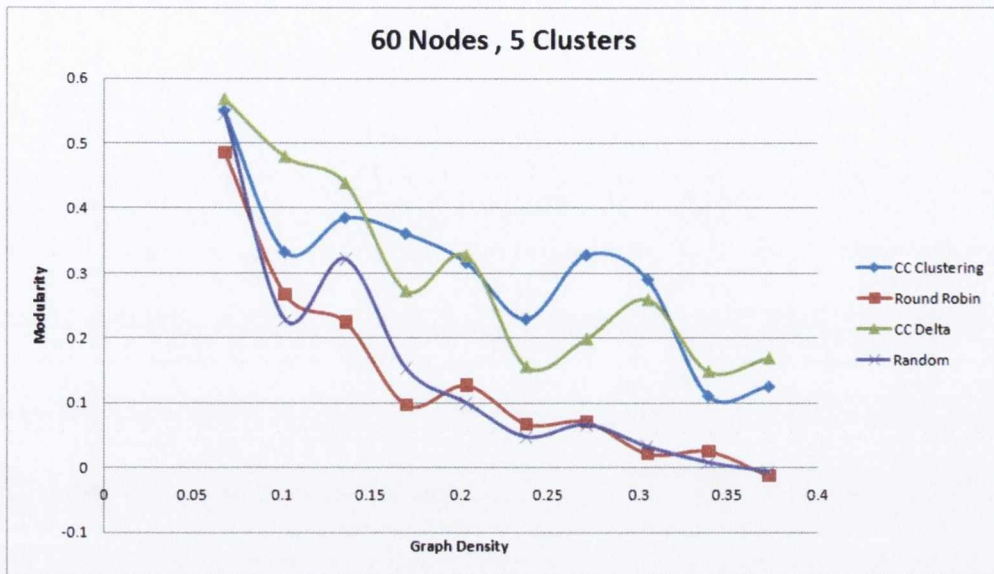


Figure 3.4: Modularity for building 5 clusters.

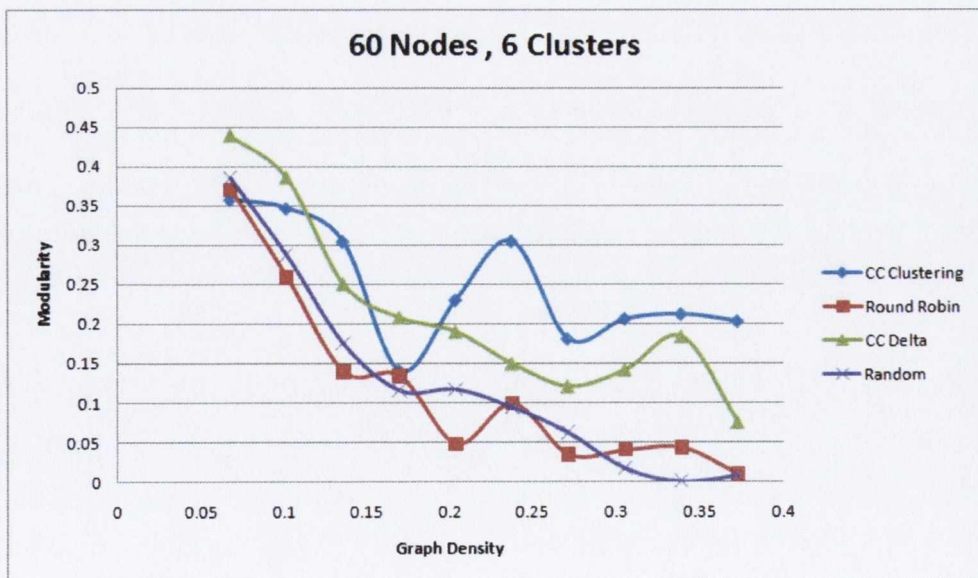


Figure 3.5: Modularity for building 6 clusters.

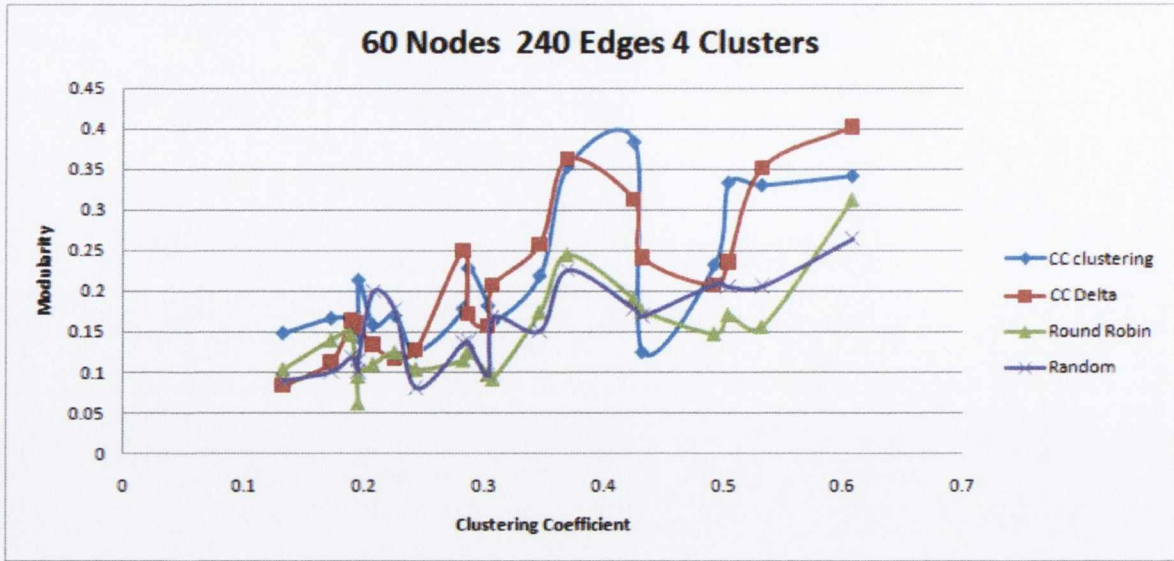


Figure 3.6: The modularity of graphs containing 60 nodes and 240 edges increasing in clustering coefficient, when clustered using 4 nodes of interest.

	Clustering Coefficient	Round Robin
Soccer	0.9	0.9
Swimming	0.6038	0.5877
Tennis	0.4208	0.5563
Gym	0.565	0.441304
Running	0.9	0.4088
Cycling	0.925	0.010265

Table 3.2: Level of interest in the sport of the node of interest by cluster.

The fluctuation in modularity is a result of the randomness in generating the graphs and the fact that the most well connected nodes are automatically selected as nodes of interest, which can change significantly for each random graph. It can be seen in Figure 3.7 that a similar result was achieved for 5 clusters.

Sports Club Data Set

We generated 6 clusters for the sports club data set and compared the use of the clustering coefficient approach to the round robin approach. For each cluster we calculated the sum of the levels of interest in each activity for each person in the group and normalised this value by dividing it by the number of people in the group. In Table 3.2 we display the scores for the specific activity associated with the cluster. It can be seen that for every activity apart from tennis, use of clustering coefficient results in a equal or better score for the cluster. The lower value for tennis compared to using the round robin approach results from a larger number of members being interested in tennis, but not as their primary activity.

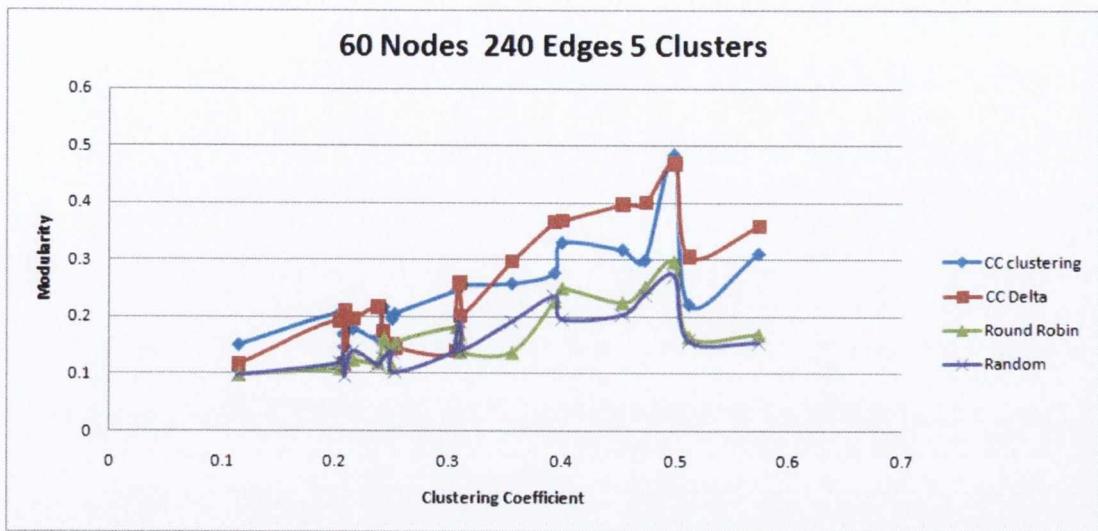


Figure 3.7: The modularity of graphs increasing in clustering coefficient, when clustered using 5 nodes of interest.

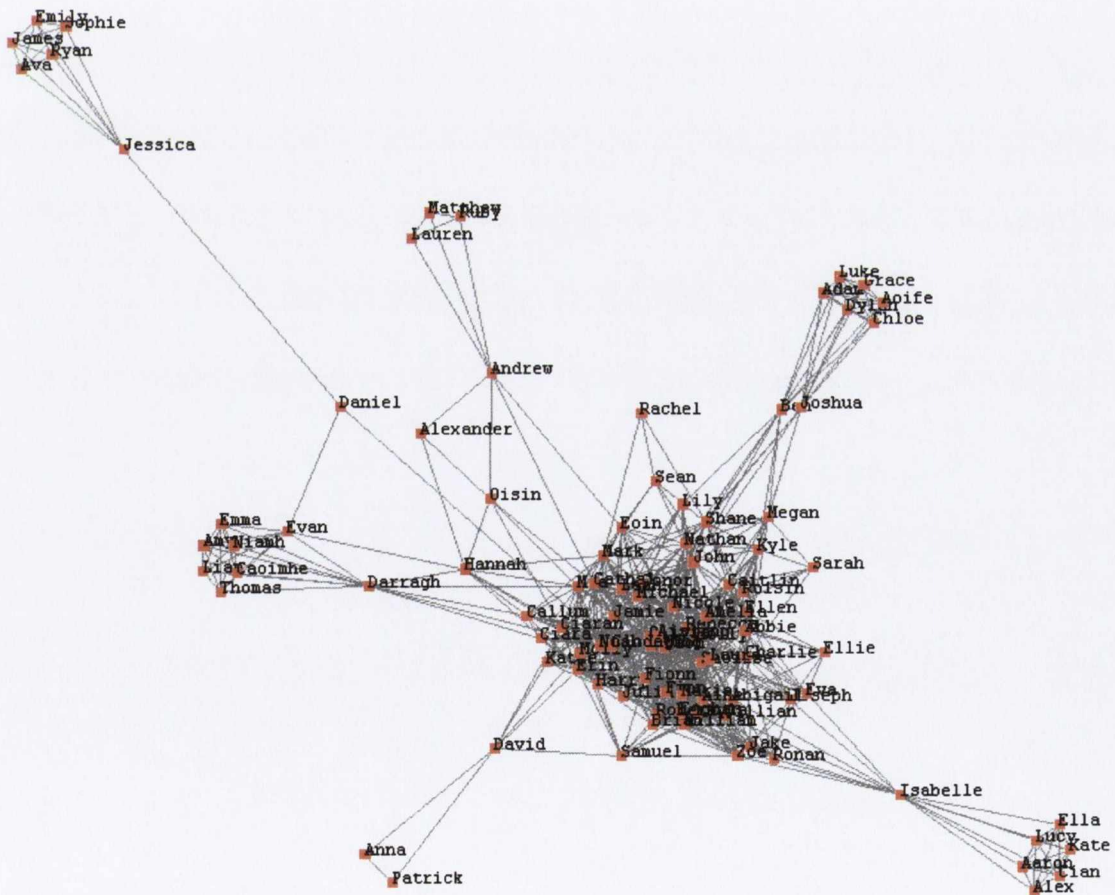


Figure 3.8: Sports club graph with 100 vertices and 803 edges laid out using a simple force directed algorithm.

Examining the number of members assigned to clusters that they have no interest in reveals a clear result. Using the clustering coefficient approach, this only happened for four members. Using the round robin approach, 17 members are assigned to clusters where the node of interest represents a sport that they have no interest in.

Influence Data Set

	Clustering Coefficient		Round Robin		C.C. Delta	
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
Actor	286	37	282	41	286	37
Artist	54	154	104	104	59	149
Mathematician	23	147	60	110	47	123
Musician	44	115	68	91	27	132
Philosopher	520	66	183	403	393	191
Scientist	133	350	151	331	40	443
Total	1060	869	848	1081	854	1075
Total %	54.9508	45.0493	43.9606	56.034	44.27164	55.72836

Table 3.3: Correct assignments of nodes by profession to clusters using each approach.

For our analysis of the influence data set each node, representing a person, has been labelled with a primary profession (as also done by van Ham and Wattenberg [vHW08a]). The classification by professions were as follows: Actor (including comedians, 321 nodes), artist (209 nodes), mathematician (170 nodes), musician (159 nodes), philosopher (586 nodes) and scientist (484 nodes). The previously described clustering and layout approach has been applied to this data set, with the nodes of interest being manually selected as one prominent individual from each set. Respectively these individuals were George Carlin, Vincent van Gogh, Carl Gustav Jakob Jacobi, Ludvig Van Beethoven, Friedrich Nietzsche, and Albert Einstein.

A completely accurate assignment of nodes to clusters based on profession is not expected as there are many other characteristics which affect relationships between nodes, such as indistinct boundaries between professions (particularly for mathematicians, philosophers and scientists). Also the era in which the person chosen as a node of interest lived impacts relationships and influence. The selected nodes of interest were chosen as people who were strong examples of each field, the nodes representing them were well connected, and their professions were more clearly defined. However, choice of node of interest does have a large impact on the data set. For example if a more contemporary musician such as Miles Davis is marked as a node of interest, many of the classical musicians end up being associated with the Philosophers cluster.

The breakdown of the correctness of resulting clusterings, categorised by profession, can be seen in Table 3.3. We performed a comparison using the round robin approach and clustering coefficient delta approaches of assigning nodes to clusters. It can be seen



Figure 3.10: Genealogy of Influence graph with 100 vertices and 803 edges laid out using a simple force directed algorithm.



Figure 3.11: Graph from figure 3.10 using our approach. The purpose of this diagram is to convey the scale of the data set and to show the impact of clustering on layout. The actual results of our approach are described in table 3.3 and figure 3.12

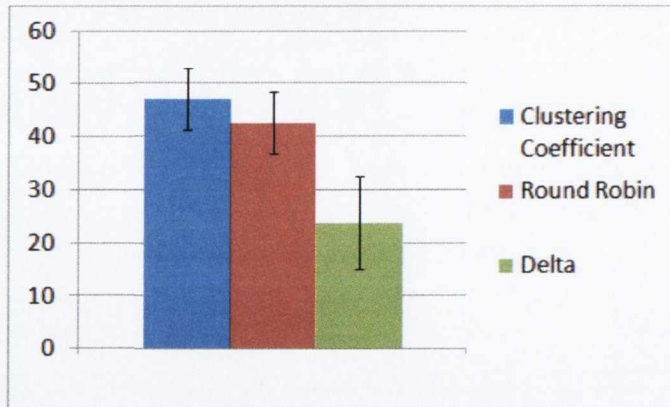


Figure 3.12: Chart indicating the number of correctly clustered nodes averaged over 20 different input sets of nodes of interest. The error bars indicate the standard deviation.

from the above tables that using clustering coefficient as a heuristic resulted in nodes being assigned to clusters that they were more conceptually related to in terms of profession. It is also clear that assigning nodes to the cluster with the largest resulting average clustering coefficient categorized the nodes more effectively than using the average local clustering coefficient delta or round robin approach.

To verify this we also ran our algorithm over 20 different sets of nodes of interest built using popular nodes for each profession. On average the clustering coefficient approach was correct 46.9% of the time, the round robin approach 42.5% of the time and the average cluster coefficient delta 23.9% of the time (see figure 3.12). The delta approach fared poorly as there were many cases where a disproportionately large number of nodes ended up in a small number of clusters. The clustering coefficient approach outperformed the round robin approach in 17 out of the 20 cases. In the worst of these 3 cases, the round robin approach scored 1.71% higher than the clustering coefficient approach. In the best case overall the clustering coefficient approach scored 11.15% higher than the round robin approach.

3.4.5 Conclusions of our Initial Investigation

It can be seen from the above examples that using clustering coefficient as a heuristic resulted in nodes being assigned to clusters that they were more conceptually related to. It is also clear that assigning nodes to the cluster with the largest resulting average clustering coefficient categorized the nodes more effectively than using the average local clustering coefficient delta. The previous evaluation has been useful in providing an initial evaluation of clustering coefficient as a heuristic in a broad range of applications, however a more thorough evaluation is required against a larger set of graphs with well defined characteristics, as well as a comparison to other heuristics. We provide such an evaluation in the following section.

3.5 Maximising Clustering Coefficient Approach

The preceding work shows us that there is some benefit in using clustering coefficient as a heuristic. However the preceding evaluation does not include comparison with other heuristics, and the approach is dependent on the sorting of nodes, to provide an ordering of addition of nodes to clusters. We have since taken an alternative approach which aims to maximise the clustering coefficients by adding the node to a cluster which results in a higher cluster clustering coefficient than the addition of any node to any other cluster. This also removes the dependency of using a breadth first search to determine node distances from a node of interest, and the requirement for a secondary sort of nodes by neighbourhood size. Maximising clustering coefficient regardless of cluster size prevents cluster node distribution becoming unfairly balanced. We also use a stronger metric than largest degree to determine the second node of a cluster. We also provide a more thorough evaluation against a much larger set of graphs.

3.5.1 Clustering Approach

Our newer approach consists of a clustering algorithm which agglomerates clusters, based on maximising the heuristic score of the resulting clustering incrementally. We grow the clusters around each of the nodes of interest by iteratively adding individual nodes to clusters so that the heuristic score for the graph will be increased as much as possible each iteration. In order to evaluate our approach thoroughly we use alternative cluster agglomeration heuristics for comparison. We describe our heuristics in the following section, and include any impact made to the initial cluster setup and node assignments for these heuristics in the description of our approach.

3.5.2 Chosen Heuristics

For our experiment heuristics we have chosen:

- Modularity, which we denote MOD.
- Modularisation Quantity, which we denote MQ.
- Average cluster clustering coefficient, which we denote AC₃.
- Random assignment to a connected cluster, which we denote RAND.

Modularity (MOD)

We have previously used Newman and Girvan's modularity as a metric and now we also use it as a heuristic. Modularity is described in detail in section 2.2.3.

Modularisation Quantity (MQ)

Modularisation Quantity (MQ) is a measure developed by Mancoridis *et al* [MMR⁺98] and utilised in the “Bunch” tool as a function to be optimised to provide a good clustering. Auber *et al* [ACJMo3] and Chiricota *et al.* [CJMo3] use it as a quality metric to evaluate clusterings. It is described in more detail in section 2.2.3.

Average Cluster Clustering Coefficient (AC₃)

When we calculate the average clustering coefficient of a cluster (AC₃ score), we only consider the nodes and their neighbours from within that cluster. The AC₃ score of a cluster describes how well inter-connected the nodes of a cluster are. This implies that the higher the clustering coefficient of a cluster the more strongly related to each other the nodes within the cluster are.

Random Assignment (RAND)

In order to provide a comparison clustering in which no heuristic is used, we have also implemented a random assignment of nodes to clusters. A node is chosen at random from the combined neighbourhood of the clusters and then randomly assigned to one of the clusters that it is connected to. The process is then repeated until all nodes are assigned. Nodes can only be assigned to clusters in which they have a neighbour, to allow a reasonable comparison with the preceding heuristics.

Heuristics as Evaluation Metrics

Each of these heuristics, apart from random assignment, can also act as a quality measure to evaluate clusterings of a graph. Therefore we use them not only to create our clusterings but also to evaluate the quality of our resulting clusterings. When utilising clustering coefficient as a metric we take the average clustering coefficient of each of the clusters generated so far. This is unlike MOD and MQ as they provide a score for the clustering of the graph across all clusters. Therefore a high average of the AC₃ score for each cluster does not imply that all clusters have a high average clustering coefficient. A large standard deviation between the AC₃ score of the clusters indicates that some clusters have been created with a low quality of clustering. Therefore we also measure and report the standard deviation of the AC₃ score for each cluster.

In the cases where a clustering using a heuristic other than clustering coefficient produces clusters containing only one or two nodes, it is not possible to calculate the AC₃ score of the cluster as a metric. Therefore, in such a case we assign the cluster a clustering coefficient of -1.0. This results in a suitably decreased score that reflects the poor quality of the clustering, when rating the graph using the average AC₃ score as a metric.

Analysis of the average AC₃ score and its standard deviation will also indicate if there is an unjustifiably extreme clustering. If a clustering has resulted in an extreme distribution of nodes, such that one cluster contains most of the nodes that cluster will have a very low clustering coefficient. If some small clusters are formed with very high clustering coefficients, the remaining clusters will have much lower values, which will be reflected in the standard deviation. These combination of these two metrics avoids the necessity for a metric such as Wu *et al's* Non Extreme Distribution [WHH05].

3.5.3 Initial Cluster Set Up

The initial set of nodes of interest that are to form the basis of the clustering is selected by the user. If our heuristic is either MOD or MQ, or we are using the random approach (RAND), we can begin to add further nodes to the clusters once the initial cluster nodes have been specified. This is because it is possible to calculate modularity and MQ heuristic value, or randomly choose a node, if a cluster contains only 1 node.

However this is not the case for the AC₃ heuristic, as we need to have at least two nodes existing already in the cluster before we can calculate a valid heuristic value for a new node being added. Therefore, before we start adding candidate nodes to the cluster using AC₃ as a heuristic, we need to add a second node to the cluster of each of the nodes of interest. The nodes that are candidates for addition are nodes within the neighbourhood of the node of interest.

We would like to add a node that is similar as possible to each of the nodes of interest, so we use the *Jaccard index* of the node of interest and the candidate second node's respective neighbourhoods.

The Jaccard index ρ of two sets of nodes A and B, is defined as:

$$\rho(A, B) = \frac{|A| \cap |B|}{|A| \cup |B|} \quad (3.1)$$

If the node of interest is denoted by v and the candidate node is denoted by u we can write the Jaccard index as

$$\rho(\Gamma(v), \Gamma(u)) = \frac{\Gamma(v) \cap \Gamma(u)}{\Gamma(v) \cup \Gamma(u)} \quad (3.2)$$

The node which is used as the secondary node of the cluster based around v is the node u for which $\rho(\Gamma(v), \Gamma(u))$ is the largest. Ideally we aim to select a node where the neighbourhood Jaccard index is 1. If the node chosen has already been assigned as a neighbour of one of the other nodes of interest, we assign the node to the node of interest which results in the best Jaccard neighbourhood index. The node of interest with the lower resulting Jaccard index is assigned its neighbour node with the next highest resulting Jaccard index. If this replacement node has also been assigned, we repeat the revaluation until all nodes

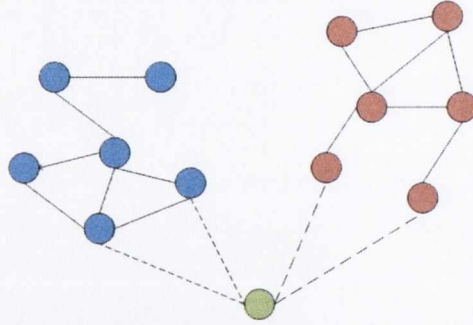


Figure 3.13: An example considering whether the green node should be clustered with either the red or blue clusters using the clustering coefficient heuristic. The green node is added to the blue cluster increasing the cluster's AC_3 score to 0.48. Adding it to the red cluster would reduce AC_3 score to 0.33. If the MQ or MOD heuristics were used new scores would be calculated for the graph as a whole for the addition of each node to the cluster. The clustering with the highest final score is made permanent.

of interest have been assigned distinct neighbours. Our previous approach simply used the largest degree neighbour of a node of interest. Using the Jaccard index ensures that there is an overlap in neighbourhoods, and as a result a stronger similarity between the initial nodes of the cluster.

3.5.4 Assignment of Nodes to Clusters

Once the initial clusters are created, we store the neighbourhood of each cluster and use this as input set of nodes which can be potentially added to a cluster. Given a clustering of p clusters $C = \{C_1, C_2, \dots, C_p\}$ where each element of C contains a disjoint subset of the graph $G = (V, E)$ such that $C_i = \{v_{i1}, v_{i2}, \dots, v_{in}\}$, $n = |C_i|$ and $C_i \subset V$, we define the neighbourhood of a cluster i as

$$\Gamma(C_i) = (\Gamma_G(v_1) \cup \Gamma_G(v_2) \dots \cup \Gamma_G(v_n)) \quad (3.3)$$

Each of the candidate nodes is added temporarily to a cluster and a score based on that addition is calculated. The node which maximises the heuristic score of the graph (or of each cluster) is permanently added to the cluster. Once a node is added the process is repeated until all nodes have been assigned to clusters. The MOD and MQ heuristics are scored across all clusters, so once a node is added, the scores will have to be recalculated in the next round of assignments. If the AC_3 heuristic is being used, only the score of the cluster which has had a node added will have changed. The AC_3 scores calculated for the other clusters and their candidate nodes will be unchanged from previous rounds. This allows caching of the results for later reuse, which decreases computation time.

3.6 Clustering evaluation

3.6.1 Evaluation Graphs

We use Watts and Strogatz's beta approach for creating small world graphs, described in section 2.4.1, for evaluating the effectiveness of the heuristics. This approach allows us to create a large set of graphs of various densities and various levels of structure, from regular lattices, to small worlds graphs, to completely random graphs. Each graph in our test set consists of 200 vertices. We have generated graphs varying the input probability to the beta model from 0.5 to 0.95. We have also varied the Edge density of the graph from a graph theoretic value of value of $d = 0.03$ to $d = 0.59$, resulting in the most dense graph having 11,800 edges. This is equivalent to a range of linear edge densities from $d_l = 3$ to $d_l = 59$. We have clustered each graph using our described heuristics. Four nodes with the largest neighbourhoods have been selected as the nodes of interest, resulting in four clusters. Due to the random nature of the graph generation we have averaged each result across 3 graphs generated with the same input parameters. Our full test set of data consists of 285 graphs for each of the three generation runs. We chart the results of the clustering, for each metric, for a sample of two lower density graphs ($d_l = 7$ and $d_l = 3$), in figures 3.15 and 3.16, a sample of high density graphs in figures 3.18 and 3.17, a sample of the more structured graphs in figure 3.19 and a sample of the more random graphs in figure 3.20. The standard deviation of the heuristics across the 3 graphs is displayed as the error bounds. The chart displaying metric scores for specific densities display a range of graphs progressing from nearly fully structured $p = 0.05$ to nearly full random $p = 0.95$. The charts displaying metric scores for specific levels of structure, with densities ranging from $d_l = 0.03$ to $d_l = 51$. The charts displayed in these figures encompass 191 distinct graphs. The maximum number of edges the a 200 vertex undirected graph can have is 19,900 which is approximately 100 times the number of vertices. This means that for our graph set the linear density is approximately 100 times the graph theoretic edge density.

Evaluation Graph Clustering Coefficient

Each of the graphs in our evaluation has 200 nodes. The constant node count impacts the small world characteristics of the graphs as they become more dense. It can be seen from figure 3.14 that, for lower density graphs $d_l = 3$ ($d = 0.03$) and $d_l = 7$ ($d = 0.07$), there is a very large difference between the clustering coefficients of the more structured graphs $p = 0.1$ and the nearly random graphs $p = 0.95$. However as the graphs become more dense the difference diminishes. This is not an effect of the linear density. Instead it is caused by the graph theoretic density. At $d = 0.5$, 50% of all possible connections in the graph will be made. This means that many of a nodes neighbours will be connected to each other, even if the graph is fully random (hence the clustering coefficient value of approximately 0.5, for

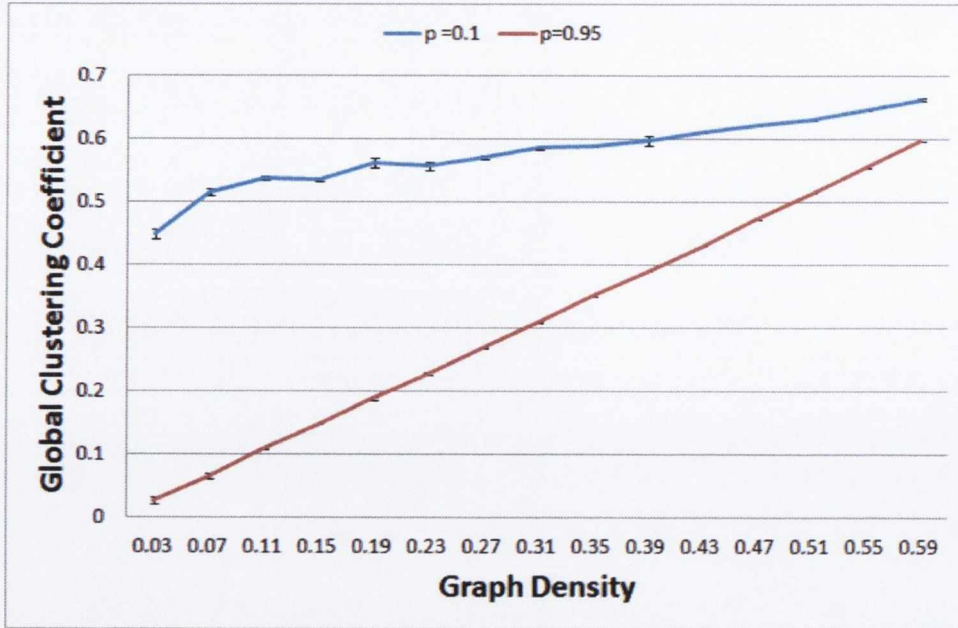


Figure 3.14: The average clustering coefficient of structured test graphs ($p = 0.1$) and almost fully unstructured test graphs ($p = 0.95$) across a range of densities. The error bars indicate one standard deviation.

graphs with an edge density of 0.5). We include the full range of graphs for our analysis to ensure a thorough evaluation, but it is worth bearing in mind that it is at the lower densities that the small world characteristics are strongest.

3.6.2 Results and Analysis

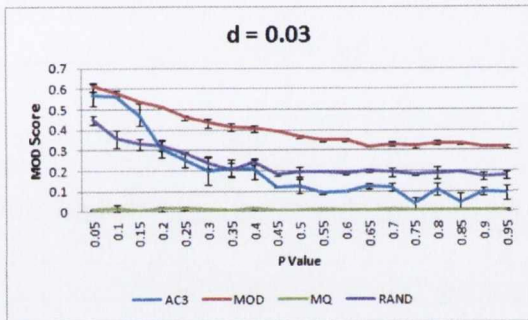
The effectiveness of each heuristic differs depending on the density of the graph, how random the graph is as a result of the input probability p of the generation algorithm, and the metric used for evaluation. In the more dense graphs, it can be seen that the random approach of assignment of nodes to clusters generally scores close to zero when evaluated using MQ or modularity. This is to be expected as both of these metrics lie in the range $[-1, 1]$, with zero being equivalent to a clustering with no level of structure. In the less dense graphs, sometimes the random approach does score slightly above zero for a low input probability p , when the graphs are less random. This is because of the fact that our random approach does rely on nodes to be connected to the clusters they are added to, reducing the number of options for less well connected nodes. For higher density graphs this is not evident as a node will have a larger set of clusters it can be assigned to.

It is clear from each of these figures that a graph scores well when it is rated with a metric that is also used as the heuristic to build the clusters. It also seems surprising that using MQ as a heuristic often results in a high average per cluster clustering coefficient, however looking at the standard deviation of the per cluster clustering coefficients shows that the

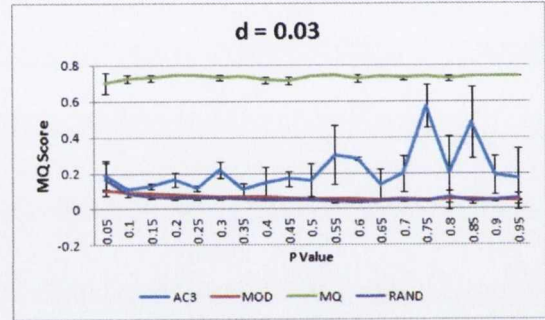
individual clusters vary wildly in quality. This is a result of a very imbalanced clustering, which will not be of benefit to a user if the majority of nodes are placed in a cluster with a low average clustering coefficient. This means that the nodes within the cluster will be less strongly related to each other.

Lowest Density Graphs ($d_1 = 3$)

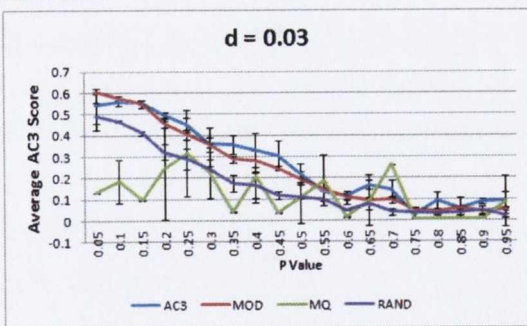
Figure 3.15 shows the resulting modularity, MQ and clustering coefficient values when the algorithm is run on graphs of increasing randomness with a relatively low density, $d_1 = 3$. Due to the relatively low density, there are fewer nodes to be chosen from when adding new nodes to the clusters, so nodes being added to a cluster are more likely to closely relate to several of the other nodes within the cluster. This is reflected by the higher scores for the random layout approach for each heuristic for a low levels of rewiring probability.



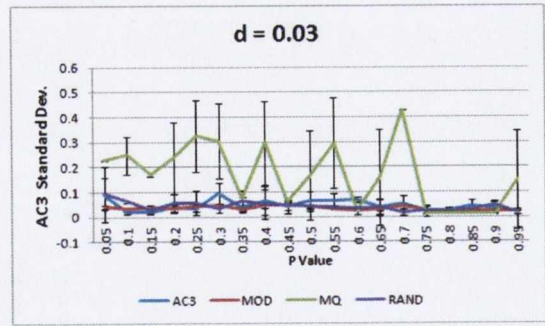
(a) Resulting graph modularity for each heuristic.



(b) Resulting graph MQ score for each heuristic.



(c) Resulting average cluster clustering coefficient for each heuristic.



(d) The standard deviation of the average cluster clustering coefficient of each of the four clusters for each heuristic.

Figure 3.15: Evaluation of graphs with 200 Nodes and a density of 0.03 ($d_1 = 3$), and an increasing level of randomness, denoted by p value.

Rating the graph based on modularity (figure 3.15a) results in the best results for the MOD heuristic with the AC3 approach not far behind for the more structured graphs ($p = 0.1$), but the gap widening as structure randomness increases. The reason for the large drop off in clustering coefficient performance is to do with the basis for clustering

coefficient as a calculation. If a node has less than two neighbours we assign a clustering coefficient of 0. In a graph with 200 vertices and 600 edges many nodes will have less than two neighbours. Those nodes that have 2 or more neighbours are also quite unlikely to have any neighbourhood overlap. The ability to calculate useful clustering coefficients depends not only on edge count, but also the edge distribution. When the graph is more structured we are more likely to see an overlap in neighbourhood, hence the closeness in performance between for the more structure graphs. MQ scores very poorly, while the random approach does surprisingly well. This is due to the constraint on nodes only being randomly assigned to clusters that they are connected. The low number of connections means that node may have only one viable cluster to be added to, each iteration.

Rating the graph based on MQ (figure 3.15b) results in a consistently high score regardless of the level of randomness when using MQ as a heuristic. This is true for all densities of graph evaluated. Using AC₃ as a heuristic results in a low score of 0.2 for the more structured graphs, which increases while fluctuating as the graph becomes more random. The MOD heuristic performs worse starting at approximately 0.1 and diminishing to less than 0.05.

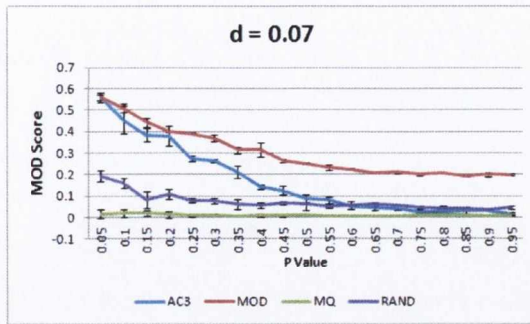
Rating the graph using AC₃ (figure 3.15c), we can see that AC₃ and MOD perform best and they also perform similarly for the full range of graph structure. The MQ heuristic produces lower quality result with a larger variation for each of the 3 input graphs, where the other approaches are more consistent in their results. Examining the AC₃ of each of the clusters created by the MQ heuristic (figure 3.15d), shows that the quality of the clustering is quite poor until the graphs become more random. A large standard deviation of the average of the AC₃ scores of the 4 clusters indicates that while some clusters have a high clustering coefficient others will have a very poor one. This means the MQ does not in fact provide a good consistent average cluster clustering coefficient, therefore clusters will be created where adjacent nodes do not have many mutual neighbours, and will be less conceptually alike.

Conclusions: For graphs of the lowest density ($d_l = 3$) MOD is the preferred heuristics across all levels of structure. However AC₃ performs very closely in terms of modularity. As the graphs become more random modularity becomes the sole preferred heuristic, in terms of reducing the number of edge crossings.

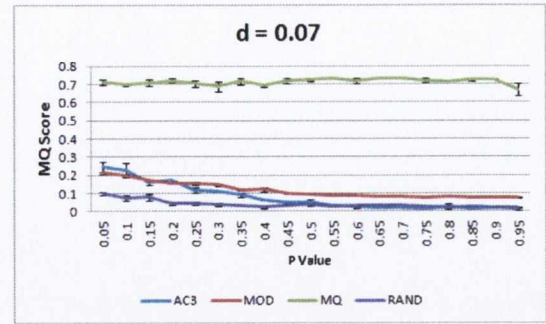
Low Density Graphs ($d_l = 7$)

Figure 3.16 shows the resulting MOD, MQ and AC₃ values when the algorithm is run on graphs of increasing randomness with a relatively low density, $d = 0.07$, $d_l = 7$. Due to the relatively low density, there are fewer nodes to be chosen from when adding new nodes to the clusters, so nodes being added to a cluster are more likely to closely relate to several of the other nodes within the cluster. This is reflected by the higher scores for the random

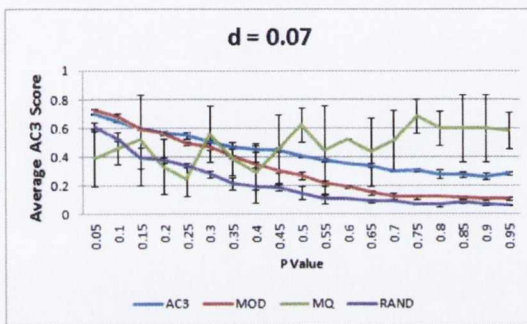
layout approach for each heuristic for a low levels of rewiring probability.



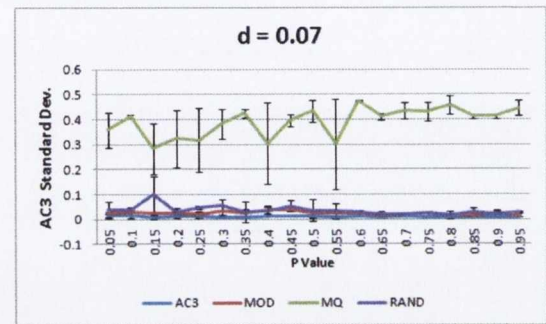
(a) Resulting graph modularity for each heuristic.



(b) Resulting graph MQ score for each heuristic.



(c) Resulting average cluster clustering coefficient for each heuristic.



(d) The standard deviation of the average cluster clustering coefficient of each of the four clusters for each heuristic.

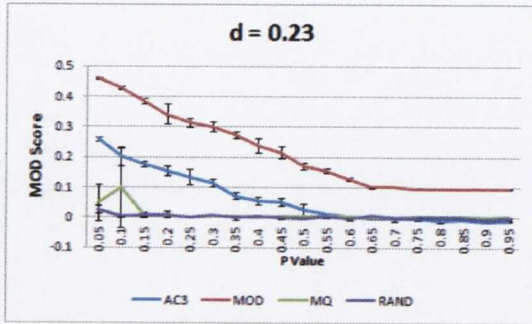
Figure 3.16: Evaluation of graphs with 200 Nodes and a density of 0.07 ($d_1 = 7$), and an increasing level of randomness, denoted by p value.

Rating the graph based on MOD (figure 3.16a) results in the best results for the MOD heuristic and, once again, the AC3 approach is not far behind. The MQ approach noticeably scores similarly to the random approach. Rating the graph based on MQ (figure 3.16b) results in a consistently high score regardless of the level of randomness when using MQ as a heuristic. Using the average cluster clustering coefficient as a heuristic results in a low score of 0.2 for the more structured graphs, but this diminishes toward 0.0 as the graph becomes more random, making it no more effective than the random approach. The MOD heuristic scores similarly to the AC3 heuristic, for structured graphs and also diminishes, but to a lesser degree than the AC3 approach.

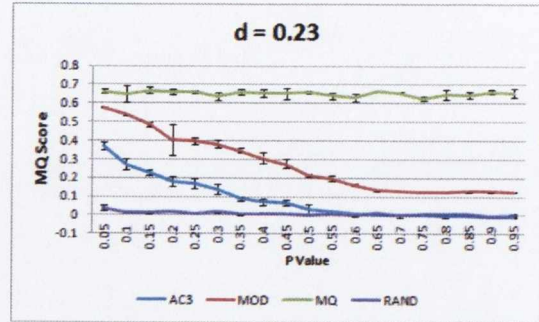
Rating the graph using AC3 (figure 3.16c), we can see that AC3 and MOD heuristics perform similarly for structured graphs and diverge as the graphs become more random. We can also see from the error bounds that the MQ heuristic produces varying results for each of the 3 input graphs, where the other approaches are consistent in their results. Even with the large error bounds, once the graph becomes sufficiently random, MQ appears to provide the highest average AC3 score of the resulting clusters. However, if we look at the standard deviation of the AC3 score of each of the clusters created by the MQ heuristic

(figure 3.16d), as for the the $d_1 = 3$ set of graphs, we can see that the quality of the clustering is very poor and the resulting clusters will not be conceptually alike.

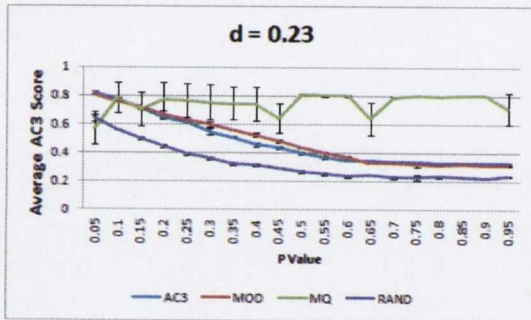
Conclusions: For the test graphs of density $d_1 = 7$ AC3 and MOD are the preferred heuristics when the graph contains structure. As the graphs become more random MOD becomes the sole preferred heuristic.



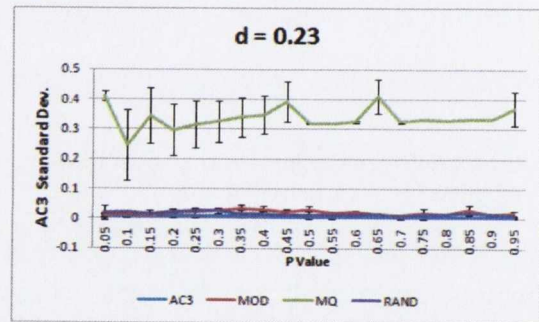
(a) Resulting graph modularity for each heuristic.



(b) Resulting graph MQ score for each heuristic.



(c) Resulting average cluster clustering coefficient for each heuristic.



(d) The standard deviation of the average cluster clustering coefficient of each of the four clusters for each heuristic.

Figure 3.17: Evaluation of graphs with 200 Nodes and a density of 0.07 ($d_1 = 7$), and an increasing level of randomness, denoted by p value.

High Density Graphs ($d_1 = 23$)

Changing the density of the graph has an impact on the performance of each of the heuristics. Figure 3.17 show results for graphs with a high density, $d = 0.23$, $d_1 = 23$. Rating the graphs by the modularity metric, we see that they overall scores are lower than the previous less dense graphs. We can also see that while AC3 performs much better than the random or MQ based heuristics, MOD scores nearly double for most levels of structure. When we examine the MQ ratings, we can see MOD and AC3 perform better than before as heuristics. This is due to the increased edge density, as more clusters now have a higher level of intra-cluster edges, compared to the maximum possible. MOD also out performs AC3 in a manner similar to how it did when using modularity as a metric. Modularity MQ

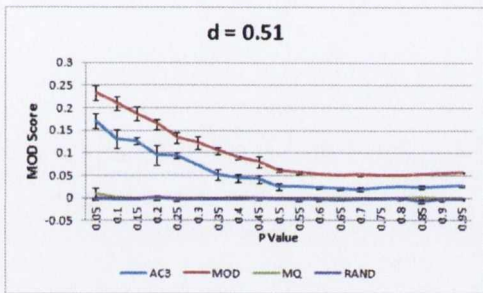
score scores very highly as expected.

When we rate the graphs based on AC₃ score we can see that, as has occurred in the other density of graphs, MQ offers an imbalanced clusterings. The AC₃ and MOD heuristics perform best and track closely together for all levels of structure.

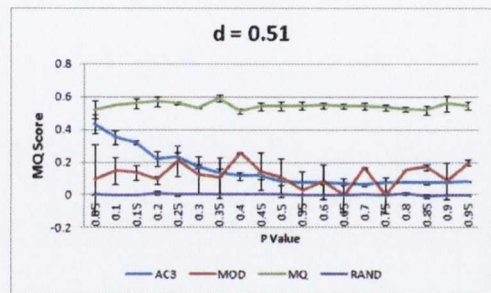
Conclusions: MOD is the clear preferred heuristic. AC₃ performs the second best. It results in more inter-cluster edges, indicated by its modularity score. However it does also form cohesive clusters, as indicated by the average cluster clustering coefficient.

Very High Density Graphs ($d_1 = 51$)

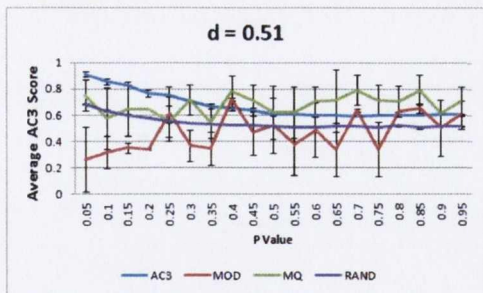
Figure 3.18 show results for graphs with a high density, $d = 0.51$, $d_1 = 51$. Rating the graphs based on modularity (figure 3.18a), we see that overall the scores are lower when compared to the less dense graphs. The difference between AC₃ and MOD for the structured graphs is more pronounced than before.



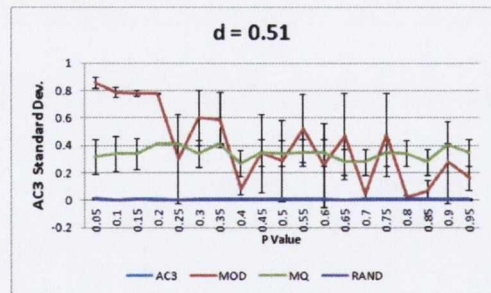
(a) Resulting graph modularity for each heuristic.



(b) Resulting graph MQ score for each heuristic.



(c) Resulting average cluster clustering coefficient for each heuristic.



(d) The standard deviation of the average cluster clustering coefficient of each of the four clusters for each heuristic.

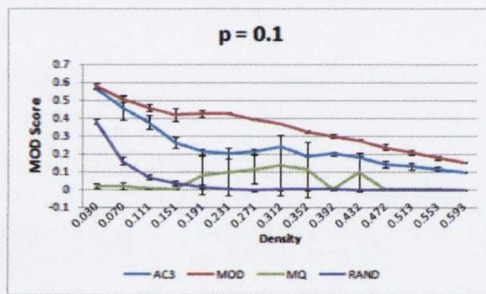
Figure 3.18: Evaluation of a graphs with 200 Nodes and a density of 0.51 ($d_1 = 51$), and an increasing level of randomness, denoted by p value.

Rating the graphs based on MQ score (figure 5.3d), we again see the MQ heuristic performs well. A noticeable difference is the improved performance of the AC₃ heuristic for the more structured graphs. The MOD heuristic performs better relative to AC₃ as graphs become more random, but the scores are less consistent, with larger standard deviations.

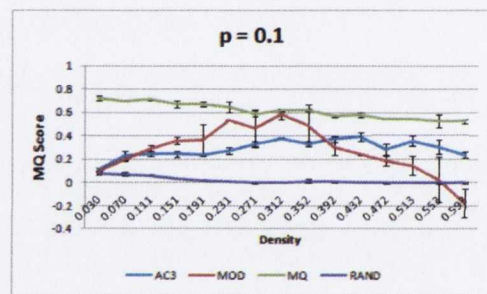
Rating the graphs based on average AC₃ scores results in the clustering coefficient heuristic performing the best for the more structure graphs. As the graphs become more random, the MQ heuristic performance does appear to perform slightly better, but the larger standard deviation in results across the input graphs reveals it does not do so. Also, as for the less dense graphs, the standard deviation of the average AC₃ score (figure 3.18d) of each cluster is much higher than the AC₃ approach. It is noticeable that for most of the graphs the MOD heuristic performs even worse than using the random approach and that the clusters generated vary largely in average clustering coefficient.

Conclusions: AC₃ is the most consistently high performing heuristic across all metrics. MOD results in a large deviation in the AC₃ score of individual clusters within a graph as long as there is some structure in it.

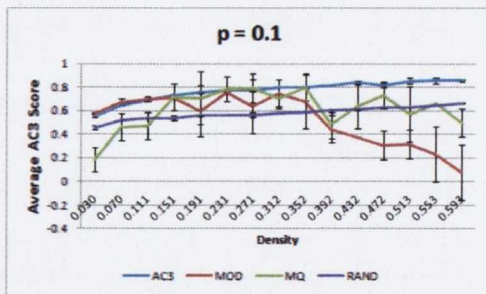
Low Randomness Graphs



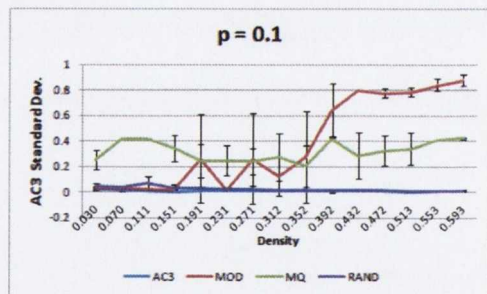
(a) Resulting graph modularity for each heuristic.



(b) Resulting graph MQ score for each heuristic.



(c) Resulting average cluster clustering coefficient for each heuristic.



(d) The standard deviation of the average cluster clustering coefficient of each of the four clusters for each heuristic.

Figure 3.19: Evaluation of a graph with 200 Nodes and a constant input rewiring probability $p = 0.1$, and an increasing density.

These are the graphs which exhibit small world properties. Rating the graphs based on modularity (figure 3.19a), we see the MOD heuristic score best as expected, however the score decreases as the graphs become more dense. AC₃ scores best out of the other heuristics and also decreases similarly to the modularity approach as the graphs become

more dense. For the lower density graphs, MOD and AC₃ are quite close. When the density becomes greater than 0.15, the difference becomes larger, which is also shown in figure 3.17a.

Using MQ as a heuristic, MOD behaves erratically, with large error bars and scores worse than random for the less dense graphs, and similar to random for the more dense graphs, with a large standard deviation.

Rating the graphs based on MQ (figure 3.19b), we see the expected high score for MQ. Interestingly we see low scores for MOD for both the less dense and most dense graphs, however for graphs in the mid range of densities it does improve considerably, just about outperforming the clustering coefficient approach.

Looking at the average AC₃ score (figure 3.19c), we see clustering coefficient performs the best at all densities, with close competition from modularity at lower graph densities. The average AC₃ rating for the MQ heuristic still exhibits a large standard deviation between individual clusters (3.19b) for all densities. For more dense graphs, the use of MOD as a heuristic performs quite poorly.

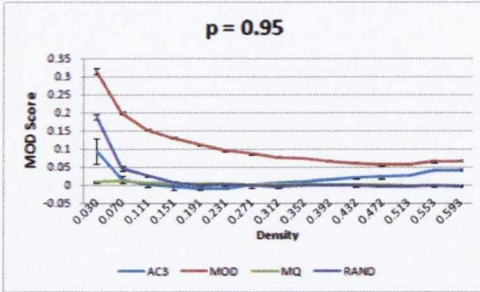
Conclusions: The AC₃ heuristic performs relatively well across all levels of density for all metrics. The closest rival is MOD, which is similarly effective, in terms of average AC₃ score until a density of approximately 0.38 ($d_l = 38$) is reached. Modularity does provide the fewest inter-cluster edges for the full range of densities, but it is quite small for the lowest and highest density graphs.

High Randomness Graphs

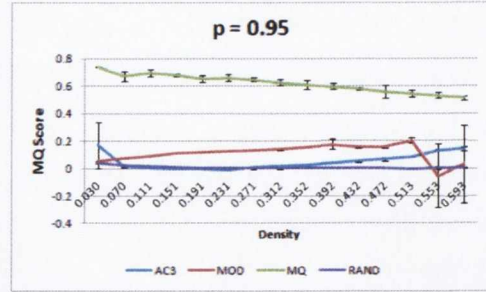
These are the graphs which exhibit a high level of randomness, and thus exhibit no small world properties. These graphs can give us insight into what approaches are affected most by the absence of a high clustering coefficient. All heuristics other than modularity perform poorly when rated using graph modularity (figure 3.20a). Rating the graph using MQ (figure 3.20b), we see, as expected, MQ performs very well, with MOD performing poorly but better than random or the AC₃.

When we rate the graphs using average AC₃ score (figure 3.20c) we see that there are some small improvements over RAND using MOD and AC₃ as heuristics, and that as graph density increases the scores for these approaches increase in a manner similar to the random approach. This is to be expected given the random nature of the graph. From this figure and figure 5.3d, we can see that when using MQ as a heuristic and rating the final clustered graph using MQ, the results are not reliable as they appear to be independent of graph randomness and only slightly affected by graph density.

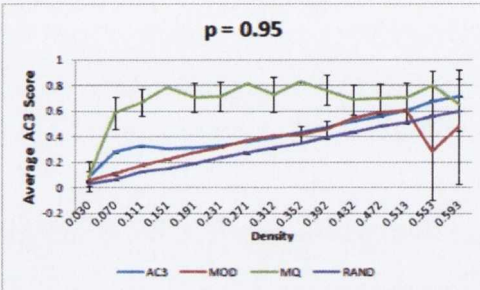
Conclusions: Overall the best heuristic for graphs which are more random and less structured appears to be MOD, until the graphs become very dense $d = 0.51$, ($d_l = 51$), when AC₃ becomes marginally better. This is due to the fact that when the graphs become extremely dense, the global clustering coefficient of the graph increases.



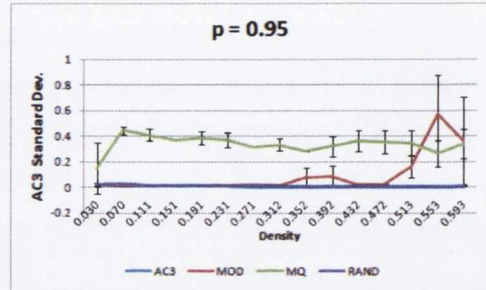
(a) Resulting graph modularity for each heuristic.



(b) Resulting graph MQ score for each heuristic.



(c) Resulting average cluster clustering coefficient for each heuristic.



(d) The standard deviation of the average cluster clustering coefficient of each of the four clusters for each heuristic.

Figure 3.20: Evaluation of a graph with 200 Nodes and a constant input rewiring probability $p = 0.95$, and an increasing density.

3.6.3 Comparison with Edge Betweenness Centrality Clustering

To provide a comparison with a state of the art clustering approach we performed a similar analysis on our test data set having applied Edge Betweenness Centrality clustering [NG04], using the $d_l = 7$ and $d_l = 51$ densities of graphs. This is a top town clustering approach which tries to find naturally occurring clusters within the data. Unlike our approach, the number of clusters is not usually specified and there is no equivalent of a user specifying nodes of interest. However it is an effective algorithm which can distinguish the clusters which naturally occur within a small world graph. The algorithm generates a hierarchy of partitions. The partitioning with the best modularity score is chosen from this hierarchy as the final clustering. This can result in a high number of clusters depending on the density and structure of the graph, as can be seen in figure 3.23. In many cases a very large number of clusters are created. Therefore, for our comparison we are constraining the number of clusters formed by the Newman and Girvan approach to 4, the same number used for our agglomerative clustering analysis. Evaluation uses the same approach as that used for evaluating our clustering heuristics and the results can be seen in figures 3.21 and 3.22.

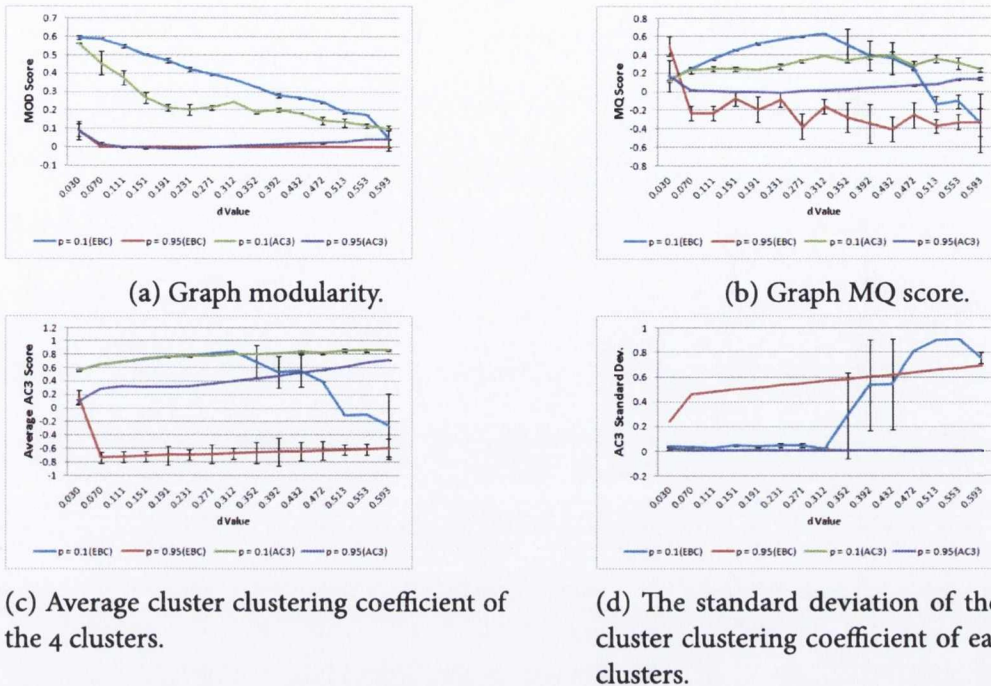


Figure 3.21: Evaluation of test graphs when clustered using Newman and Girvan's Edge Betweenness Centrality clustering (EBC) and our clustering coefficient heuristic (AC₃) for comparison. The graphs display the metrics for well structured ($p = 0.1$) and unstructured graphs ($p = 0.95$) of increasing density, where the number of clusters is constrained to 4.

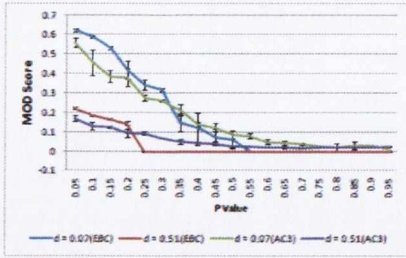
Lower Density Graphs ($d_1 = 7$)

Compared to the AC₃ heuristic agglomerative approach, the cluster count limited version of Edge Betweenness Centrality clustering predictably scores higher on modularity (figure 3.22a). This is as the clustering is not constrained by the user specifying nodes of interest to form the basis of clusters and the algorithm is very effective at finding the small amount of clusters in the more structured graphs (see figure 3.23). Predictably as the graph becomes more random this difference diminishes until the AC₃ approach produces cluster with a higher level of modularity, as there are fewer naturally occurring communities for more random graphs.

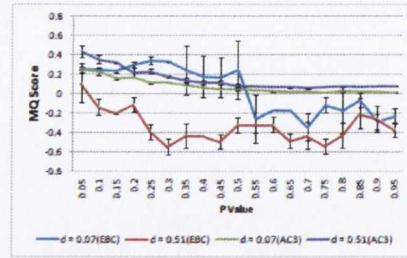
For the MQ score, (see figures 3.22b , 3.21b) Edge Betweenness Centrality clustering is superior, however the difference is not as large, and once the graphs become less structured the performance of the approach drops off significantly. In terms of average AC₃ score, Edge Betweenness Centrality clustering performs similarly for the more structured graphs but drops off significantly as the graphs become more random.

Very dense graphs ($d_1 = 51$)

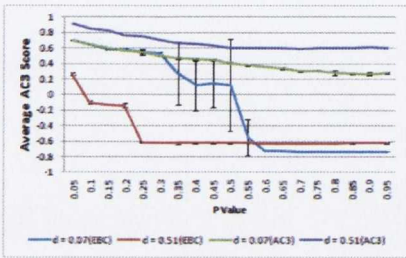
From figure 3.21 it can be seen that graphs with stronger small world graph characteristics modularity is slightly better for Edge Betweenness Centrality clustering, but the AC₃ ap-



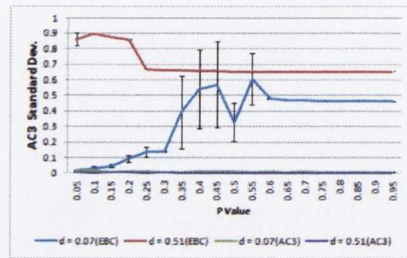
(a) Graph Modularity.



(b) Graph MQ score.

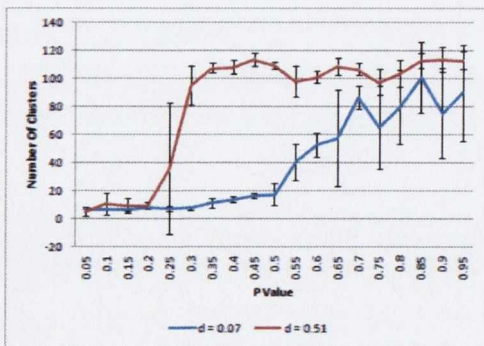


(c) Average cluster clustering coefficient of the 4 clusters.

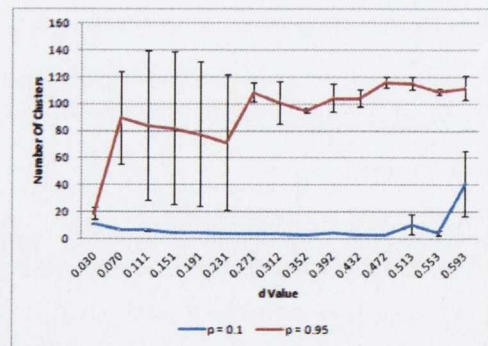


(d) The standard deviation of the average cluster clustering coefficient of each of the clusters.

Figure 3.22: Evaluation of test graphs when clustered using Newman and Girvan’s Edge Betweenness Centrality clustering (EBC) and our clustering coefficient heuristic (AC3) for comparison. The graphs display the metrics for low density ($d = 0.07$) and the highest density graphs ($d = 0.51$) with an decreasing level of structure (p increasing), where the number of clusters is constrained to 4.



(a) Graphs with a constant density.



(b) Graphs with a constant rewiring probability

Figure 3.23: Number of clusters generated using Edge betweenness Centrality Clustering.

proach performs better once the graphs become slightly more random (at approximately $p = 0.2$, so the underlying structure is still quite strong). However for the MQ score we find that, for the more dense graphs, the AC₃ heuristic consistently outperforms the Edge Betweenness Centrality clustering approach. Our AC₃ approach also provides equivalent and better average clustering coefficient for clusters and far higher clustering coefficient values for the more random graphs (due to all of the singleton clusters). Our approach also maintains more consistently high average AC₃ scores for the most dense graphs ($d > 0.352, d_1 > 35$) than Newman and Girvan's approach. The low standard deviation between the AC₃ scores also indicates that the resulting average cluster clustering coefficient is balanced across multiple clusters.

3.6.4 Evaluation Conclusions

Based on the preceding analysis the most consistently effective heuristics for agglomerative clustering around nodes of interest are AC₃ and MOD. Where a graph has small world characteristics, AC₃ performs very well and produces clusters with a high average cluster clustering coefficient that is balanced across all clusters. The MQ scores for all heuristics other than MQ are generally quite low, but the AC₃ heuristic does perform well for dense graphs and with a high level of structure. The AC₃ was also more stable when run over different graphs generated with the same input parameters, as evidenced by the smaller error bars on the preceding charts.

Modularity (MOD) also works as an effective heuristic for agglomerative clustering, and is more effective than AC₃ approach when the graphs become more random. Its effectiveness is mainly in terms of reducing inter-cluster edges (i.e. having a high score when rated by modularity), but it also still produces clusters with a high average AC₃ score.

It is worth emphasising that the purpose of using the AC₃ heuristic is not only to reduce inter-cluster edges, it is also provide clusters where a nodes neighbours in a cluster have similar relationships to the node. This will not always align with edge crossing reduction as a node will be clustered with its neighbours that have a similar neighbourhood, rather than with its neighbours that have completely disjoint neighbourhoods.

MQ performed the least successfully of the heuristics when used for agglomerative clustering, particularly in terms of producing a high average AC₃ score across clusters, with a balanced distribution across all clusters.

We also compared our agglomerative approach using AC₃ as a heuristic to Newman and Girvan's Edge Betweenness Centrality algorithm, constrained to produce four clusters. The comparison is not a direct one as the agglomerative algorithm focuses on building clusters around nodes of interest and the betweenness centrality algorithm defines clusters without any such constraints. As expected the Edge Betweenness Centrality clustering algorithm performs very well on structured graphs with low density. However as the graphs

become more dense the agglomerative algorithm, using AC_3 as a heuristic, performs close to the level of the centrality algorithm and by some metrics (MQ and AC_3 score) it outperforms the algorithm for graph with a density of $d = 0.255$, $d_l = 25.373$. Given that the agglomerative approach is designed to focus around nodes of interest to aid in visualisation rather than discover communities, we feel our algorithm compares favourably with the Edge Betweenness Centrality algorithm.

3.7 History of Infoviz Data-Set Example

In order to further show the practicality of our approach to agglomeratively build clusters around nodes of interest, we present an example using a known data set. The “history of infoviz” data-set [YuFo8] is a benchmark data set that was originally presented as part of the IEEE 2004 Infoviz Data Contest. The nodes of the graph produced from the data set represent papers published in the field of visualisation. The edges of the graph represent citations between papers in the data set. The graph contains 605 nodes and 1953 edges.

Author information has been provided for each article. 419 of the 605 papers contain keywords, which are the original keywords submitted by the authors of the paper. Sanitised forms of the keywords, in terms of spelling and similar terminology, have been provided as part of the data set. These keywords can be considered an identified set of features, which can be used to categorise the nodes. The data set can be seen in figure 3.24. It consists of articles from the field of information visualization (often referred to as infoviz) from 1995 to 2002, as well as the citations between them, which are represented as edges. An example of a case study using this graph is given by van Ham [vHo4], and the data set has also been used as an evaluation dataset for later research such as CiteWiz [ET07], an application for visualising scientific citation networks. As van Ham notes, not all of the provided keywords are useful. For example the keyword “information visualisation” is specified for 100 of the records, but does not help distinguish the contents of the such papers from the others in the data set. The graph can be considered a small world graph as it has an average path length of 3.974 and an average local clustering coefficient value of 0.158. An equivalent random graph was generated for comparison and had an average path length of 3.657 and an average local clustering coefficient value of 0.0115, a full order of magnitude smaller. While the linear density of the graph is relatively low compared to some of our experiment graphs ($d_l = 3.23$) the distribution of edges is different and it can be seen in figure 3.24 that the density of the graph impacts layout, resulting in a very dense core of nodes.

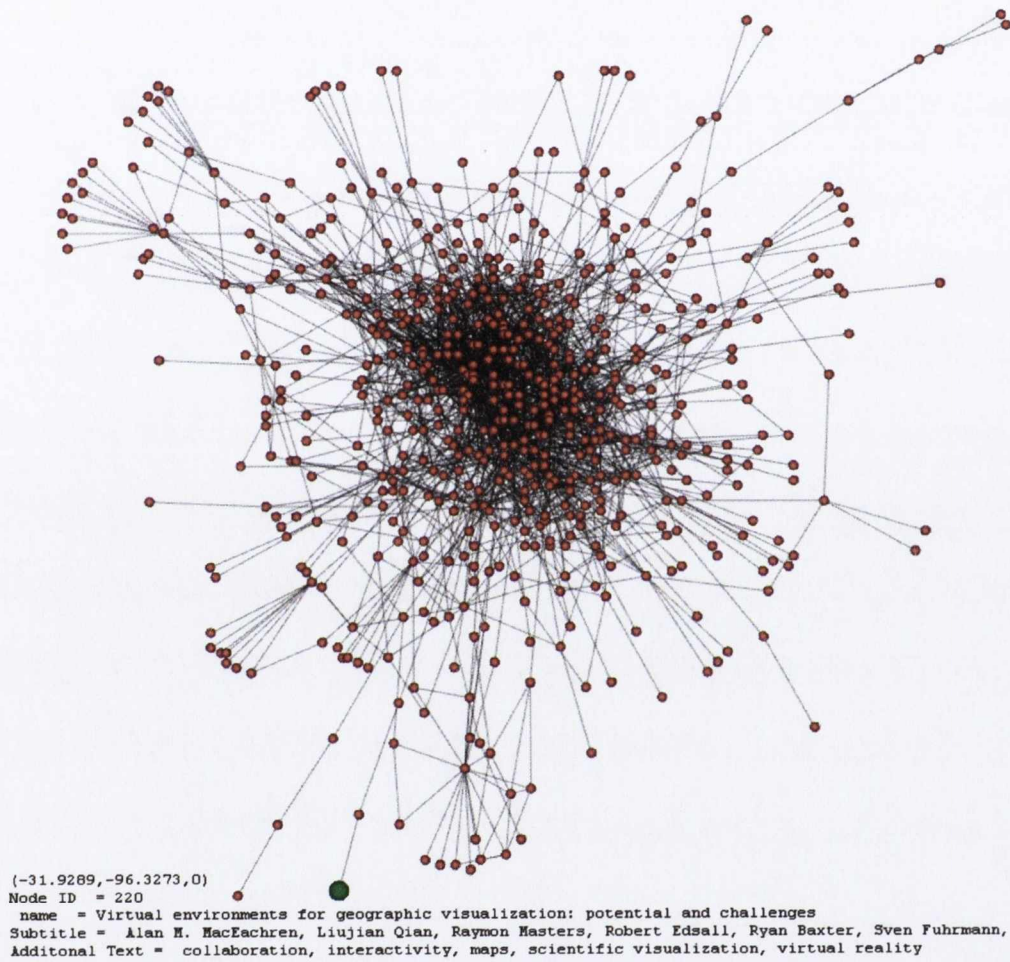


Figure 3.24: The infoviz data set laid out using Hachul and Jünger's FM₃ multi-level layout algorithm with a input inter-node distance of 15 (equivalent to a k value of 15). Each node in the image has a radius of 1. The implementation used is the Open Graph Drawing Framework [TDoG13] version of the FM₃ algorithm. There are 605 nodes and 1953 edges. The text overlay displays the information for the highlighted green node.

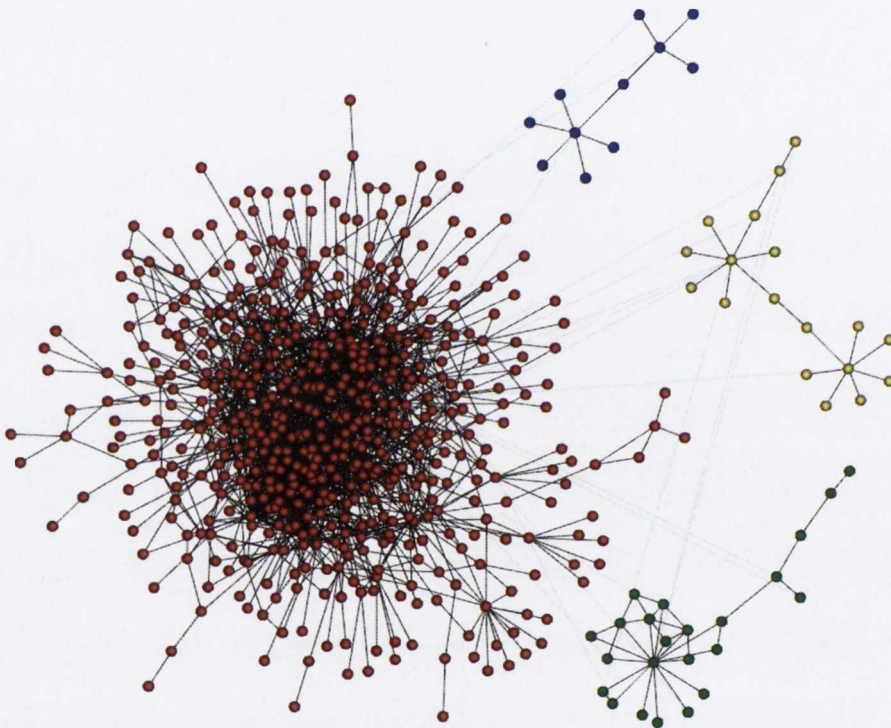


Figure 3.25: The infoviz data set split into 4 clusters using edge betweenness centrality clustering. Each cluster is laid out using our Fruchterman Reingold implementation (utilising a k -value of 5 and a grid size of $2k$). The colours are used to distinguish the clusters and do not imply a relationship with the nodes of interest used for agglomerative clustering.

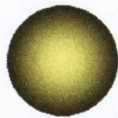



Title	Keywords	Cluster colour
Hierarchical flip zooming enabling parallel exploration of hierarchical visualizations	flip zooming; focus+context visualization; hierarchies; information visualization	
Techniques for nonlinear magnification transformations	data visualisation; domain constraint; efficiency; expressiveness; magnifications combination; multiple transformations combination; nonlinear magnification transformations; normal views; piecewise linear methods; smooth interpolation	
Pad++: a zoomable graphical interface system	authoring; hypertext; information navigation; information visualization; interactive interfaces; multimedia; multiscale interfaces; navigation; world wide web; zooming	
IVEE: an environment for automatic creation of dynamic queries applications	database query; dynamic queries; information exploration; information visualization; tight coupling	

Table 3.4: The nodes of interest for the history of infoviz example.

3.7.1 Clustering Approach

Using the edge betweenness centrality clustering algorithm on this data set produces 91 clusters. If we limit the number of clusters to a more practical values such as 4, the resulting graph contains 3 small clusters and one very large dense one on as seen in figure 3.25. Neither result allows the viewer to gain any further insight into the graph. There is no selection of nodes as an input to this top down clustering.

Figure 3.26 shows the result of using our AC₃ approach to creating four clusters. We have selected four papers as nodes of interest, detailed in table 3.4. Each of the papers has a list of keywords and has several well connected neighbours. For the figures displaying the resulting graphs we have utilised our own Fruchterman Reingold implementation for the cluster layout and have faded the intensity of the inter-cluster edges in order to make the internal structure of the clusters more apparent.

3.7.2 Clustering Evaluation

For clustering around the four nodes of interest we utilise our modularity and AC₃ approaches, as well as the random breadth first search approach in order to provide a heuristic free comparison. We omit using MQ score as a heuristic and metric due to its poor performance as a heuristic in the previous experiment.

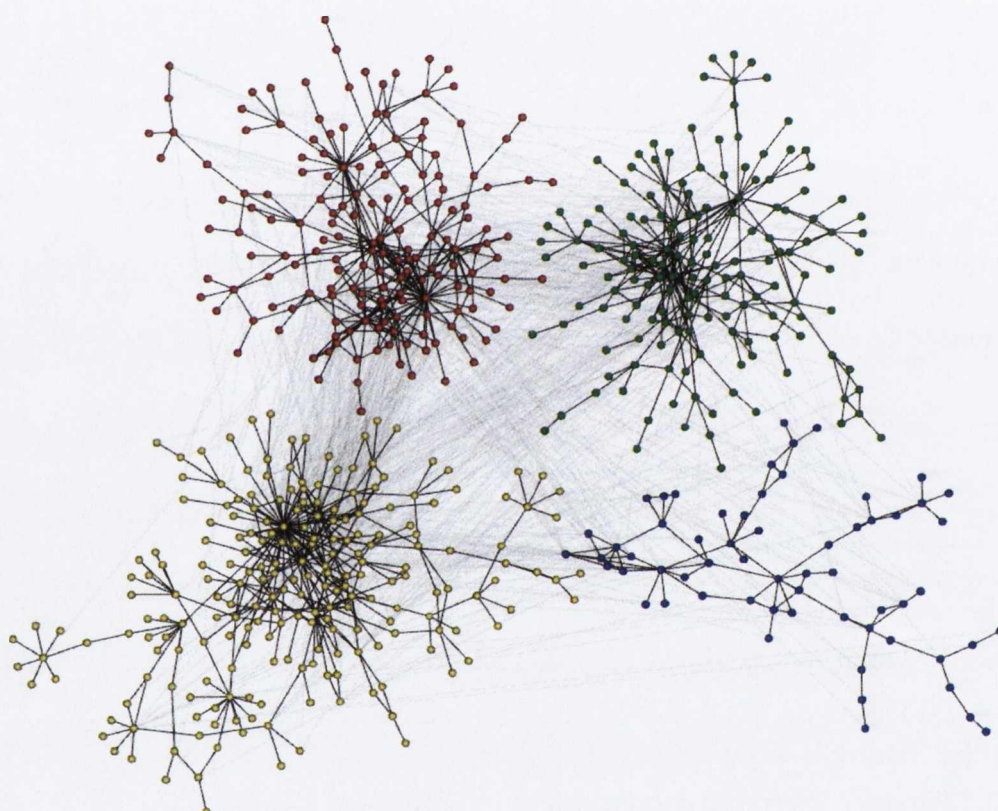


Figure 3.26: The infoviz data set split into 4 clusters using the AC₃ approach. Each cluster is coloured based on the cluster colour associated with each node of interest, seen in table 3.4.

Chosen Heuristic	AC ₃ Score	CC std. Dev	Mod score
AC ₃	0.24657	0.0167651	0.251933
MOD	0.176172	0.0174256	0.372124
BFS	0.129411	0.0342558	0.173883

Table 3.5: The results for evaluating the infoviz graphs using the approach from section 3.6. The values for the randomised breadth first search are averaged over 3 runs of the clustering algorithm.

Heuristic Results

We performed an evaluation on the resulting clusterings using the approach described in section 3.6. The resulting scores can be seen in table 3.5. As with the previous experiments both the AC₃ and Mod heuristics performed best when used as their own evaluation metrics. The AC₃ score and modularity scores for the randomised breadth first search seem quite high for a random approach, particularly given the relatively low linear density of the graph. However it is worth noting that 135 of the 605 papers only have a single neighbour. This results in fewer inter-cluster edges (improving modularity) and a higher level of interconnectivity of node neighbourhoods (improving the average cluster clustering coefficient). Both the AC₃ and MOD heuristics show effectiveness when compared to the random BFS approach. The AC₃ approach provides clusters which are more strongly interconnected, with a larger overlap of node neighbourhoods indicated by the high clustering coefficient. The MOD approach (which can be seen in figure 3.28) results in lower ratio of intra-cluster to inter-cluster edges. The two approaches do provide quite different results in terms of how the nodes are clustered. For example the bottom right cluster around the node “IVEE: an environment for automatic creation of dynamic queries applications” contains far more nodes using the MOD approach. It is difficult to determine which is preferable for a user without the user of some higher level information, so we also examine the distribution of nodes containing a keywords related to each the nodes of interest.

Keyword Results

In order to demonstrate the difference in impact of our agglomerative approaches, we have chosen a distinctive keyword from each node of interest. These terms were “focus+context”, “hypertext”, “transform” and “query”, (see table 3.6). In order to obtain nodes containing both “query” and “queries” as a keyword the search term “quer” was user when selecting nodes by keyword.

In figures 3.27,3.28 and 3.29 we have coloured each node related to the specified keywords. All other nodes are coloured grey and each cluster is in the same relative position for each graph rendering. For example, the top left cluster always uses the paper “Pad++: a zoomable graphical interface system” as its node of interest. Table 3.7 shows how many







Keyword	Node Count	Cluster colour
focus+context	15	
transform	7	
hypertext	14	
query	30	
query, focus+context	30	
query, hypertext	30	

Table 3.6: The cluster colours associated with each keyword. The colours match the colours of the original nodes of interest that are associated with each keyword, except for the two cases where 2 keyword are applicable to a node.







	AC ₃		MOD		BFS	
	Matching Cluster	Other Cluster	Matching Cluster	Other Cluster	Matching Cluster	Other Cluster
	9	5	3	11	9	5
	17	13	10	20	8	22
	9	6	9	6	9	6
	3	4	2	5	2	5
	1	0	0	1	0	1
	1	0	0	1	1	0
Total	40	28	24	44	29	39

Table 3.7: Assignment of keyword nodes to clusters matching the original node of interest. The nodes with 2 matching keyword are considered matching if either of its selected keywords are shared with the node of interest in the assigned cluster.

nodes with with the specified keywords were matched to the cluster based around the corresponding node of interest. It can be seen that more nodes end up in the same cluster as their original node of interest for the AC₃ approach than for the modularity approach. For example using the modularity approach most of the nodes with the keyword “hypertext” (red nodes) end up in the cluster belonging to the node of interest associated with the keyword “transform” (blue node). When using the AC₃ approach most of the nodes associate with the term with “query” (Green) are assigned to the correct cluster. Using modularity as a heuristic these nodes are distributed across all cluster of the graph. Overall the AC₃ approach assigns 40 of the selected keyword nodes to a cluster with a matching node of interest, while the MOD approach only assigns 24. An iteration of the BFS approach even out performs the MOD approach in this case. The most probable reason for this is that MOD aims aims to reduce the number of inter-cluster crossing, taking into consideration the amount of connections between cluster. This may result in adjacent highly connected nodes being placed in a cluster together to reduce the number of inter-cluster edges. The AC₃ approach does not take take inter cluster edge count into consideration, only how well a nodes neighbours are interconnected, when a node is being added to a cluster. The BFS approach chooses neighbouring nodes at random, independent of their characteristics so

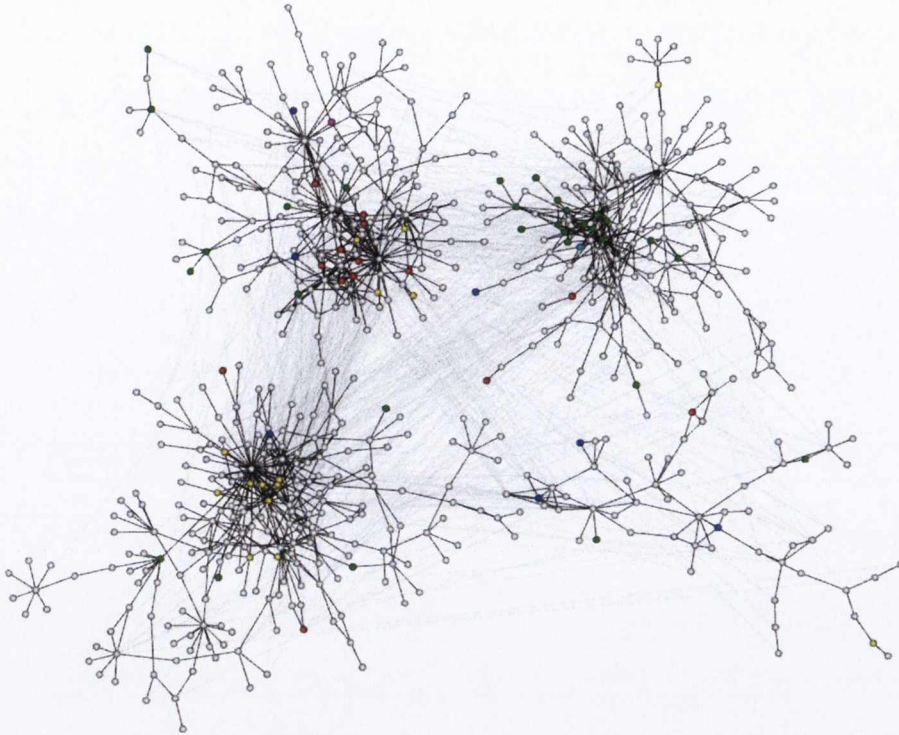


Figure 3.27: The infoviz data set split into 4 clusters using the AC₃ approach. Nodes are highlighted by keyword. Their colour corresponds to the colour of the node of interest associated to the keyword. The turquoise and magenta nodes represent papers which share 2 keywords. Each cluster has the same relative position as in figure 3.26.

there is no bias towards well connected nodes being assigned together.

Infoviz Data Set Conclusions

In terms of evaluating the heuristics using AC₃ and MOD as metrics there is no clear superior heuristic for the infoviz graph. However in examining the distribution of nodes with a shared distinctive keyword it is evident that the AC₃ approach resulted in more conceptually alike clusters. This evaluation provided an illustrative example of the use of the two heuristics to guide agglomerative clustering. The resulting clusters do depend on the initial choice of nodes and the distribution of keyword related nodes does depend on choice of a distinctive keyword. Use of a more vaguely applicable keyword term such as “information visualisation” would convey very little information to the user about cluster structure. Selection of a node with only one neighbour would most likely not result in clusters that are conceptually alike to the node of interest. However, it is evident that given a sensible choice of input node and a sensible node classifier, our agglomerative clustering approach, using clustering coefficient as a heuristic, provides the user with a different perspective on the graph structure which may be an aid to gaining insight.

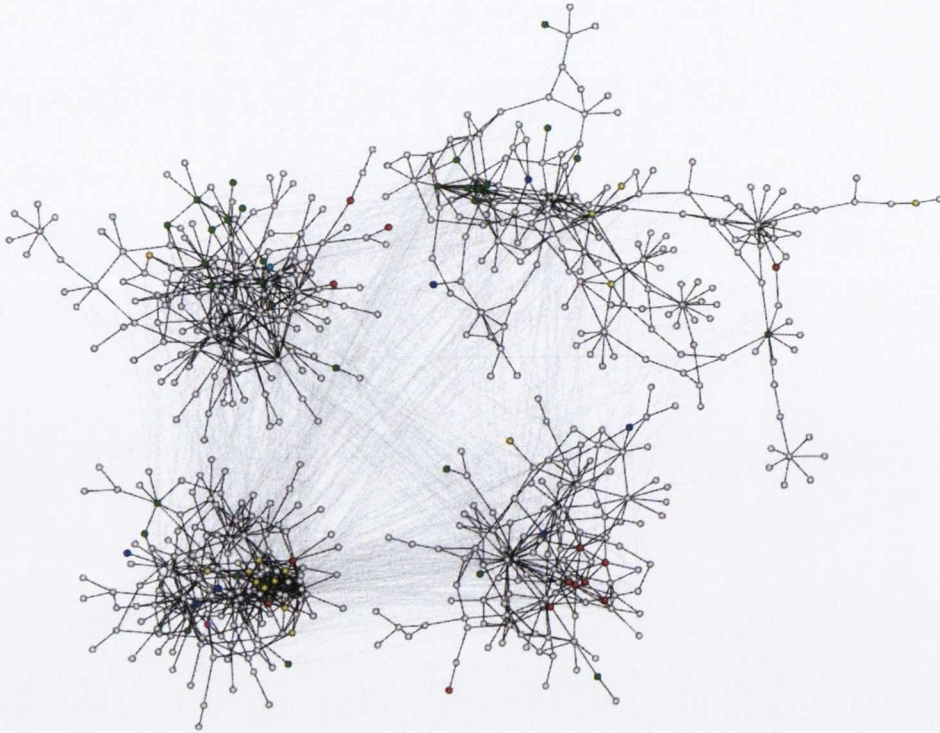


Figure 3.28: The infoviz data set split into 4 clusters using the modularity approach (MOD). Nodes are highlighted by keyword. Each cluster has the same relative position as in figure 3.26.

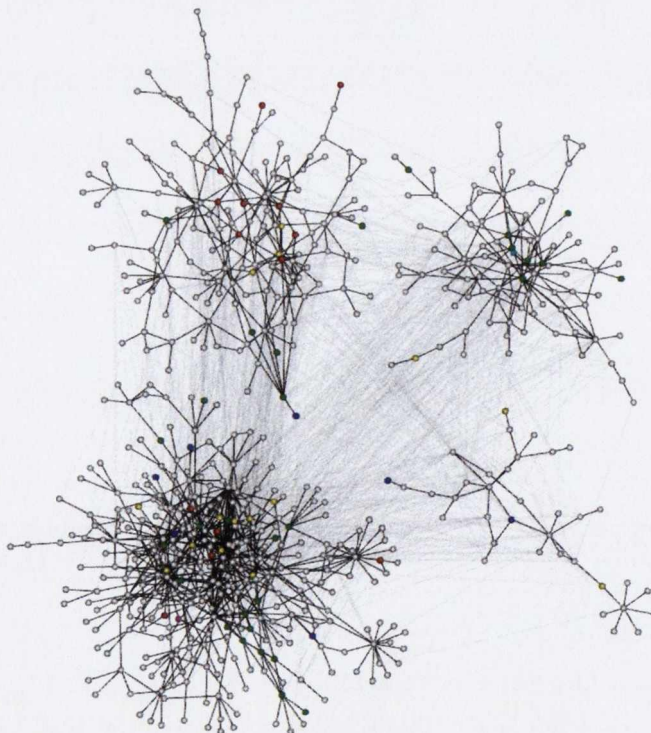


Figure 3.29: The infoviz data set split into 4 clusters using the randomised breadth first approach. Nodes are highlighted by keyword. Each cluster has the same relative position as in figure 3.26. This rendering is of the first of the 3 runs using this clustering approach.

3.8 Conclusions and Future Work

In this chapter we introduced our approach to agglomerative clustering of small world graphs around nodes of interest. An important difference between our clustering approach and many existing approaches is that we use a set of input nodes specified by the user as the basis for building the clustering. Each node of interest specified by the user forms a basis or a cluster, which is grown agglomeratively.

We initially suggested that average clustering coefficient of clusters could be an effective heuristic for our agglomerative clustering, testing its use on a mix of procedurally generated and real world graphs. We then evaluated using average clustering coefficient of a cluster (AC₃) as a heuristic, for agglomerative clustering around nodes of interest. We have compared it to using other heuristics to guide to agglomerative process, as well as comparing it to an established top down clustering algorithm. We have established that the average clustering coefficient of a cluster provides an effective heuristic to guide agglomerative clustering around nodes of interest.

We also provided a practical example using a benchmark dataset. The history of infviz dataset describes over 600 papers and the citations between them over several years in the field of information visualization, and exhibits small world graph properties. We demonstrated the effectiveness of the AC₃ approach to agglomerative clustering, using the author's keyword classifications as validation.

One possible avenue for future work is the automation of the selection of nodes of interest to create a more general approach, for when the user does not have nodes of interest in mind prior to graph analysis. For practical reasons, in our experiments we automated the selection of nodes of interest by selecting nodes based on node degree. This approach was purely practical, ensuring that the similar nodes were selected for each run of the algorithm for different graphs. There are other centralities such as vertex betweenness or node clustering coefficient which could be used to specify an initial node set for the agglomerative clustering. These centralities could possibly be used in conjunction with an independent set filtration, as used by the GRIP layout algorithm[GK01], to ensure a distribution of nodes across the diameter of the graph. Any approach used to generate an initial node set would need to be thoroughly evaluated experimentally against a wide range of input graphs and varying agglomerative clustering input node sets, to be sure that resulting clusterings are of high quality and stable across a range of input graphs. The performance of our agglomerative clustering with different sizes of automated initial nodes sets, on graphs with different characteristics such as size density and edge distribution could also prove significant in terms of automating the input node set.

In terms of further future evaluation, a higher level user qualitative evaluation, based around real data and a high level task and a user survey, such as that of Ridsen *et al.* [RCMCoo], would help gauge how useful subject matter expert users find different met-

rics to guide the agglomeration process. It would also help definitively determine how users would use the ability to rearrange graphs around nodes of interest, to aid in a task.

Chapter 4

Graph Layout

THE AESTHETICS OF GRAPH LAYOUT play a large role in the user understanding of a graph visualisation [Pur97]. In the preceding chapter we presented and analysed our approach for agglomerative clustering of small world graphs around nodes of interest. Our primary contribution of this chapter is the utilisation of our clustering approach, described in chapter 3, as the basis for a graph layout which allows for the use of edge routing techniques and structures the layout reflecting the relationship between the clusters. Our layout approaches are developed from our clustering approach by extending it from a flat hierarchy into a hierarchical clustering, making the associated graph a compound graph. The different layers of our clustering hierarchy are utilised as an aid to graph layout. The hierarchy is also used to route edges using Hierarchical Edge Bundling [Holo6]. We also address how to optimise the circular layout of clusters within a clustering hierarchy. This approaches described in this chapter are useful for generating graph layouts which reflect top level inter-cluster relationships. However laying out complex graphs is a difficult problem, so we acknowledge that rather than providing a complete solution our approaches provide an initial step which may be enhanced by future work.

Chapter structure: The chapter is structured as follows:

- In sections 4.1 and 4.2 we describe our motivation and the related work for this chapter.
- Section 4.3 describes our approach to improving the positioning of nodes in circular clusters.
- Section 4.4.1 describes how we extend our flat agglomerative clustering into a clustering hierarchy.
- Section 4.4.2 describes how we layout the cluster hierarchy using a hierarchical layout.

- Section 4.4.3 describes how we layout the cluster hierarchy using a multilevel layout.
- In section 4.5 we show the results of our approach to layout.

4.1 Motivation

When laying out graphs clustered with the approach described in the preceding chapter it is possible to lay out the clusters, separately using a force directed approach as we did for our influence graph example, in the preceding chapter (see figure 3.11), and as can be seen in figure 4.1. While this approach is effective for smaller graphs, there is not much information provided for edge routing and no consideration is given for the inter-cluster relationships of nodes.

Our motivation is to convey the relationships between the clusters generated based on the user's selection of nodes of interest. We propose that this can be achieved by the generation of a hierarchy reflecting the relationships between the graph's agglomerative clusters, and the layout of the graph utilising that hierarchy as an input.

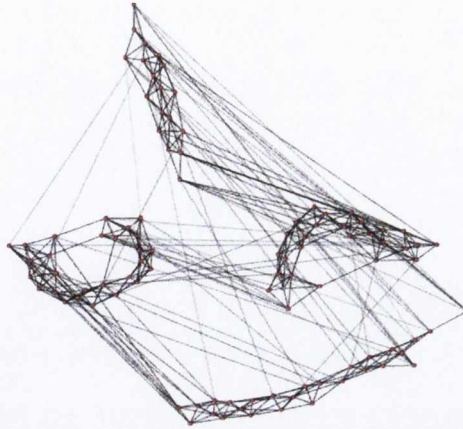


Figure 4.1: A 100 node procedurally created small world graph clustered around 4 nodes of interest. The cluster nodes are positioned using a per cluster Fruchterman-Reingold force directed layout.

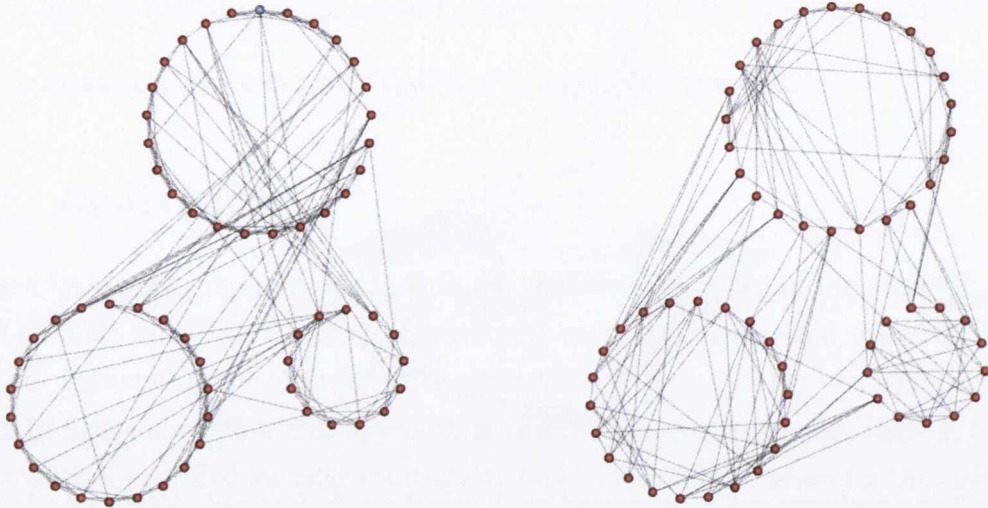
4.2 Related Work

An overview of graph layout approaches is given in section 2.3. Our approaches to the layout or circular clusters are influenced by the circle rotation approaches utilised by Topolayout [AMA07], [FLM95], as well as Symeonidis and Tollis [ST04]. Edge routing techniques are described in section 2.5. Holten's hierarchical edge bundling technique [Holo6], which we utilise in this chapter, is described in section 2.5.1.

Multilevel layouts are described in section 2.3.3. For our multilevel layout we follow the approach taken by Walshaw [Walo1] as this was the most suitable for the reuse of our clustering as a basis for generating a clustering hierarchy. An approach such as Hachul and Jünger's [HJ05] requires a more complex partitioning of data which was less readily adaptable to our clusters, as was the spectral partitioning approach used by Frishman and Tal [FT07] and the maximally independent vertex set approach of GRIP [GK01]. These methods are closely entwined to their approaches to graph coarsening in terms of the layout phase. We wish to maintain our concept of nodes of interest throughout our multilevel, hierarchy. Walshaw's use of the Fruchterman and Reingold's force directed layout is also suitable for application to our generated hierarchy with the least amount of modification.

4.3 Circular Layout of Clusters

As described in section 2.3.6 circular layouts provide a simple approach to graph layout. Figure 4.2a shows a small graph with 3 clusters, each laid out using a circular layout. This graph is laid out simply with no consideration given to the number of edge crossing resulting from the node positions. We utilise an intelligent initial placement of nodes combined



(a) Unoptimised layout with 1392 edge intersections and 459 inter-cluster edge intersections.

(b) Optimised layout with 772 edge intersections and 79 inter-cluster edge intersections.

Figure 4.2: An example small world graph containing 60 nodes and 180 edges, illustrating the impact of node reordering combined with cluster rotation. The graph is clustered using our approach with clustering coefficient as a heuristic. The three nodes with the largest degree have been selected as nodes of interest.

with a force directed rotation of the circular clusters in order to reduce the number of edge crossings as illustrated in figure 4.2b. As part of their topology based approach to graph layout [AMA07], Archambault *et al.* utilise force based rotation of circular cliques, minimising the torque caused by edges terminating outside of the clique, in an approach that they have adapted from Frick *et al.*'s GEM layout [FLM95].

4.3.1 Initial Node Ordering

We initially place nodes in each cluster in such a way that nodes with inter-cluster edges are positioned closer to their neighbour nodes, which lie in other clusters, as shown by the simple example in figure 4.3. To initially place nodes we calculate the average position of neighbours in other clusters for each node. We assume an even distribution of cluster nodes around each cluster's circumference. Each node has a potential position in a slot around the circumference. Each node is given an ideal position, where a line between the cluster center and the average neighbour position intersects the cluster circumference. The nodes are sorted in order of ideal position. The nodes are then assigned to the empty slot closest to their ideal position. If a node is assigned to a slot which is occupied by a preceding node it is assigned to the next available slot. This can lead to multiple nodes with similar ideal positions being assigned slots progressively further away from their ideal position as seen in figure 4.4b. This effect is counteracted by the rotation phase, as seen in figure 4.4c.

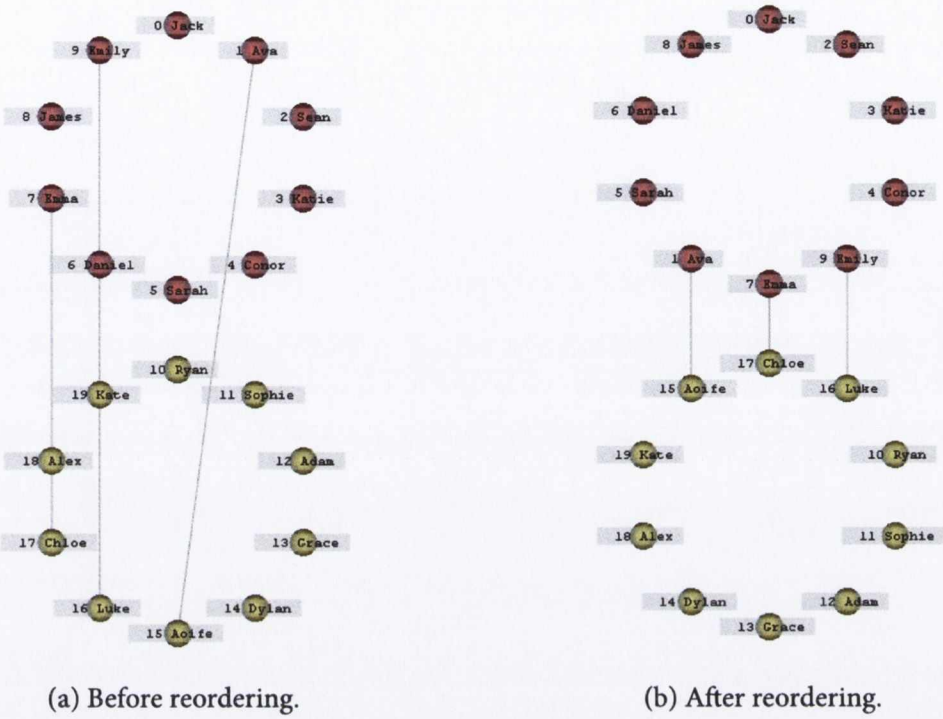


Figure 4.3: A simple example illustrating node reordering with 2 clusters.

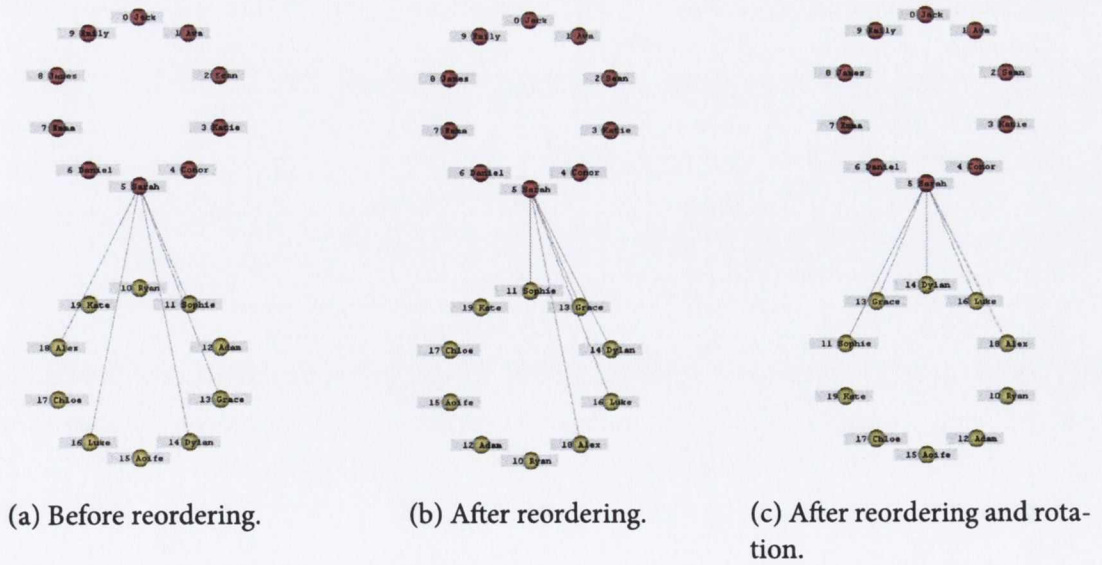


Figure 4.4: A simple example illustrating node reordering with overlapping ideal node positions, combined with rotation.

4.3.2 Cluster Rotation Implementation

Our approach is a version of the torque calculation of Archambault *et al.* [AMA07]. It is executed after the node ordering phase as this means cluster nodes with neighbours in the same cluster will be positioned near each other. It is essentially a force directed layout except the forces are not repositioning clusters, but rotating them about their normal axis. For our two dimensional graphs, each cluster's normal axis is perpendicular to the graph plane.

Given a circular clustering of p clusters $C = \{C_1, C_2, \dots, C_p\}$ where each element of C contains a disjoint subset of the graph $G = (V, E)$ such that $C_i = \{v_{i1}, v_{i2}, \dots, v_{in}\}$, $n = |C_i|$ and $C_i \subset V$, for each cluster C_i there exists a set of edges E'_{C_i} which contains the set of all edges $e_i = (v_x, v_y)$ where $v_x \in C_i$ or $v_y \in C_i$ but $\{v_x, v_y\} \notin C_i$, i.e. E'_{C_i} is the set of all inter-cluster edges which have a node in C_i . We denote the set of nodes which have neighbours in other clusters as V'_{C_i}

For each element $e_i \in E'_{C_i}$, we calculate the a vector \bar{v}_x which indicates the direction between v_x and average position of v_y , as v_x . We average \bar{v}_x for all edges in E'_{C_i} that contain v_x a We also calculate a vector \bar{p}_x describing the position of v_x relative to the center of cluster C_i . We calculate angle α_{e_i} in the range $[-\pi, +\pi]$ required to rotate the cluster C_i , in an anti-clockwise fashion about the cluster normal, so that \bar{p}_x is parallel to \bar{v}_x . For a two dimensional graph the cluster normal would be constant for all clusters. The final rotation applied to the cluster is the average of all such rotations for all $e_i \in E'_{C_i}$.

Algorithm 3 Algorithm for calculating rotation angle for an individual cluster.

```

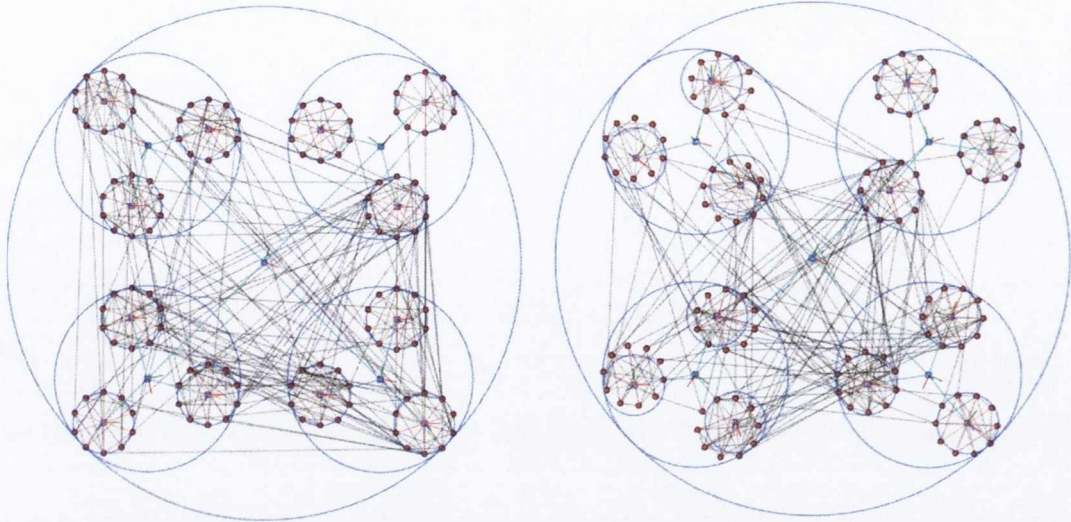
cluster_rotation = 0
for all  $e_i = (v_x, v_y) \in E'_{C_i}, v_x \in C_i, v_y \notin C_i$  do
     $\bar{v}_x = v_y.\text{position} - v_x.\text{position}$ 
     $\bar{p}_x = v_x.\text{position} - C_i.\text{position}$ 
     $\alpha_{e_i} = \text{getAntiClockwiseAngle}(\bar{v}_x, \bar{p}_x)$ 
    cluster_rotation = cluster_rotation +  $\alpha_{e_i}$ 
end for
cluster_rotation = cluster_rotation /  $|C_i|$ 

```

When calculating the angle between \bar{v}_x and \bar{p}_x , we use the dot product of the two vectors, in conjunction with a cross product to ensure that the rotation is anti-clockwise about the cluster normal. Applying multiple rounds of node ordering followed by cluster rotation will generally reduce node crossings further.

We apply this algorithm iteratively in a manner similar to a standard force directed layout. The algorithm is applied to all clusters, each iteration. We also utilise the force directed layout concept of a cooling function, to force the rotations to converge to a final value.

Our previous examples show simple flat clusterings, however if a clustering hierar-



(a) Unoptimised, with 7712 total edge intersections and 2092 inter-cluster edge intersections.

(b) Unoptimised, with 2803 total edge intersections and 1926 inter-cluster edge intersections.

Figure 4.5: An example using a three level deep hierarchical clustering of a randomly generated graph containing 120 nodes and 282 edges. The hierarchy is indicated by the blue nodes and lines.

chy is present our approach can be applied in a bottom up fashion, initially applied to the leaf clusters and then to the higher level clusterings as shown in figure 4.5. We will utilise this optimisation for some of our hierarchically clustered graphs for our edge routing experiments in chapter 5 as well as our hierarchically laid out graphs in section 4.5.

4.3.3 Circular sifting

Our approach to ordering the nodes places them closer to their neighbours external to the cluster, reducing edge length, which implicitly reduces the likelihood of crossings. However approaches do exist which directly reduce edge crossings. Baur and Brandes' [BB05] circular sifting approach reduces edge crossings by rounds of swapping neighbour nodes in the circular layout, using edge crossing count as a heuristic. Such an approach could be altered to consider the crossing of inter-cluster edges, rather than internal edge crossings. However inter-cluster sifting of edges so a less finitely bound problem, as the reduction of crossings based on node order will be strongly dependent on the order of nodes in other clusters.

4.4 Layout and Hierarchy Generation

Inspired by multilevel approaches of Walshaw and others [Walo1, GK01, HK01, HJ05] we propose two approaches to graph layout, a hierarchical layout approach and a multilevel approach. We use our clustering approach to define the coarsest level of graph abstraction. We extend our flat agglomerative clustering into a multi-level clustering hierarchy, creating less coarse clustering abstractions based on the relationships between the cluster nodes and the clusters defined at the hierarchy level above. It is through the relationships of a cluster's nodes with other clusters that we maintain the concept of nodes of interest throughout the generated graph hierarchy. The clustering hierarchy also provides us with a means to route edges in the graph using hierarchical edge bundling [Holo6].

The hierarchical approach, while laying out clustered abstractions of the graph in decreasing levels of coarseness, is not a full multilevel approach as described by Walshaw. Our layout algorithms, both force directed and circular, only consider the nodes within the current clustering abstraction independently of all other abstractions being laid out. Interaction between nodes in different clusters at the same clustering level are prevented by strict cluster bounds and a lack of connectivity between clusters which do not have the same parent cluster node, and not by definition of node forces and an optimisation grid across each entire level as done for multilevel algorithms. While influenced by the approach of Walshaw *et al.*, particularly in terms of forces between cluster abstractions with a common parent, the approach could more accurately be described as a hierarchical layout than a multilevel one. The advantage of this approach over a multilevel approach is that hierarchy clusters will always remain close to their parent cluster, allowing for better reuse of the hierarchy as an aid to edge routing.

The multilevel approach utilises each level of the generated hierarchy as an abstraction of the level below it. Each level is positioned using a force directed layout, and the position of the cluster at that level are used to provide initial starting positions for the next level of the hierarchy. The forces applied to each level of the hierarchy are modified so that the lower level layouts do not disrupt the positions of the higher level layouts.

For both the hierarchical and multilevel approaches we utilise a circular layout of the leaf cluster nodes, to avoid the layout of the individual nodes of the original graph disrupting the previous levels of layout, and allow for the repositioning of nodes with inter-cluster edges.

4.4.1 Generating a clustering hierarchy

The initial top level of our hierarchical clustering is done by our agglomerative method described in chapter 3. This is the first stage where the user has specified their nodes of interest. We aim to replicate the relationship between nodes of interest through the multi-

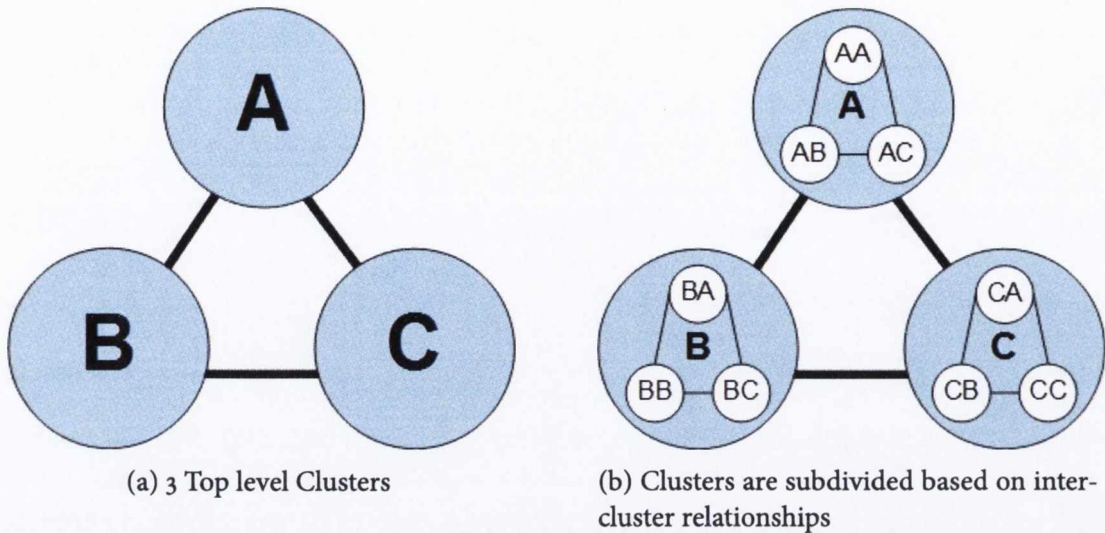


Figure 4.6: A simple example of hierarchical clustering, using 3 clusters. The relationship between cluster A b and C are reflected in the sub-clustering of each cluster.

ple levels of the clustering hierarchy. The number of nodes of interest defines the number of clusters at the top level. It also defines the maximum number of sub-clusters for each top level cluster. Each top level cluster is sub-divided based on which of the other top level clusters it's constituent nodes more conceptually belong to. As a simple example, consider a clustering of a graph around 3 nodes of interest as per figure 4.6. Each sub cluster only considers the level above it within the hierarchy. For example, cluster A will be subdivided into clusters as follows:

- Nodes which have no relationship with nodes outside of A, and can only belong to A are assigned to AA
- Nodes which have a stronger conceptual relationship with nodes from cluster B are assigned to AB
- Nodes which have a stronger conceptual relationship with nodes from cluster C are assigned to AC

Each cluster AA, AB and AC can be subdivided again recursively, based on the relationships between those sub-clusters. The number of sub-clusters in a subdivision depend on how many sibling clusters the cluster being subdivided has. To implement this approach to clustering, we need to be able to determine how strongly a node relates to the other peer clusters, in a manner similar to how we define our top level clustering.

We describe our hierarchy in terms of levels. L_0 is the clustering hierarchy tree root and L_1 is the top level of the clustering hierarchy, representing the clusters defined by our agglomerative clustering algorithm. L_2 contains all clusters subdivisions of L_1 . In multilevel

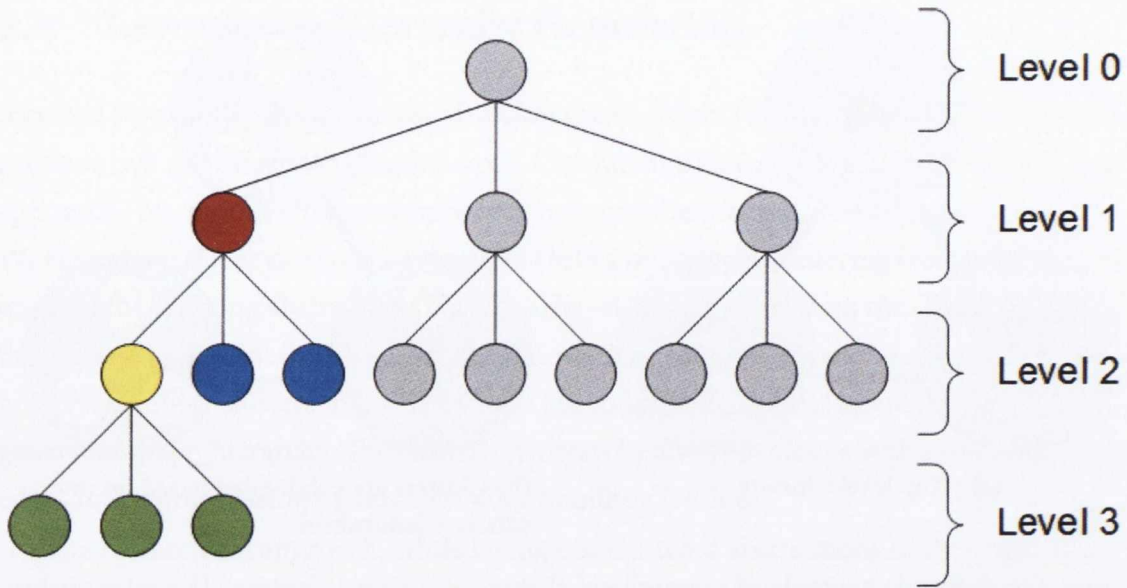


Figure 4.7: An illustration of node assignment to clusters at different levels. The L_3 green clusters consist of nodes from the L_2 yellow cluster and are assigned to each green cluster based on their relationships with the L_2 yellow nodes' two sibling clusters (coloured blue). The yellow and blue nodes are siblings because of their common parent, coloured red.

layout terms, L_1 can be considered a more coarse version of L_2 . For a given level L_i , its clusterings represent the relationships between a parent nodes siblings from L_{i-1} . This is illustrated in figure 4.7.

If the clustering hierarchy has N levels, it is not correct to say that level L_{n-1} contains the full set of graph nodes, as we are not building a balanced hierarchy. Clusters most likely have different sizes, resulting in some being further subdivided than others. The graph nodes are the leaf nodes of each branch of the clustering hierarchy tree. Repeated subdivision of clusters to sizes of one or two nodes, results in some very deep leaf nodes and is aesthetically not pleasing in terms of layout so we set a maximum cluster size in terms of leaf nodes. If a cluster in a level has less than the maximum number of leaf nodes it is not subdivided any further. In addition to the maximum cluster size we also utilise a maximum hierarchy depth to avoid the creation of deep leaf nodes.

Cluster Subdivision Function

For our top level clustering we were able to agglomerate nodes to form clusters. However for our subsequent clusterings we cannot simply re-run our top level approach at each level of the clustering hierarchy. One significant reason is that it would be unrealistic for users to specify sub-nodes for each subsequent level, and it would also be difficult for an agglomerative approach to reflect previous hierarchy structure. Therefore we are using a divisive or top down approach as described above. When a node from L_{i-1} is being assigned to a subcluster in L_i we assign a score for each subcluster to that node. The node is assigned

to the L_i subcluster that corresponds to the L_{i-1} sub cluster with the highest score. We use node inter-cluster degree as a metric to assign a cluster score to each node. We add the node to the sub-cluster representing the parent level cluster that it is most strongly connected to. In effect this is similar to using modularity, as we are assigning the node to the subcluster, corresponding to the parent cluster, that would have resulted in fewest inter-cluster edges.

4.4.2 Hierarchical Clustering Layout

Each level of our hierarchical clustering is laid out using a Force directed algorithm algorithm. We use a weighted version of Fruchterman and Reingold's basic algorithm [FR91]. For the layout of nodes within the leaf level clusters we use a circular layout. The circular layout is applied to each cluster initially. At this stage we are not concerned with the inter-cluster edges as the full hierarchy has not been laid out, so no optimisation is necessary. The initial circular layout provides us with size bounds for the leaf clusters, which we use in determining the spring weights for higher levels.

We apply the force directed layout algorithm on a per level basis in a bottom up approach. Given N levels of a hierarchy, we begin at L_{N-2} and apply the force directed layout algorithm to the child clusters of each cluster in that level. The nodes in each cluster in L_{N-1} have already been positioned as part of the circular layout. Once the layout of the level's clusters is complete, the position of the nodes contained in the cluster are updated to reflect the change of the clusters position. The resulting radius of the parent cluster is also updated. This process is repeated moving up the hierarchy until finally the clusters representing the initial agglomerative clustering (i.e. L_1) are positioned as the children of L_0 .

Our force directed layout of sibling clusters in each hierarchy level uses a weighted ideal distance between clusters. We use the cluster radius as a weight, which results in different clusters with different radii having a different ideal distance between them. The user can also specify a scaling value to be applied to the cluster weights during layout see figure 4.8. The sibling clusters being positioned as part of the force directed layout are only considered connected if there is an edge between the clusters at a lower level.

4.4.3 Multilevel Cluster Layout

We have also implemented a multilevel algorithm, which rather than laying out the nodes of a clustering level as separate sets of sibling nodes, performs a layout of the entire cluster level simultaneously. While based of Walshaw's approach, ours has some fundamental differences. Walshaw's algorithm uses a bottom up clustering, merging pairs of nodes. This ensures a balanced clustering. Our approach is top down and clusters are not even in terms of the underlying nodes they represent, nor is the hierarchy balanced in terms of depth.

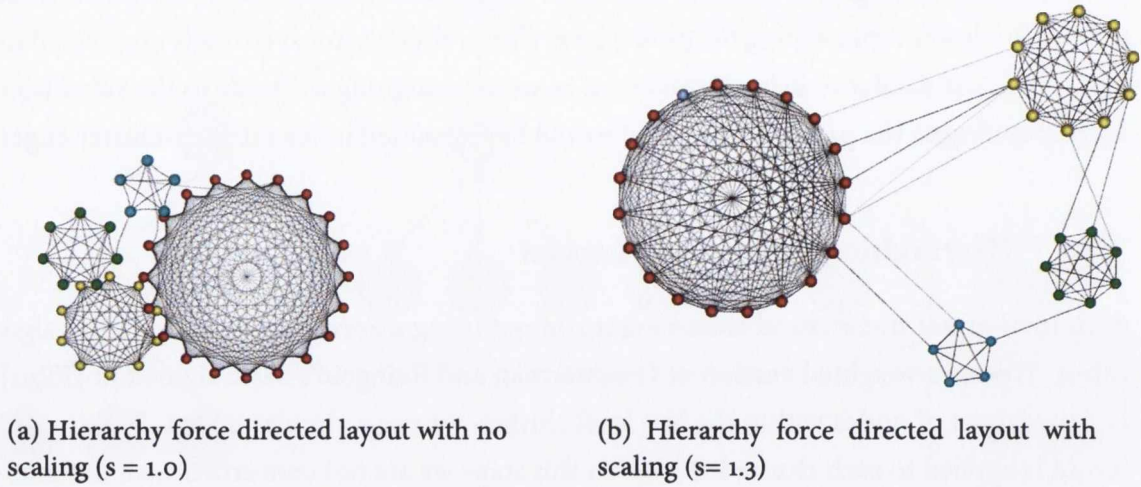


Figure 4.8: An example of the hierarchical force directed layout on the single level hierarchy with different sized clusters, illustrating the effect of scaling the forces.

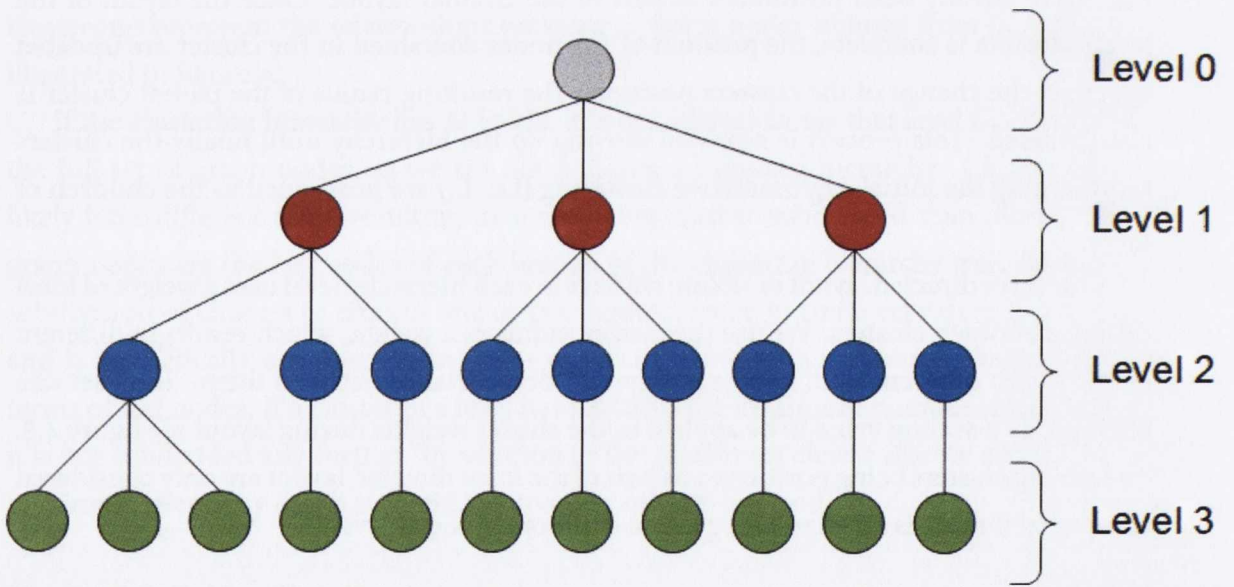


Figure 4.9: An illustration of the processing of different levels as part of the multilevel layout for a clustering hierarchy with $N = 4$ levels. The L_1 red nodes are positioned using a force directed layout. Next each of the blue clusters is assigned an initial position based on its parent red cluster. The blue clusters are positioned using a force directed layout, with different ideal spring lengths to the nodes of the previous level. Note that for the lowest level of this hierarchy L_3 , nodes are extended from L_2 if they have no children.

To ensure a proper layout of the lowest hierarchy level clusters on a N -deep hierarchy, it is necessary to extend the hierarchy leaf clusters which are not in level L_{N-1} through the hierarchy, so that all leaf level clusters are represented on level L_{N-1} , see figure 4.9. Leaf clusters are extended to lower levels by duplicating them then assigning the original cluster as the parent of the duplicate.

Default spring length: A key component of the force directed layout is the default spring length. This value, denoted k , represents the ideal distance between a pair of connected nodes, and conceptually represents a spring between two nodes at its rest length. For different levels in the multi-level layout different spring levels are used, this avoids lower level layouts disrupting the layout of higher level abstractions. Use of an optimisation grid, usually proportional in size to the spring forces, at each level also help avoid distant unconnected vertex pairs from disrupting the effects of higher level layout

Walshaw's approach uses a spring length at each level which is derived from the spring utilised at the level above, $k_L = \alpha k_{L-1}$, with $\alpha = \sqrt{4/7}$. However as part of his clustering algorithm, every node in level L_{i-1} represents the same amount of nodes in level L_i unlike our top down clustering. Clusters at any given level will represent different numbers of nodes in the main graph. Therefore we apply a weighted force directed layout at each level

Rather than adjust the k value for each level, we weight the spring distance between two cluster clusters in each level using the the number of graph nodes represented by each cluster. Given two clusters $C_1, C_2 \in L_i$ we weight the default spring distance between them as the square root of the sum of the squares of their node counts: $k_{C_1, C_2} = \sqrt{|C_1|^2 + |C_2|^2}$, where $|C_i|$ represents the number of nodes in cluster i , by the number of graph nodes represented by $|C_i|$. As the algorithm progresses through the clustering hierarchy levels, the number of nodes represented by each cluster gets smaller, as the clusters on level L_i will contain fewer, or possibly the same amount of nodes as level L_{i-1} .

We derive our layout temperature and square size for our optimisation grid, for each level, based on the maximum spring length k_{max} between two clusters at that level. We have found a values of $2k_{max}$ to be effective for grid size and temperature. We cease the round of force directed layout when the average displacement for the level reaches $0.01k$.

4.5 Results

We generated two small world graphs using the approach described in section 2.4.1 to demonstrate our layouts. These are a 100 node graph with 400 edges and a 400 node graph with 1600 edges. We show the resulting graphs under the each layout, as well as in edge bundled form. The nodes of interest for each graph were selected based on maximum node degree. For the 400 node graph, 5 nodes of interest were selected and a hierarchy depth was limited to 5 levels deep and a maximum cluster size of 30 nodes was used. For the 100 node

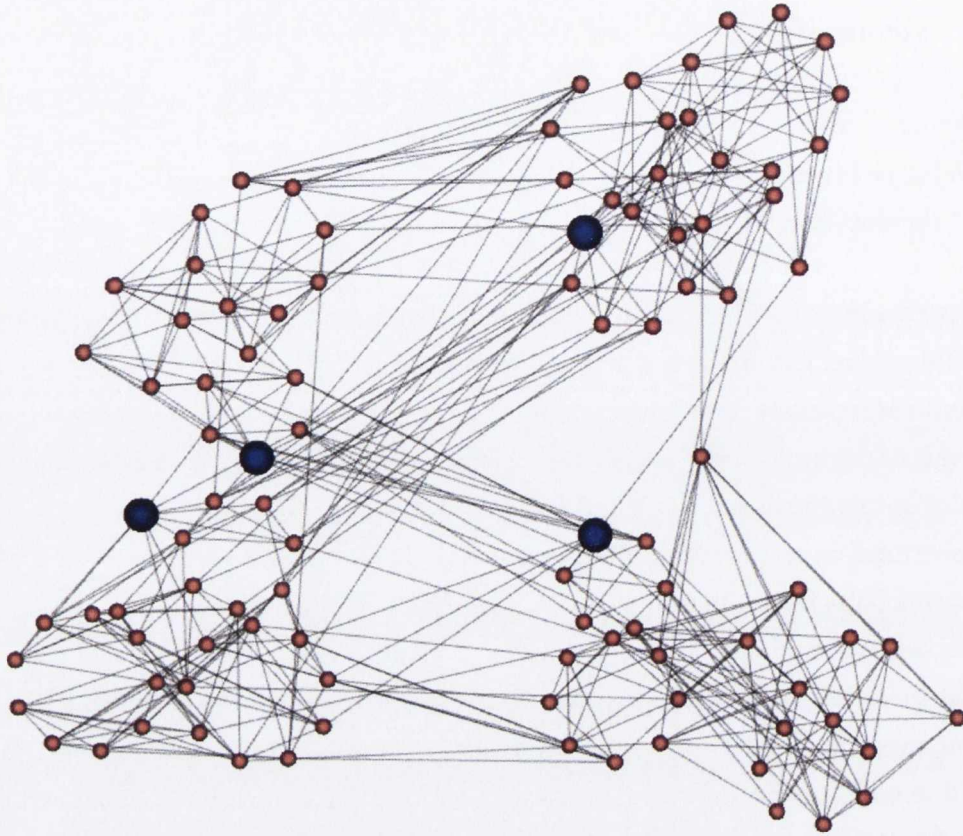


Figure 4.10: A layout of the 100-node small world graph clustered around 4 nodes of interest using the hierarchy approach, combined with our node ordering and rotation optimisation. The large blue nodes indicate the nodes of interest.

graph, 4 node of interest were selected and a maximum cluster size of 10 was used. A single level agglomerative clustering of the 100 node graph, using a per-cluster force directed layout, was shown in figure 4.1.

4.5.1 Hierarchical Layout

The results of using out hierarchical layout on each graph can be seen in figures 4.10 and 4.11. The hierarchical layout does not consider connections between non-sibling clusters at each level, therefore requires the use of our node ordering and rotation optimisation to ensure that unconnected clusters are not adjacent to each other in the layout. In both figures, it can be seen that clusters that are not in the top tier of the hierarchy are positioned quite closely together. This is because the inter-cluster connections between sibling clusters in the hierarchy are a reflection of the parent nodes relationship with its siblings. As a result there are many connections pulling the clusters together, and the forces cancel each other out only when the nodes are very close to each other. Using the scaling factor does not alleviate this issues as it scales both the attractive and repulsive forces for every cluster.

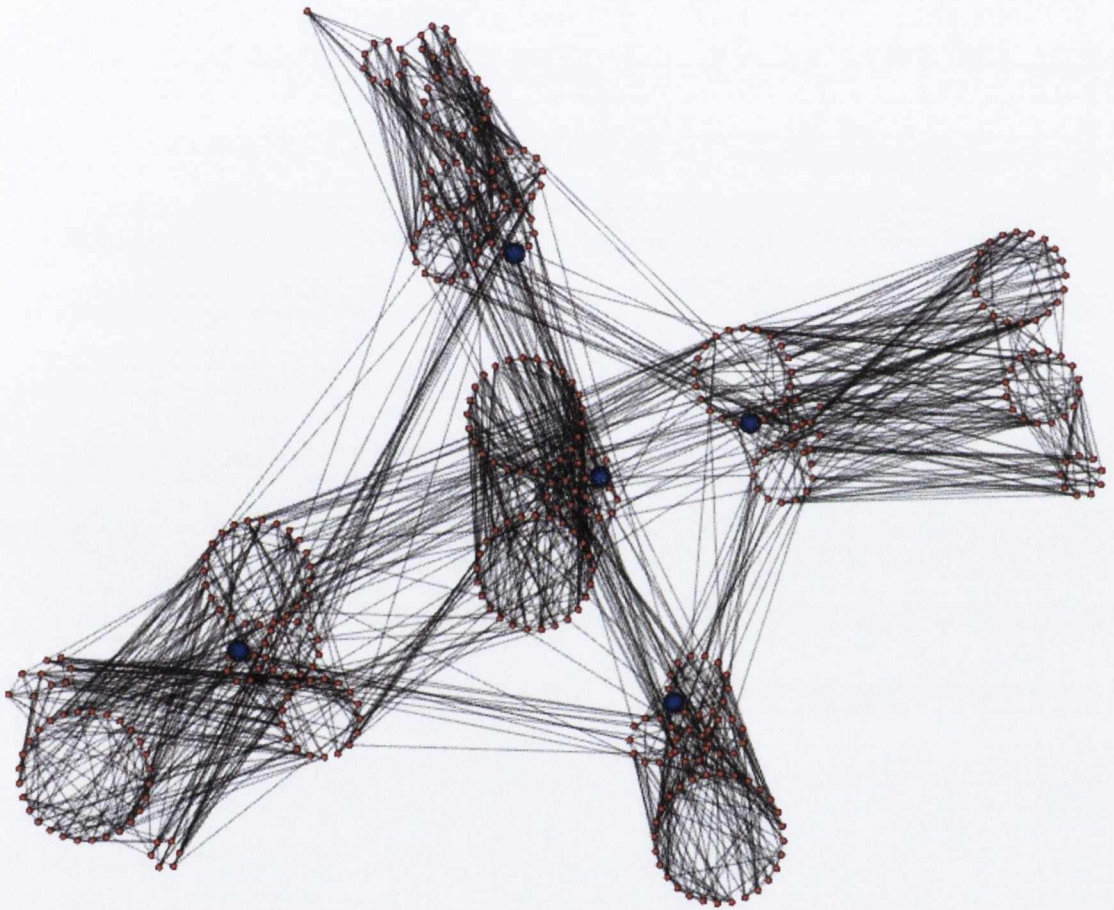


Figure 4.11: A layout of the 400 node small world graph clustered around 5 nodes of interest using the hierarchy approach, combined with our node ordering and rotation optimisation. The large blue nodes indicate the nodes of interest.

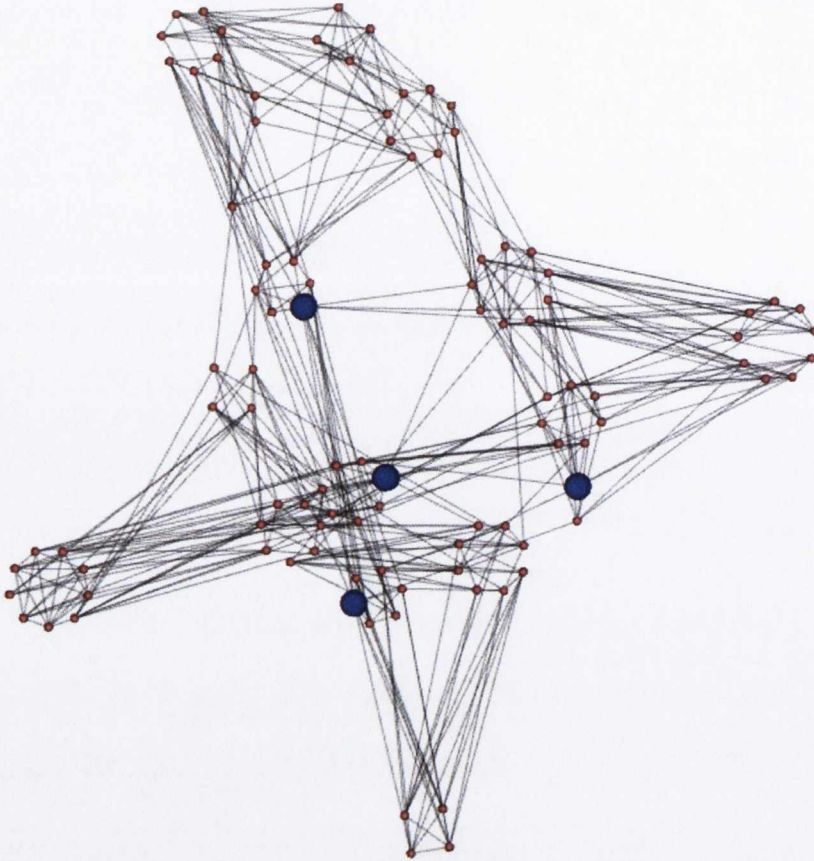


Figure 4.12: A layout of 100 node small world graph clustered around 4 nodes of interest using the multilevel approach. The large blue nodes indicate the nodes of interest.

4.5.2 Multilevel Layout

The multilevel layouts provide a clearer distinction between clusters at all levels of the hierarchy, as can be seen in figures 4.12 and 4.13. Due to the interaction of forces between all clusters in a layout level, clusters are well positioned relative to the clusters so there is no need for a hierarchical rotation to separate clusters in different branches of the hierarchy.

4.5.3 Hierarchical Edge Routing

Both approaches allowed the hierarchy to be used for edge routing, as can be seen in figure 4.14. However, the lack of spacing between clusters at lower levels for the hierarchical approach obscures the edge bundles, resulting in the multilevel layout being the better approach if edge bundling is utilised.

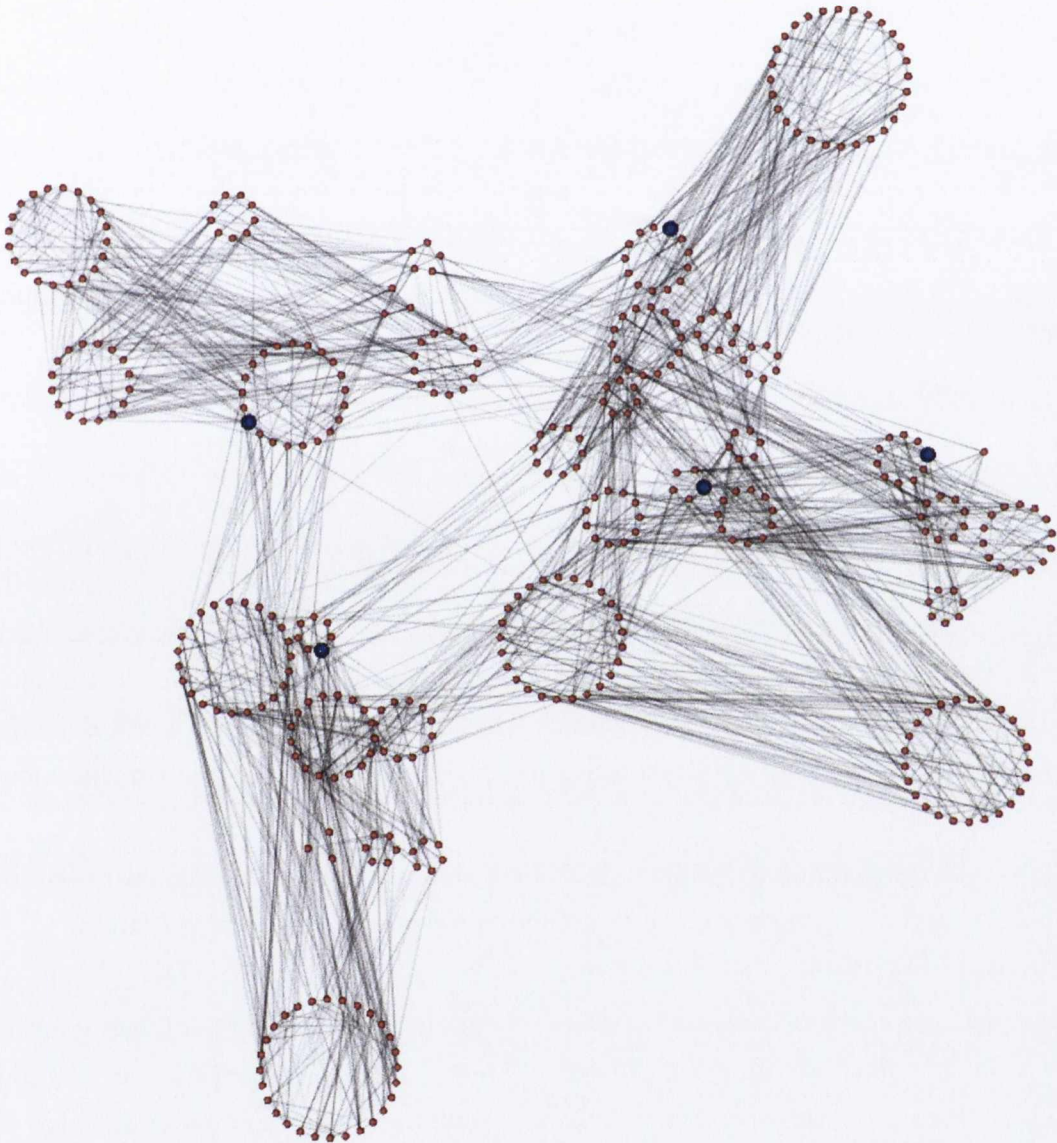


Figure 4.13: A layout of 400 node small world graph clustered around 5 nodes of interest using the multilevel approach. The large blue nodes indicate the nodes of interest.

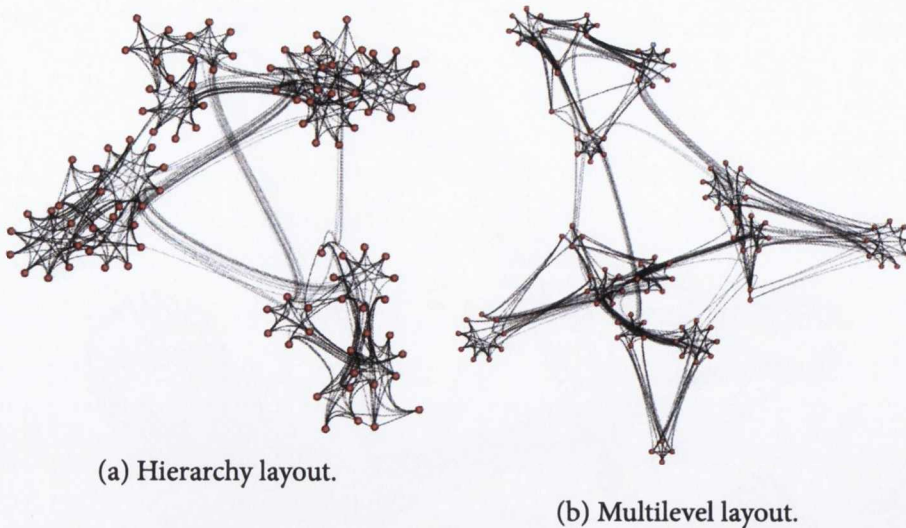


Figure 4.14: Hierarchically edge bundled versions of the 100 nodes small world graphs, using a bundling strength of $\beta = 0.9$.

4.6 Conclusions and Future Work

We have introduced two approaches to graph layout utilising our agglomerative clustering as an input. Our approaches both utilise a clustering hierarchy. We built the clustering hierarchy based on the relationships between nodes at the top level of the clustering to maintain the concept of node of interest throughout the multiple levels. We use this clustering hierarchy as the input for two approaches to graph layout. This hierarchy allows for multilevel layout and also provides a structure to allow the routing of edges using hierarchical edge bundling.

Of our two approaches to graph layout, the multilevel approach is the favoured one, due to the superior spacing of clusters at different levels of the clustering hierarchy.

As stated in the introduction to this chapter, laying out complex graphs is a difficult problem. The approaches described here are an initial step, and not a complete solution. We have illustrated the usefulness of generating a hierarchy to aid in graph layout and edge routing. However, further work is required on the weighting of graph nodes for force directed layout for both the hierarchical and multilevel approaches. The limitations of the generated clustering hierarchy need to be further examined, particularly in terms of usefulness of clustering beyond the initial cluster subdivisions. We also need to examine the impact of much larger scales of graph, as well as the practicality of other layouts (such as some form of force directed layout) for the nodes of leaf clusters. Another potential avenue of research is the usefulness of our layout approach as an initial starting point for graph analysis. When a user is analysing a graph, further manual editing of an existing layout may help them find answers to their questions about the data the graph represents. Research by Purchase *et al.* [PPP12] suggests that graph drawing systems should integrate

automatic layout with the user's manual editing process. Our approach to layout provides a view of graph structure that can be used as a starting point for graph analysis.

4.6.1 Conclusions and Future Work

The first part of this paper has presented a new method for the automatic generation of test cases for a program. The method is based on the use of a heuristic search algorithm to generate test cases that are likely to expose errors in the program. The method has been applied to a number of programs and has been found to be effective in generating test cases that expose errors in the programs.

The second part of this paper has presented a new method for the automatic generation of test cases for a program. The method is based on the use of a heuristic search algorithm to generate test cases that are likely to expose errors in the program.

The third part of this paper has presented a new method for the automatic generation of test cases for a program. The method is based on the use of a heuristic search algorithm to generate test cases that are likely to expose errors in the program. The method has been applied to a number of programs and has been found to be effective in generating test cases that expose errors in the programs.

Chapter 5

Edge Routing

IN CHAPTER 4 WE DESCRIBED OUR CLUSTERED HIERARCHICAL LAYOUT OF GRAPHS. However, this is only one aspect of the presentation of a graph to an end user. Often edges are the main source of clutter, and as a result they become the main cause of degradation at graph analysis tasks. As graphs become more dense, edges obscure each other and become difficult to trace, making it more difficult to see the relationships between nodes and spot high level trends in edge flow. Force directed layouts, as described in section 2.3 work exceptionally well for low density graphs, however as density increases force directed layout often results in overlapping edges being indiscernible. In order to reduce the clutter associated with graph density, the routing of the edges which cause much of the clutter needs to be considered.

In section 4.5 some of the graphs were displayed using edge bundling, an edge routing technique, described in section 2.5. While this approach has become very popular and has led to many variations [Holo6, CHH⁺08, ZYC⁺08, BD07, LBA10b, LBA10a, HW09, GK07, PNBH12, LLCM12, RDLC12], there is a lack of empirical evaluation of edge bundling, with much of the existing literature displaying images of the result of their technique, but not a formal user evaluation of it. To our knowledge this chapter is the first attempt at such an evaluation.

Previous work has also shown that user performance at a low level graph task is improved with three dimensional layout and stereoscopic display of a graph [WF96, WM05, SM93, HHL10]. We extended the standard edge bundling technique, so that the bundles utilise three dimensional depth, and render them stereoscopically. We have performed a second empirical user evaluation using a stereoscopic display to determine if the addition of stereoscopic depth improves user performance.

Chapter structure: This chapter is focused on two main concerns. The first is the evaluation of edge bundling in 2D. The second is the extension of edge bundling into 3D and an associated evaluation.

In section 5.1 we describe our evaluation of edge bundling in 2D. This section is further broken down as follows:

- In sections 5.1.1, 5.1.2, 5.1.3 and 5.1.4 we describe our evaluation motivation, previous experiment approaches, our hypotheses and our choice of edge bundling approach.
- In section 5.1.5 we describe our approach to generating graph suitable for our edge routing experiment, as well as how they were displayed to the end user.
- In section 5.1.6 we describe our experiment methodology.
- In section 5.1.7 we provide the results and analysis of our experiment.

In section 5.2 we describe our approach to extending edge bundling into 3D, as well as our motivation for utilising stereoscopic 3D. This is followed by an empirical evaluation in section 5.3. We describe our conclusions in detail in section 5.4.

5.1 Edge Bundling Evaluation

There is very little empirical data on the impact of effectiveness of edge bundling as a graph visualization technique, despite its popularity as a graph visualisation technique. Therefore, we experimentally evaluated the impact of edge bundling on user performance using graphs of different size and density, as well as with different levels of bundling on graph edges.

5.1.1 Evaluation Motivation

Purchase [Pur97] has demonstrated how the crossing of edges is the graph aesthetic which affects most human understanding of a graph. Unfortunately in large dense graphs, edge crossings are unavoidable, as once a graph has more than $(3|V| - 6)$ edges, where $|V|$ denotes the number of nodes, it is mathematically impossible to lay out the graph in a planar fashion such that no edges cross. While Weidong *et al.* [WSHEo8] have shown that the maximising of angles where edges cross also helps increase comprehensibility, bundled edges can frequently overlap and cross paths and intersect at acute angles. Bundling also introduces bends to edges, which have been demonstrated to have a significant effect on user errors and an approaching significant effect on user reaction times for low level relational tasks [Pur97]. Bundled edges also lack the good edge continuity identified by Ware *et al.* [WPCMo2] as an important graph aesthetic.

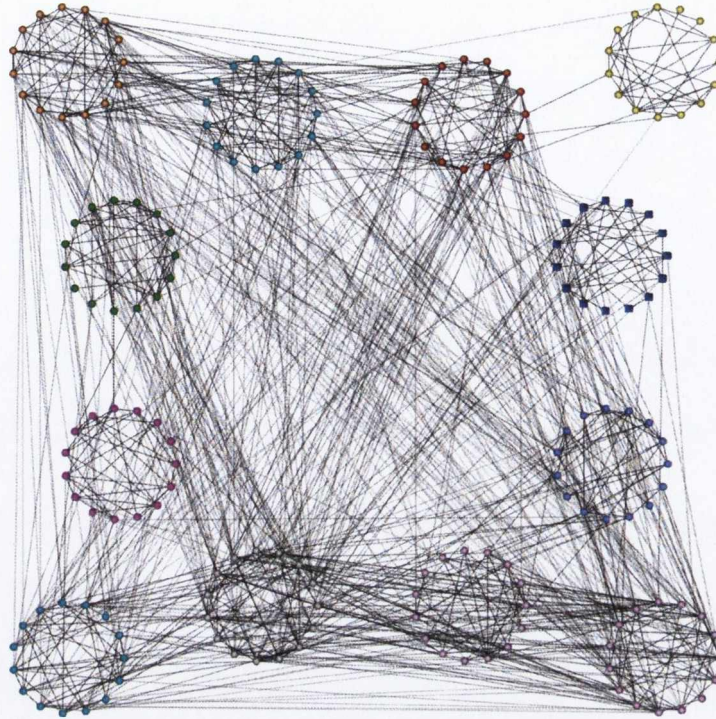


Figure 5.1: An example of one of the more dense graphs used in our experiments, $d_l = 5.26$.

Edges and Dense Graphs

In section 2.1.4 we discussed graph density and differentiated between graph theoretic edge density and linear density. While most real-world graphs have a value of $d_l \leq 10$ [Melo6], this is still enough to cause a large amount of clutter. Given the frequency that dense graphs are encountered in the real world, the fact that bundling is considered a clutter reduction technique and the fact that edges can be a major source of clutter in a graph, it is important to include edge density as part of our analysis. Therefore for our experiments we consider edge density as one of the experiment factors. One of the more dense graphs used in our experiments can be seen in figure 5.1, while a smaller lower density one can be seen in figure 5.2.

5.1.2 Previous Experimental Approaches

As described in section 2.4.3, empirical evaluations of graph visualisation techniques frequently use a simple low level relational task, such as path tracing or a similar variant, as a basis for the evaluation [Pur97, WPCMo2, WSHEo8, WMo8, HvWo9]. In some cases, such as Ridsen *et al.*'s work [RCMCoo], a high level task is used, such as determining where files should be placed in a directory structure. However, this type of high level evaluation usually only applies to a specific visualisation problem and does not necessarily generalise to a wider category of visualisation. While path tracing is a simple task, it a common one

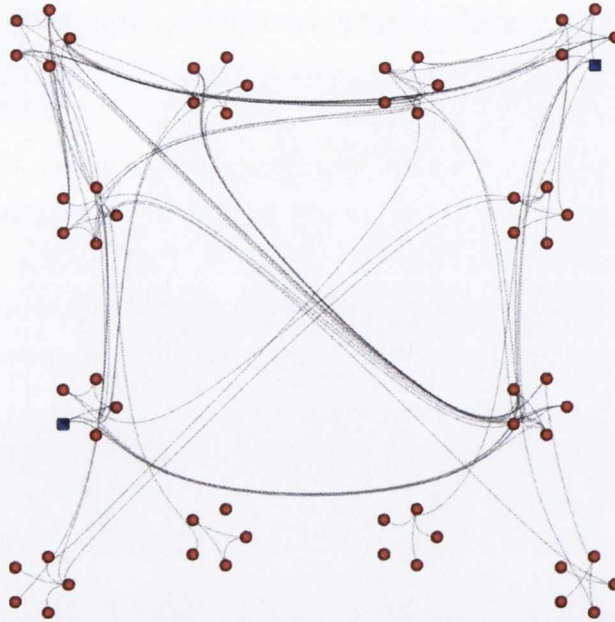


Figure 5.2: An example of a graph generated for our experiments, rendered with tightly bundled edges.

within node-link graph analysis. However it is also a very low level specific task and it is not the only way in which a graph can be read. For example, if a graph is clustered using a hierarchy, higher level trends of connectivity can be spotted from the links between clusters at different levels of the branches. Knowing that node A is connected to node C by a graph distance of two is useful, but also knowing that node A and node C are in two very strongly connected clusters may also be useful. These high level trends are important in graphs particularly when graph sizes and edge density become large, and the analysis for individual paths contributes less to the understanding of the graph as a whole. Other approaches than using path tracing tasks exist. For example Huang *et al.* [WSHEo8] investigate the impact of crossing angles using specifically designed diagrams containing edge crossings. Techniques such as eye-tracking and user surveys can also be used [HEHo8]. These other techniques often provide information on the how and why of a specific effect, however evaluation is concerned with the consequences of bundling, not the how and why.

5.1.3 Evaluation Hypotheses

Our primary hypotheses are that bundling improves user comprehension of low level connectivity tasks and also user comprehension of higher level trends. Our secondary hypotheses are that bundling improves user performance at both levels as graph size and edge density increase, relative to using straight line edges.

We have performed two user experiments to this end. The first examines user performance at a path tracing task to evaluate user comprehension of the connectivity between

individual nodes. The second examines user understanding of higher level inter-cluster connectivity trends, by asking the user to identify which cluster is most strongly connected to a highlighted cluster. We performed our experiments using compound graphs of various size and density, with different levels of edge bundling and laid out using a simple balloon tree layout.

5.1.4 Experiment Bundling Approach

We adopt Holten's [Holo6] hierarchical approach, described in section 2.5 to edge bundling for our experiments. This approach naturally lends itself to the hierarchies we generate as part of our multi-level layout described in chapter 4. This approach also does not necessitate any techniques to smooth the curves once they are drawn and allows us to generate test graphs which are consistent in bundling for different node sizes and different densities. The level of bundling can also be easily controlled as an input parameter to the experiment. The clusters of the graphs are also well defined as a result of the use of the clustering hierarchy, reducing ambiguity in what the user perceives as a cluster and what the experiment design defines as a cluster.

5.1.5 Experiment Graphs

We generated graphs for our experiments that reflect practical real world densities. However the focus of our experiments is on the impact of edge bundling, particularly in terms of how well users read bundled inter-cluster edges. Therefore it is the inter-cluster edge count that must scale between different edge densities, not just the general edge density. As mentioned by Lancichinetti *et al.* [LFR08], cluster size and edge distribution vary in real world graphs. We distribute our inter-cluster edges randomly between clusters, however the inter-cluster edge count is determined using a power law so there is a realistically wide range of inter-cluster edge counts distributed among the clusters. Adjusting the cluster sizes in a similar manner would result in some very large and very small clusters, providing extra information to the user when determining shortest paths or determining which clusters are most strongly connected. Therefore we distribute the nodes amongst the clusters evenly. Undirected compound graphs were used as the basis for the experiment. A clustering hierarchy 3 layers deep is used. The first layer contained 4 child nodes, the second layer contained 3 children for each of the 4 parent nodes. The actual graph nodes were assigned evenly to the 12 third tier hierarchy nodes. The number of nodes in the graph $|V|$ is always proportional to the number of leaf-level clusters (12) within the graph, resulting in an even number of nodes in each cluster. The graph area for each is directly proportional to the number of nodes in the graph to reduce overcrowding of the display space for the larger graphs. Each graph is displayed extending the full height of the screen. An example

of the resulting graphs generated can be seen in figures 5.2 and 5.1. For our test graphs we have small, medium and large graphs, using node counts of $|V| = 60, 120, 180$. Larger sizes than this resulted in the display becoming too crowded, and would have excessively impacted user performance, independently of bundling.

Choice of Graph Density The focus of our experiments is on the impact of edge bundling, particularly in terms of how well users read bundled inter-cluster edges. It is not appropriate to simply assign edges to nodes at random, as such an approach will not necessarily result in an even scaling of the number of inter-cluster edges along with graphs size and density. The number of inter-cluster edges is the difference between the total number of edges and the number of intra-cluster edges and hence is also related to cluster size. We denote the number of intra-cluster edges $|E_{int}|$ (internal) and the number of inter-cluster edges is denoted $|E_{ext}|$ (external). The number of intra-cluster edges is a result of the internal cluster density of the graph which we refer to as $d(E_{int})$. We refer to the inter-cluster density as the external density, $d(E_{ext})$. For path-tracing in particular, if the intra-cluster edges are so dense that the individual cluster edges form cliques, the participants will be able to infer connectivity without considering the internal edge structure of the cluster. The same can be said if the cluster's nodes form disjoint sets, with no edges between them. For the experiment to be valid the user will need to be forced to trace edges within a cluster. Towards this end, we fix the intra-cluster density such that each edge is connected to a consistent ratio of the other nodes within its cluster, setting the clusters to have an internal graph theoretic density $d(|E_{int}|) = 0.3$. Using the linear definition for density $d_l = |E|/|V|$ would result in intra-cluster edge count having a differing impact on the likelihood of an edge being traceable through a cluster depending on cluster size. This leaves the inter-cluster graph density as the means by which edge density will change between graphs. We can show that for our experiment graphs the only factor determining the maximum possible value is the node count, allowing us to use the ratio inter-cluster edges to the number of vertices as a measure of inter cluster edge density (i.e. $d_l(E_{ext}) = |E_{ext}|/|V|$). Given a graph with leaf level clusters forming a clustering $C = (c_0, c_1, \dots, c_{n-1})$ where $|C| = n$, a cluster i has its internal density defined by $d(E_{int})_{C_i} = |E(c_i)|/E(c_i)_{max}$ where $E(c_i)_{max}$ denotes the maximum number of edges possible in the cluster c_i . The maximum number of internal cluster edges in the graph is defined by

$$|E_{int}|_{max} = \sum_{x=0}^{x=|C|-1} \frac{|C_x|(|C_x| - 1)}{2}$$

The maximum possible number of intra-cluster edges is the difference between this and the maximum number of edges for the graph as a whole.

$$|E_{ext}|_{max} = \frac{|V|(|V| - 1)}{2} - \sum_{x=0}^{x=|C|-1} \frac{|c_x|(|c_x| - 1)}{2}$$

If we assume, as is the case for our experiment graphs, that each cluster is evenly sized, i.e. $|V| = \alpha|C|$, $\alpha \in I$ then $|c_x| = |V|/|C|$ and

$$|E_{ext}|_{max} = \frac{|V|^2(|C| - 1)}{2|C|}$$

Based on the graph theoretic density, we define the graph external density as $d(E_{ext}) = |E_{ext}|/|E_{ext}|_{max}$. From above, it is clear the maximum graph theoretic external edge density, $|E_{ext}|_{max}$ is proportional to the square of the number of vertices in the graph. If $|C|$ remains constant across all graphs, as it does for our experiments, it is the only variable that $|E_{ext}|_{max}$ depends on. Therefore we can state that for our experiment graphs $d_l(E_{ext}) = |E_{ext}|/|V|$.

For our path tracing experiment we used three different levels of density. For the cluster connectivity experiment we added a fourth level of density, which was not used for the path tracing experiment due to the difficulty of the path tracing in such a dense graph. For our experiments we wanted to choose values for $d_l(E_{ext})$ which visually corresponded to low, medium, high and very high densities. Through visual examination of the generated graphs we found that values of $d_l = 1, \sqrt{5}, \sqrt{10}$ and $\sqrt{20}$ produced an acceptable visual progression of graph density. A larger linear density would not be frequently encountered in real world situations with graphs of these sizes and would result in a very crowded presentation of the graphs. Linear edge density of less than 1 results in a low necessity for edge bundling for our test graphs. A table of our test graph sizes and the resulting density values can be seen in table 5.1.

Graph Layout and Display It is very difficult to rate one layout algorithm as being better than another in terms of simple relational tasks [Pur97]. We use a simple balloon tree layout, which is essentially a projection of a cone tree layout onto a 2D plane [CK95] and it can be seen in figures 5.2 and 5.1. This allows us to use a visually consistent layout for each of our test graphs. One impact of using this layout is that nodes which are close together in terms of graph distance will not necessarily be closely related in terms of geometric distance: two nodes which are neighbours may appear on opposite sides of the graph. If we were to use a force directed layout, related nodes would appear geometrically closer together. However the graph layout would be significantly different between graphs adding an even more significant confounding factor when trying to determine the impact of bundling on rendering edges. Our experiment graphs were rendered onto a 24" wide screen display,

$ V $	$ E $	$ c_i $	$ E_{int} $	$ E_{ext} $	$d_1(E_e)$	d_1	Task
60	96	5	36	60	1	1.6	1,2
60	170	5	36	134	$\sqrt{5}$	2.83	1,2
60	226	5	36	190	$\sqrt{10}$	3.77	1,2
60	304	5	36	268	$\sqrt{20}$	5.07	2
120	282	10	162	120	1	2.35	1,2
120	430	10	162	268	$\sqrt{5}$	3.58	1,2
120	541	10	162	379	$\sqrt{10}$	4.51	1,2
120	699	10	162	537	$\sqrt{20}$	5.83	2
180	558	15	378	180	1	3.1	1,2
180	780	15	378	402	$\sqrt{5}$	4.33	1,2
180	947	15	378	569	$\sqrt{10}$	5.26	1,2
180	1183	15	378	805	$\sqrt{20}$	6.57	2

Table 5.1: Experiment Graph properties, and which experiment they were used for.

with full screen anti-aliasing enabled. All of the graph renderings are static, the user was not able to manipulate the graph or alter their view of the graph (e.g. using pan or zoom functionality). Per Holten [Holo6], we also use alpha blending to allow individual edges to be more easily discerned within the bundles, by drawing shorter curves at a higher level of opacity than longer curves. Curves within the unbundled ($\beta = 0.0$) drawings are also blended based on length. Some of the previously described approaches use colour hue to indicate edge direction [Holo6, CHH⁺08] (as the underlying graph is a directed graph) or edge density [HW09, LBA10b]. We feel that this may add an extra confounding factor to the graphs, therefore all edges are shaded with no hue. The shortest edges are black, blended to grey with the white background for longer edges.

Bundling Strength The different levels of edge bundling offered by the implemented approach do not scale linearly in their visual impact (see figure 5.4). A bundling level of 0.25 offers relatively little difference when compared to a bundling level of 0.0 (i.e. straight lines with no bundling). We have chosen bundling levels of $\beta = 0.0, 0.7$ and 0.9 for the path tracing experiment, and the same levels with an additional level of $\beta = 1.0$ (the maximum tightness of bundles) for the cluster connectivity experiment. The $\beta = 1.0$ level of bundling is not used in the path tracing experiment as edges will frequently overlap as one line, making path tracing impossible in all cases.

5.1.6 Experiment Methodology

For each experiment the participant was shown a sequence of graphs on screen and was asked to perform a task specific to that experiment. To ensure the user comprehended the task and to reduce the impact of any training effect, the initial six graphs displayed were

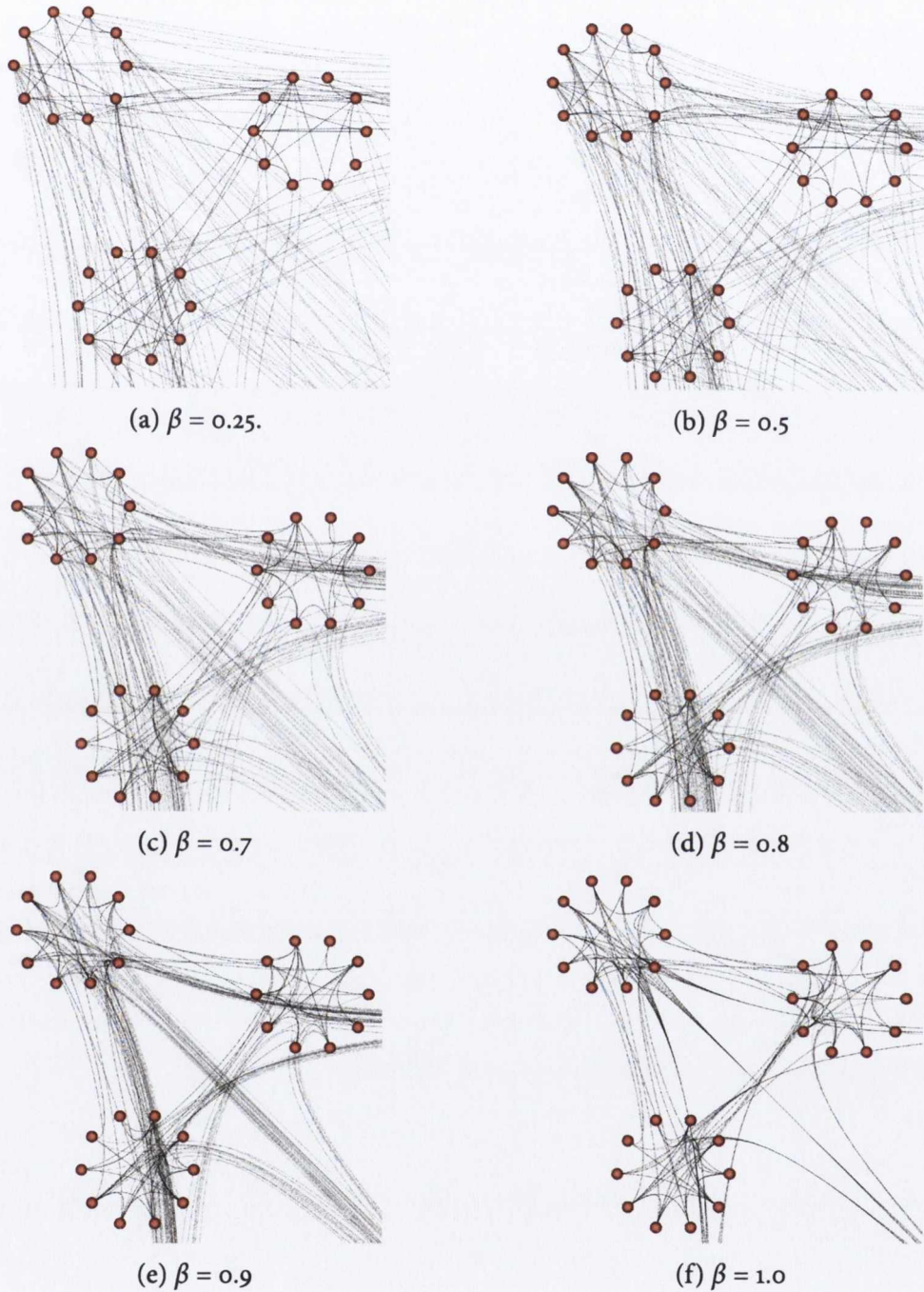


Figure 5.3: Illustration of the visual impact of different levels of bundling strength β . $\beta = 0.0$ is not shown as it simply corresponds to a straight line between nodes.

training graphs and the participant's answers for these graphs are not used in the analysis of the results. The actual experiment graphs displayed after the training graphs were shown in a random order. In order to ensure a rapid response from the participant and to avoid excessive experiment durations, each graph was only displayed for at most 20 seconds. The expiration of the time limit is considered a wrong answer. The participants were told to answer the question as accurately as possible, however if the answer is not clear, to answer with what they believe to be the most likely correct answer. The experiments were ordered so that half of the participants began with experiment 1, the other half with experiment 2. Prior to each experiment the user was given an information sheet describing the task for the graph, and was also verbally instructed on the task. All experiment participants, bar 2 from the total of 21, had a background in computer science and had some level of familiarity with the node-link display of graphs.

Path Tracing Experiment For the path tracing experiment we have four different factors that differentiate the task trials displayed to the user: 3 node counts (small, medium, large) \times 3 edge densities (low, medium, high) \times 3 bundling levels (0.0, 0.7, 0.9) \times 3 path lengths (1, 2, 3). The participant was shown a total of 81 trials. In each trial two nodes were highlighted and the user was asked to indicate the shortest path between the highlighted nodes, by pressing the corresponding key on the keyboard. The shortest path length between the highlighted nodes was always either 1, 2 or 3 and each path length was used once for all other combination of factors. The experiment graphs were displayed in a random order. The nodes were highlighted by being coloured blue and drawn with a square glyph, while the rest of the nodes were red and drawn with a circular glyph (see figure 5.2). This results in a pre-attentive effect, reducing the amount of effort the user needs to find the target nodes. The highlighted nodes were selected at random from the list of all node pairs that have the required shortest path distance between them. An additional constraint was placed so that the nodes are contained in separate clusters. This avoids simple cases where participants would only have to trace edges within a single cluster.

Cluster Connectivity Experiment For the cluster connectivity experiment the user was shown a total of 96 trials presented in random order: 3 node counts (small, medium, large) \times 4 edge densities (low, medium, high, very high) \times 4 bundling levels (0.0, 0.7, 0.9, 1.0) \times 2 repetitions (i.e. Each combination of factors was displayed twice). The average score and time for the two repetitions was used for the results analysis. In each trial, one cluster was highlighted at random and the user was required to left-click on the cluster that was most strongly connected to the highlighted one. The most strongly connected cluster is the one that shares the most edges with the highlighted cluster. The highlighted cluster was coloured blue and the nodes of the cluster were square instead of circular, as can be seen in figure 5.1. The rest of the clusters in the graph are coloured based on a random

selection from a list of colours, in order to make the clusters appear more distinct.

5.1.7 Results

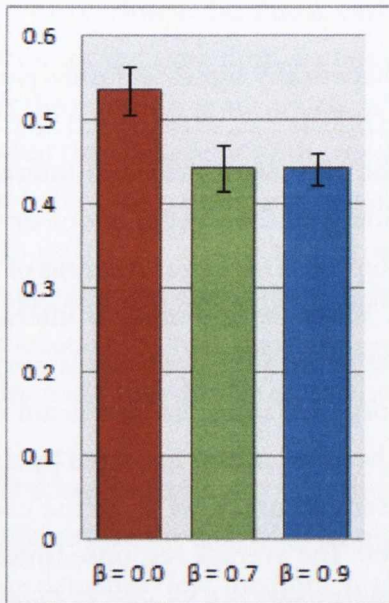
In order to evaluate our hypotheses, we must test for statistically significant differences in responses to the factors. We are interested in both Main Effects (i.e., when a particular variable or factor has an overall effect, independently of the other variables); and Interaction Effects (i.e., when the effect of a variable differs depending on the level(s) of one or more of the other variables). To test for such effects, we use Repeated Measures Analysis of Variance (ANOVA) on the data from all our experiments. When we find main or interaction effects, we explore what is causing these effects further using a Neuman-Keuls post-hoc test for pair-wise comparisons of means. We only report effects that are significant at the 95% level, i.e., where the probability that the difference between means occurred by chance is less than 5% (i.e., $p < 0.05$). When evaluating the user's accuracy, we score the user 1.0 for a correct answer and 0.0 for every incorrect answer. The average response times and the average accuracy across all participants for both experiments can be seen in figures 5.7 and 5.8.

Path Tracing Experiment

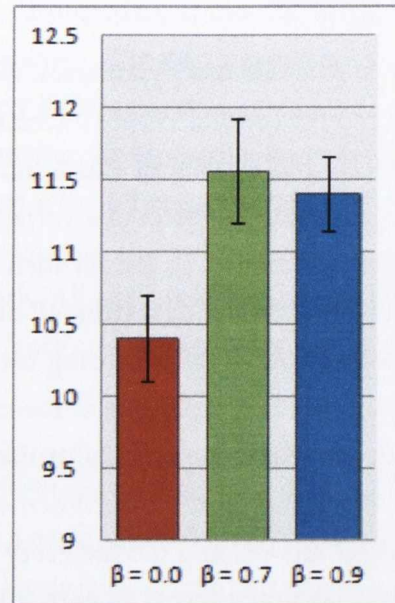
For this first experiment we performed a 4 way repeated measures ANOVA with within-subjects factors: node count \times edge density \times bundling level \times path length. We found that users performed significantly worse with both levels of bundling used for the path tracing experiment, compared to when the graphs were rendered using straight edges (see 5.4a). This clearly contradicts our primary hypothesis that bundling improves the participant's performance at tracing paths. Bundling hindered the participants performance at tracing graph paths. This is consistent with prior work which indicated that straight edges and path continuity improve the comprehensibility of graph [Pur97, WPCM02]. Bundling at either strength also caused a significant increase in the amount of time taken to answer. Table 5.2 sums up the noteworthy significant effects.

Our results also contradict our secondary hypotheses as edge bundling did not improve the participant's performance as the graph edge density and graph size increases. There is no significant interaction effect ($p > 0.05$) between bundling and node count in term of accuracy or in terms of time. Nor were there any significant interaction effect ($p > 0.05$) for edge density and bundling.

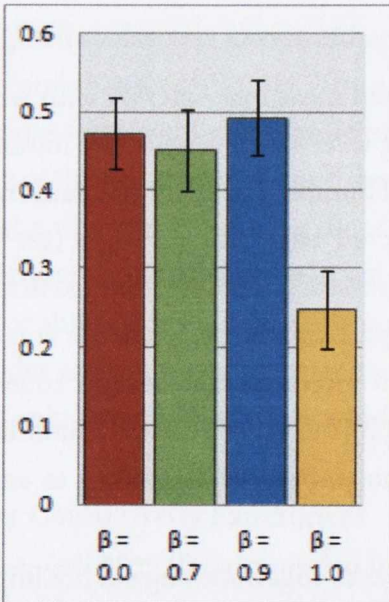
The level of bundling turned out to have a significant impact depending on the path length ($p = .0003$), in that the effectiveness of the tighter level of bundling improved over the longer paths. This may be due to the fact that 3 was the maximum path length available to the user for selection, so we cannot say that bundling helps with tracing longer paths.



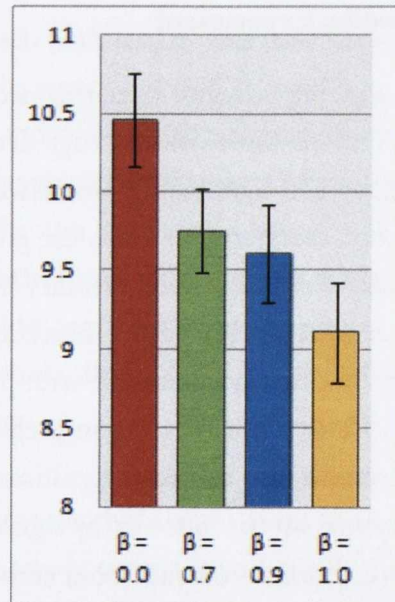
(a) Path Tracing Accuracy.



(b) Path Tracing Time.



(c) Cluster Connectivity Accuracy.



(d) Cluster Connectivity Time.

Figure 5.4: The impact of bundling on user accuracy and response time (in seconds) for each of the experiments. The vertical error bars denote +/- 1 standard deviation.

Measure	Effect	F-test	post-hoc
Accuracy	Bundling	F(2, 40) =9.8268, p=.00034	The two levels of bundling significantly degrade performance when compared to straight lines. The performance at each of the bundling levels was identical
Accuracy	Node Count	F(2, 40) =6.3469, p=.00404	The medium and high node counts cause a significant deterioration in performance when compared to the lowest. There was no significant deterioration between the medium and large graphs
Accuracy	Edge Density	F(2, 40) =2.8313, p=.07079	While there was no significant impact across all densities the Neuman-Keuls post-hoc analysis revealed that there was an approaching significant differences between the lowest edge density and the other two densities ($P = 0.06$ for the medium density and $p = 0.09$ for high density)
Accuracy	Bundling × path length	F(4, 80) =3.8084, p=.00694	For the 0.9 level of bundling , participants could significantly more easily identify the paths of length 2 and 3
Time	Bundling	F(2, 40) =11.641, p=.00010	Both levels of bundling cause a significant increase in time over straight edges, with no significant difference between the bundling levels
Time	Node Count	F(2, 40) =16.357, p=.00001	Increasing the number of nodes caused a significant increase in the amount of time taken, however the difference between the medium and large size graphs was not significant
Time	Edge Density	F(2, 40) =4.6931, p=.01476	Increasing the number of edges caused a significant increase in the amount of time taken, however the difference between the medium and high density was not significant

Table 5.2: Results for the four way repeated measures ANOVA for the user performance at the path tracing task.

Measure	Effect	F-test	post-hoc
Accuracy	Bundling	F(3, 60)=19.083, p=.00000	The only significant difference is using the strongest level of bundling $\beta = 1.0$, which significantly damages user performance, and the other 3 levels $p < 0.0002$)
Accuracy	Node Count	F(2, 40)=16.765, p=.00001	There is a significant difference between the largest node count and the other two.
Accuracy	Edge Density	F(3, 60)=27.781, p=.00000	There is a significant difference between all four levels of edge density except the medium and high levels.
Time	Bundling	F(3, 60)=11.478, p=.00000	Bundling significantly improves user response time when compared to straight lines. There is no significant difference between the $\beta = 0.7$ and $\beta = 0.9$ levels of bundling and the significant further improvement with the $\beta = 1.0$ level of bundling is irrelevant due to the significant degradation of performance
Time	Node Count x Bundling	F(6, 120)=2.7381, p=.01576	Significant for ($ V = 60$), where each of the bundling levels had significantly shorter response times than the straight line edges.

Table 5.3: Results for the ANOVA for the user accuracy and time taken at the cluster connectivity task.

Cluster Connectivity Experiment

For our second experiment we performed a 3 way repeated measures ANOVA with within-subjects factors: node count \times edge density \times bundling level.

The significant effects of this experiment are summarised in table 5.3. Results show there is a significant effect for bundling on participant accuracy. Newman-Keuls post-hoc analysis show that the only significance is between $\beta = 1.0$ and the other three bundling levels ($p < 0.0002$). From figure 5.4c it can be seen that the significant effect is a negative one, reducing user accuracy. In terms of response time bundling has a positive significant effect. Bundling at all levels $\beta > 0.0$ improves response time significantly ($p < 0.004$). The time improvement for $\beta = 1.0$ is irrelevant due to the reduced accuracy. However, as there is no significant difference between the other bundling levels and $\beta = 0.0$ we can say that, with the exception of $\beta = 1.0$, the cluster connectivity experiment's primary hypothesis that bundling will improve the participants performance at determining the relationship between clusters has been proven true.

Our results also show that our secondary hypotheses were that bundling ($\beta > 0.0$) will improve user performance as edge density and node count increase are false. The performance of each level of bundling under these conditions can be seen in figures 5.5 and 5.6. There was no significant interaction effect for bundling and node count in terms of

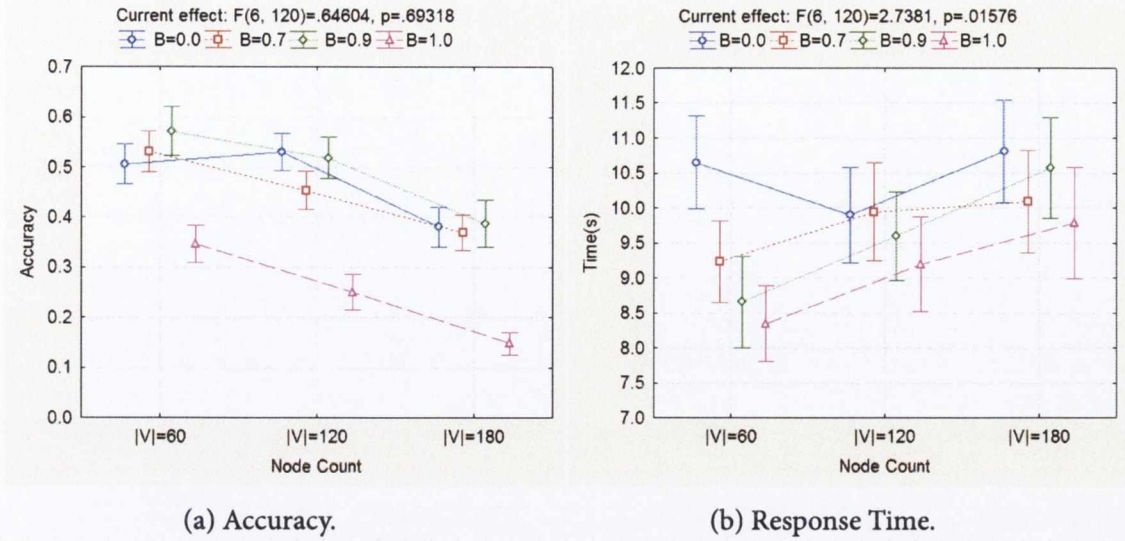
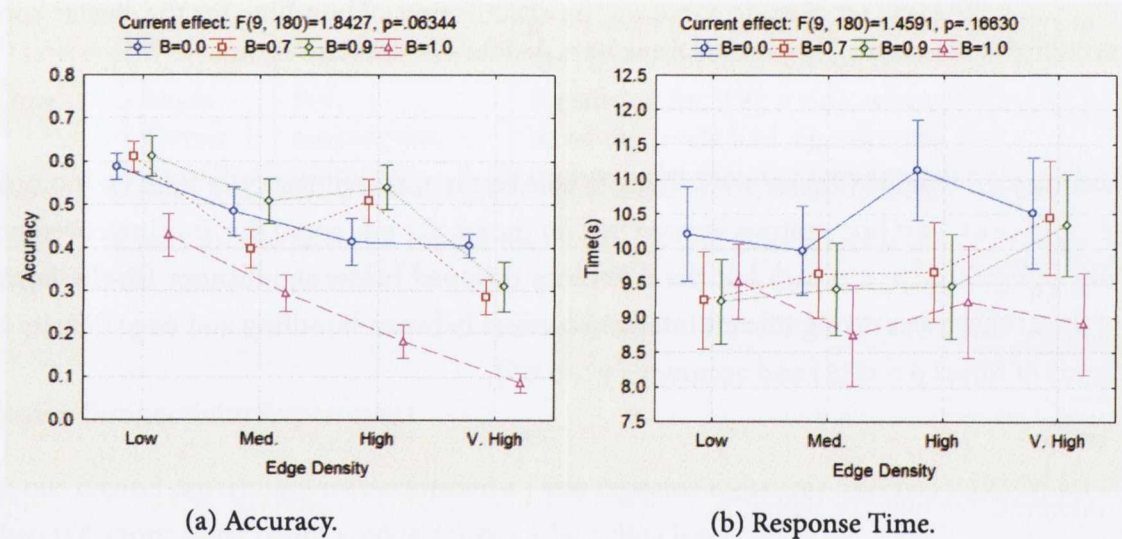


Figure 5.5: The impact of node count on the effectiveness of bundling for the cluster connectivity experiment. The vertical error bars denote +/- standard errors.

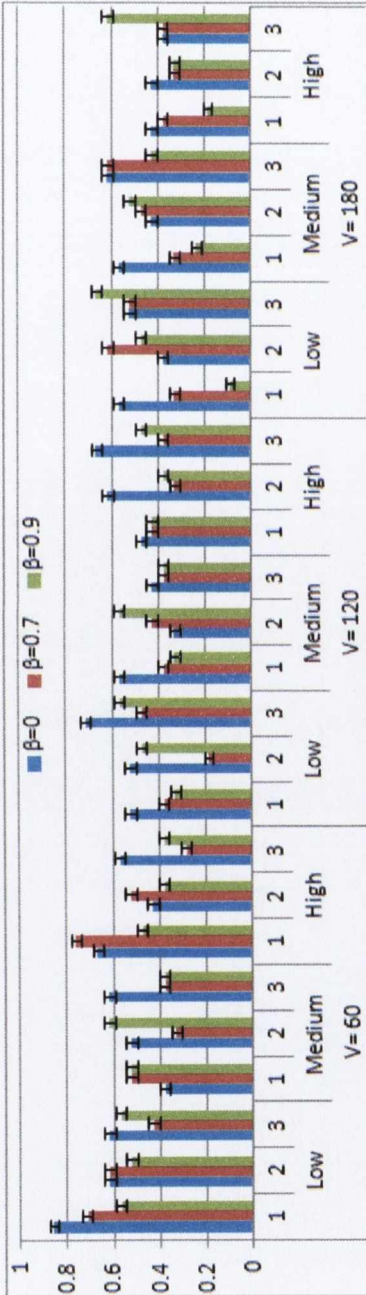
accuracy ($p > 0.69$) (Figure 5.5a). There is an effect in terms of response time ($p < 0.016$) (Figure 5.5b). Post hoc analysis showed that for the small node count the time improvement was significant ($p < 0.015$) but the difference dropped below significance for the larger graphs. There was no significant interaction effect between bundling and edge density in terms of time ($p > 0.16$) and accuracy ($p > 0.06$).



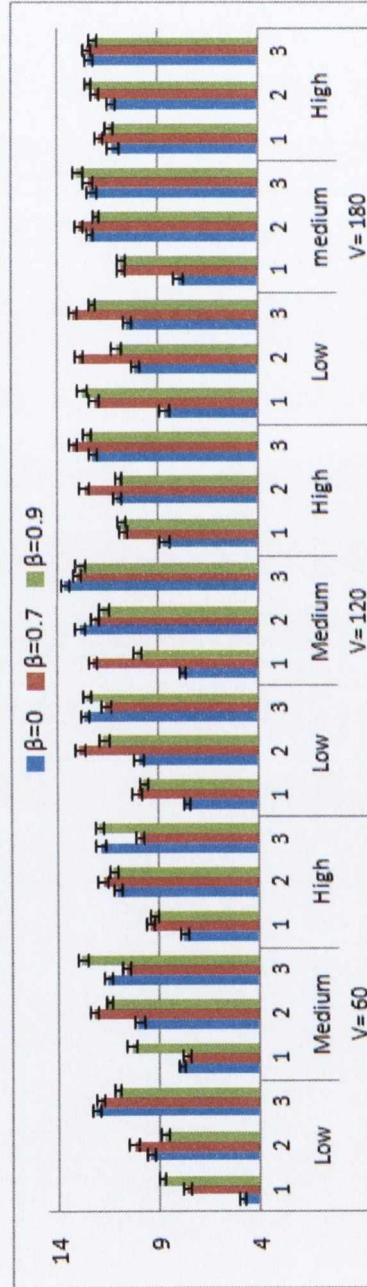
(a) Accuracy.

(b) Response Time.

Figure 5.6: The impact of edge density on the effectiveness of bundling for the cluster connectivity experiment. The vertical error bars denote +/- standard errors.

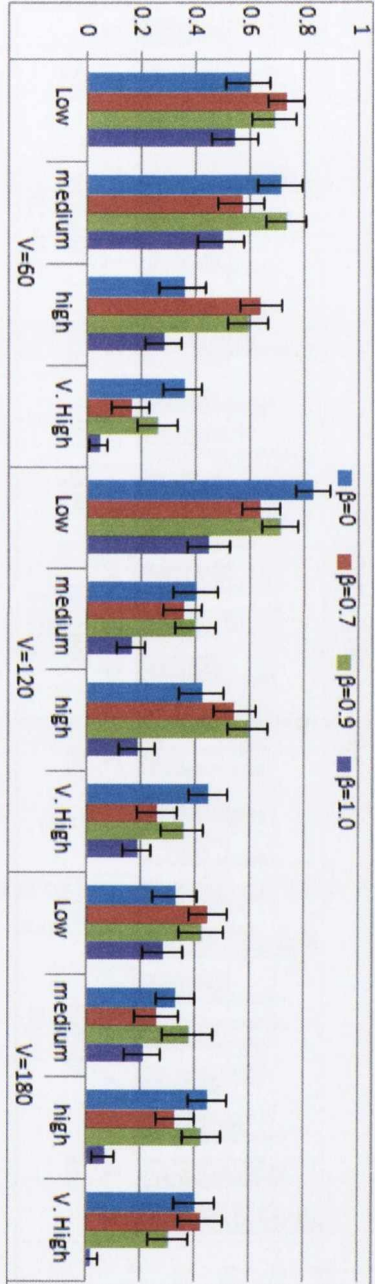


(a) Average user accuracy across all factors for the path tracing experiment.

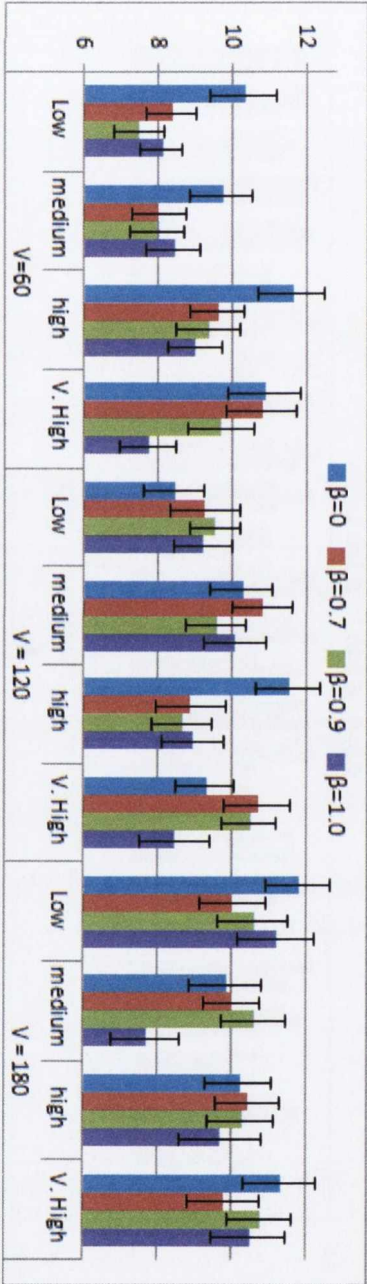


(b) Average user response time across all factors.

Figure 5.7: Overall Results for the path tracing experiment. The error bars indicate one standard deviation.



(a) Average user accuracy across all factors.



(b) Average user response time across all factors.

Figure 5.8: Overall Results for the Cluster Connectivity experiment. The error bars indicate one standard deviation.

5.2 Stereoscopic Three Dimensional Edge Bundling

We have seen in the previous section that, while edge bundling improved participant response times when it comes to recognising high level trends, it does negatively impact performances when it comes to a low level connectivity task such as path-tracing. This section is concerned with improving on the limitations of edge bundling by combining it with another technique which has shown to improve user performance at a low level path tracing task, stereoscopic rendering of a graph [WF96, WM08, SM93, HHL10]. We combine edge bundling with stereoscopic rendering by adding depth to the bundles, behind the graph plane. As part of this we determine how to apply depth to the bundles in a consistent manner.

Our results show that unfortunately stereoscopic depth is a complex technique, which is impacted by many other factors such as occlusion and graph layout, and the extension of only edges into 3D does not offer a significant benefit. However, we describe in detail our approach and experimental results in order to aid any future work on edge bundling and stereoscopic 3D.

5.2.1 Motivation

Previous work on edge routing has focused mostly on routing edges in two dimensions [Holo6, ZYC⁺08, CHH⁺08]. There is no existing evaluation or demonstration of the effect of edge bundling in three dimensions. In their work on "3D Edge Bundling for Geographical Data Visualization" Lambert *et al.* [LBA10a] put forward a three dimensional routing of edge bundles in a geographical visualisation, however this is simply a case of routing bundles around a globe. Rather than using one dimensional lines, the authors use bump mapped three dimensional tubes, as can be seen in figure 5.9. For graphs laid out in three di-



Figure 5.9: An example of the three dimensional bundles created by Lambert *et al.* [LBA10a]

mensions Ware and Mitchell [WM05] have shown that tubes actually significantly reduce user performance at path tracing tasks when compared to lines. It is worth nothing that Lambert *et al.*'s agglomeration of edges does not allow for path tracing and the tube size reflected the number of edges represented, so the choice of cylinders for edge representation is not based on edge traceability. In their initial analysis of the impact of stereo rendering on graph comprehensibility Ware and Franck [WF96] used a random layout, in the later work of Ware and Mitchell [WM05] a spring layout algorithm was used. The

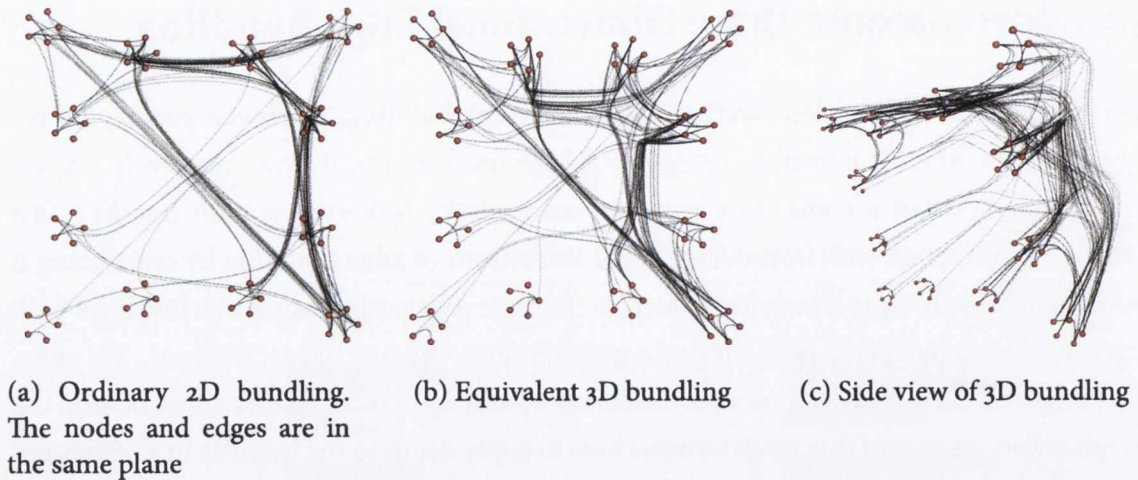


Figure 5.10: Illustration of extension of bundling into 3D.

layout algorithm was selected to allow realistically laid out graphs to be used in evaluation experiments, rather than provide any specific aid to graph visualisation.

When bundling edges in a two dimensional graph display, tighter edge bundles reduce the clutter on-screen, however they also make the graph less comprehensible. We propose that by extending the edges into three dimensions, stereoscopic viewing can allow users to more clearly see individual edges within the bundle.

5.2.2 Edge Routing in three Dimensions with stereoscopic viewing

Our approach to using stereoscopic-viewing does not rely on three-dimension positioning of nodes. All of the nodes are laid out in a two dimension plane, resulting in no obstruction of nodes when viewed from a perpendicular angle. Our aim is to improve the users understanding of graphs by extending the edges into 3D. From a two dimensional parallel projection of the graph, the bundles will be unchanged, we simply propose to alter the depth of the bundles edges perpendicular the to the graph plane. This means that under a perspective projection the bundles splines will appear slightly different (depending on the projection parameters) and under the stereoscopic viewing conditions we suggest that the individual splines will be more distinct to the user. This is not a pure layout of a graph in three dimensions, but rather the use of three dimensional space and stereoscopy to attempt to reduce ambiguity in edge bundling. We propose that this routing of edges combined with stereoscopic viewing will improve user performance at graph tasks.

Adding Depth to Curves

For most two-dimensional graphs an edge is displayed as a one-dimensional line. It may be tempting to render edges as three dimensional tubes when drawing graphs in three-

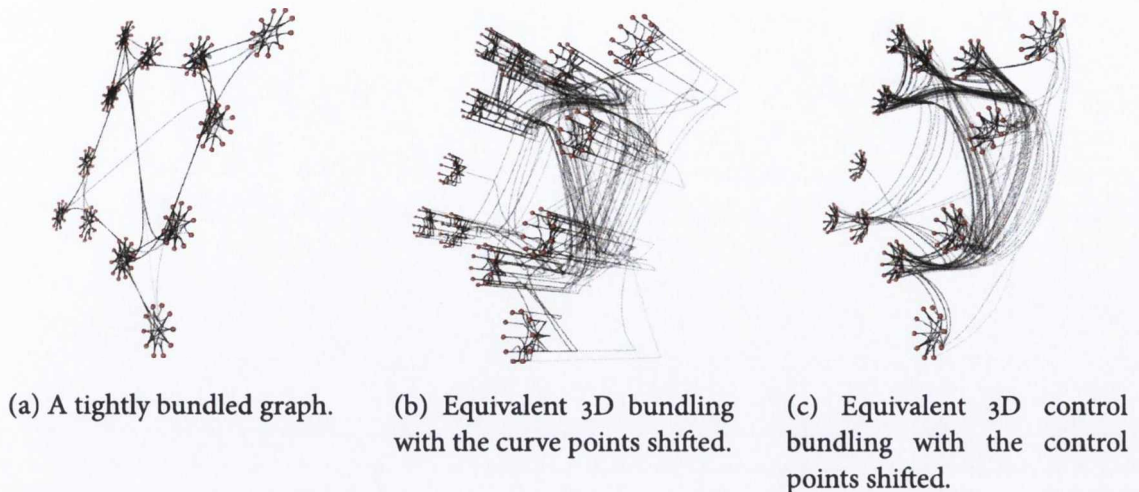


Figure 5.11: An illustration of the difference between shifting edge curve points and control points. The same graph is viewed from a side view to allow the bundle depth to be seen more clearly.

dimensional space, however previous work by Ware and Mitchell [WM05] has shown that using tubes produced a higher error rate at a path tracing task, when rendered in stereoscopic 3D.

In order to move the edge curves into three dimensions the values of the line segments of the curve can be shifted perpendicular to the node plane, or the values of the control points of curve can be shifted. If the curve points are simply shifted by a function to determine depth we get the results we see in figure 5.11b. The shift in depth is quite dramatic, and equal for every point on the curve, except for the first and last (which lie in the original graph plane). To give the depth a more gradual change, the depth could also be controlled as the result of a curve depth function. The most straight-forward way to accomplish this is to shift the control points that are used to build the curve. The curves points interpolate along the b-spline between the control points, producing a smooth curve as can be seen in figure 5.11c.

The shape of the resulting curve, perpendicular to the graph plane, depends on the number of control points in the curve, and how far each point is pushed back. If we simply push back the control points to the desired depth, we end up with a quite square shaped curve as can be seen in figure 5.12a, particularly for the longer edges. A smoother, less square curve is obtained by linearly shifting back the control points of the curve based on their position in the control point vector. The midpoint of the control point vector is pushed back the full distance and and control point halfway between the start or end point and the midpoint would be pushed back half this distance. In the case where there is no midpoint, due to an even number of control points, one is introduced as an average of the existing two middle control points.



(a) Push back control points to a desired depth.

(b) Interpolate control points to a specific depth and insert an additional control point where required.

Figure 5.12: Different types of control point shift, each using hierarchy depth as a depth function.

5.2.3 Defining Curve Depth

In addition to the question of how we extend the curves into three dimensions, we need to determine by what depth a curve should be shifted to. The notion of depth here describes the maximum distance of the curve perpendicular to the graph plane, and parallel to the view direction. The most straightforward option is to make depth a function of the curve length. We also looked at making depth a function of the number of control points, used by the curve, as well as using a more discrete depth function. This allows a number of depth planes to be specified and the curve control points snap to the closest depth to one of the planes. For example if the curves have a maximum length of 20 units and there are 5 plains specified, curves would intersect planes at 5, 10, 15, 20 units behind the graph plane. Each of these approaches can be modified by a multiplier value, which can be used to scale them to similar depth ranges.

As part of our bundling, we still use the splines straightening approach taken by Holten. This straightening takes place before the curves are shifted into 3D. This is desirable as we assume the view direction is perpendicular to the graph plane. The straightening process will then straightening the curves perpendicular to the viewing direction, which makes the individual splines easier to see for a users perspective.

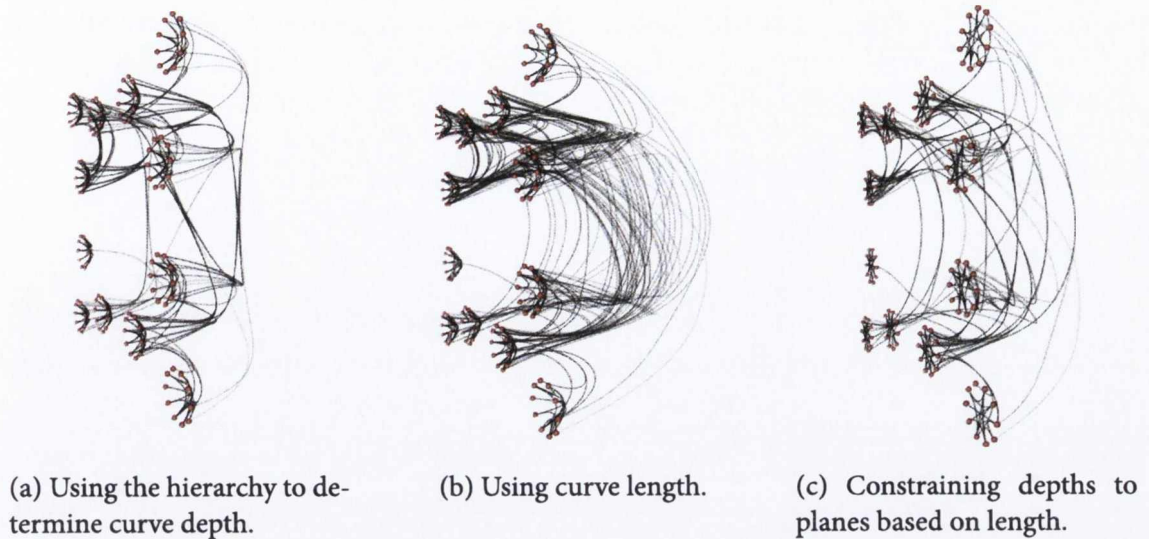


Figure 5.13: An illustration of the difference between the depth functions.

5.3 Three Dimensional Bundling Experimental Evaluation

The goal of our experimental evaluation is to determine whether it is possible to improve user performance at graph tasks with bundled graphs by extending the edge routing of those graphs into three dimensions.

For our stereoscopic experiments we reuse the methodology of our previous edge bundling experiments with some changes. We are re-using the graphs from the experiment described in section 5.1, in order to have the same structure of hierarchical graph as before. We adopt the same experiment tasks as well as experimental setup and random ordering of graphs as seen in our previous user experiment. The key differences in our experiments compared to previous stereoscopic graph experiments [WM08, vSvDZS⁺10, SM93, HHL10], are as follows:

- Our nodes lie in a two dimensional plane, only edges use depth behind the graph plane. We are focusing purely on the effect of extending edges into 3D. The position of nodes in 3D would provide an extra confounding factor.
- The graph data is clustered.
- The graphs edges are bundled.

5.3.1 Hypothesis

Our previous experiment has show that edge bundling has a negative impact on user performance at path tracing tasks. Our hypothesis is that adding depth information to the edge bundles in a direction perpendicular to the graph plane will improve user perfor-

mance when viewing the graph in stereoscopic 3D. We also hypothesise that there will also be benefits for user performance seen for a higher level cluster connectivity graph task.

5.3.2 Choice of graphs and experiment factors

Factors from the Previous Experiment

In our previous experiment we used three different sizes of graph and while there a main effect resulting from node count there was no interaction between node count and bundling (see the results in section 5.1.7). Therefore, for our stereoscopic three dimensional experiments we use a single size of graph, the 120 node graph. The 60 node graph offers a substantial boost to effectiveness and results in smaller bundles (due to the reduced maximum edge density of the graph), which reduces the necessity for the addition of depth to distinguish edges. There was no significant different between the 120 and 180 node graph sizes, so we omit the larger size from this experiment. In terms of graph density we also reduce the number of density levels. However we use two levels as Ware and Mitchell[WMo8] have shown that the relationship between edge count and node count has an significant impact on error rates for path tracing tasks in 3D. In each of our prior experiments there were no significant consistent differences between the $\beta = 0.7$ and $\beta = 0.9$ levels. Therefore for this set of experiments we use a single value of $\beta = 0.8$ in place of the original two. Intra-cluster edges (those beginning and ending in the same cluster) are not extended into 3D.

For the path-tracing experiment the path lengths, as before, are limited to one two or three hops. Once again were highlighted the randomly selected node pairs, which cannot consist of nodes from the same cluster. A summary of all factors can be seen in table 5.5.

Layout Optimisation

In the previous experiment nodes were positioned in cluster without any concern for their relationships with nodes in other clusters. This can raise ambiguity in the flow of edges, particularly if an edge needs to pass though the cluster of one of its terminating nodes as can be seen in figure 5.14a. We optimised the graph layouts using the method described previously (see section 4.3) and provided the users with optimised and unoptimised versions of each graph. The main purpose of optimising the layout is to reduce the number of edge crossings, which is an aesthetic which has been shown to have a significant effect on user comprehension of graphs [Pur97]. Table 5.4 shows the number of edge crossings in each graph before and after optimisations.

Depth Type

For our experiment graphs we use three different setting of depth These are no depth, hierarchy depth and edge length depth. The hierarchy and length depths are calculated as

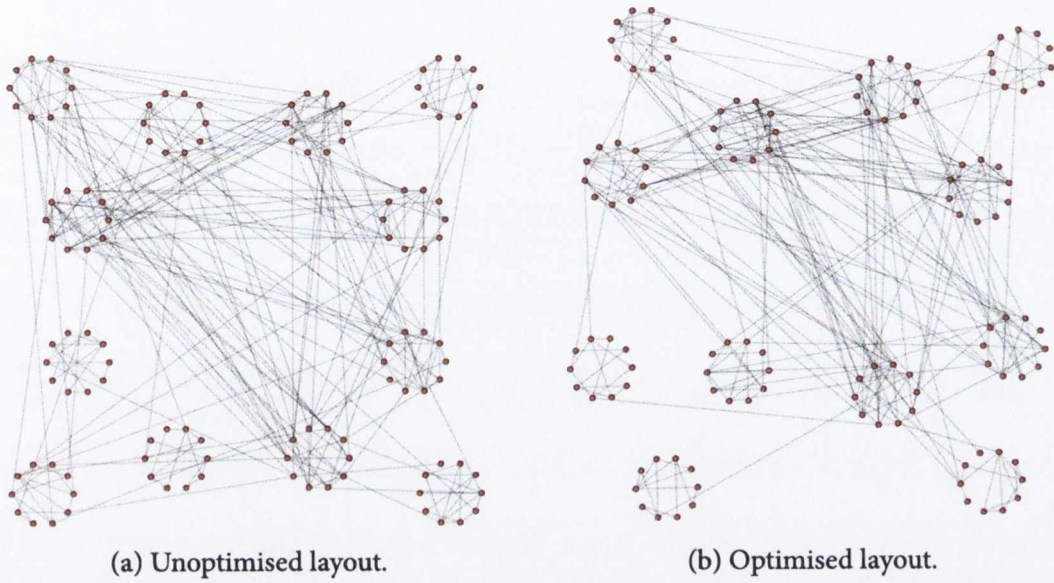


Figure 5.14: Illustration of the visual impact of rotating clusters and reordering nodes in clusters $\beta = 0.0$

Density Level	Edge Count	Edge Crossings (Unoptimised)	Edge Crossings (Optimised)
low	282	7,917	2,631
Medium	430	21,049	10,128

Table 5.4: Edge Crossing for each layout type

Factor	Count	Values	Experiment
Layout	2	Un-optimised, Optimised	Both
Depth	3	Hierarchy, length , None	Both
Path Length	3	1,2,3	Path Tracing Only
Bundling	2	$\beta = 0.0$, $\beta = 0.8$	Both
Density	2	Medium ($ E = 430$) , low ($ E = 282$)	Both

Table 5.5: A summary of the various factors considered for the ANOVA analysis of the experiment data.

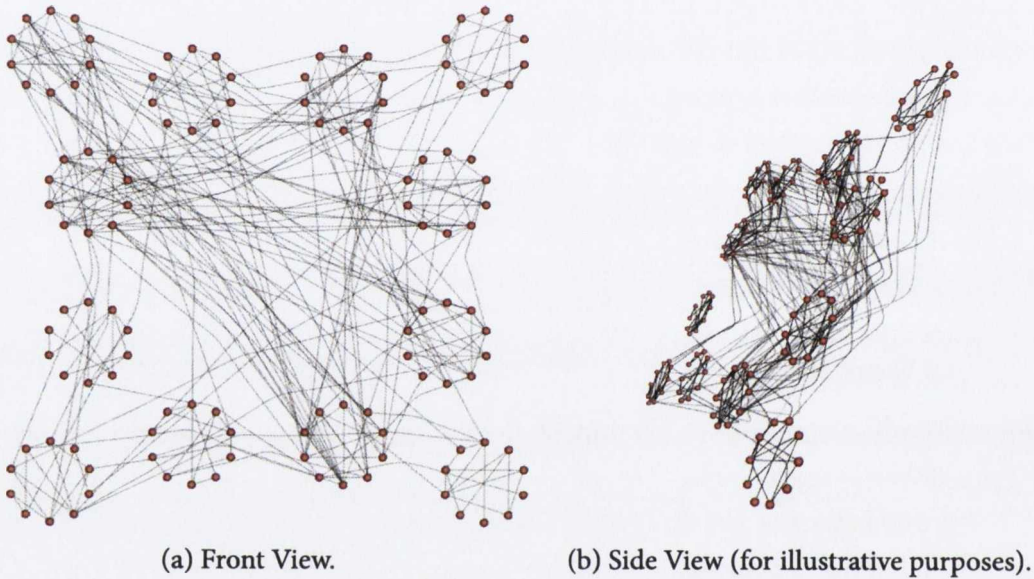


Figure 5.15: Illustration of the impact of adding depth to straight line edges $\beta = 0.0$

described previously in section 5.2.2. For the no depth option the graph is still rendered stereoscopically, but all edges lie in the same plane as the nodes. Regardless of the edge length, the depths are scaled so that the furthest control point is 50 units behind the graph plane. Therefore the range of depths is the same for both approaches, it is the distribution of depths in this range that changes. For the straight edges depth is applied in the exact same manner as it is for the bundled edges. To allow this to happen the straight edges are not simply end points, drawn with a straight line in between. They are B-splines where the control points are set such that the curve is actually a straight line and their depth can be modified just like the bundled edges. This results in a v shaped line as can be seen in figure 5.15

Graph Colouring

In our previous experiment, graphs were drawn on a white background, with edges shaded from black to grey based on length. Initial tests with the 3D display showed that the

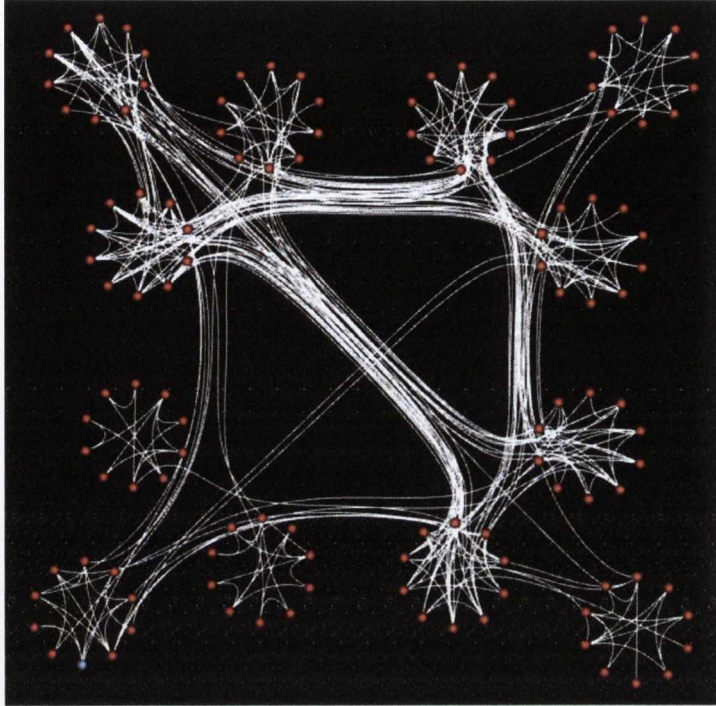


Figure 5.16: An example of the lower density experiment graph, unoptimised for layout and using the same colour scheme as used for the experiment, $\beta = 0.8$.

high contrast and bright white background caused viewer discomfort. To reduce this, the colours were inverted and the graphs were displayed on a black background, with edges shaded from white to grey based on length, as can be seen in figure 5.16. The billboarded glyphs that represent the nodes showed no artefacts when displayed, most likely because they were spheres and squares, which would not have a visual disparity between a left eye and right eye image.

In our previous experiments, we also shaded edges based on their length in a manner similar to Holten [Holo6]. This allowed overdrawn edges to be more distinguishable in a two-dimensional rendering. It does also unfortunately make shorter edges more prominent as they appear less faded, in particular if there is line anti-aliasing, as the more solid lines will appear slightly thicker when blended with the background. Furthermore, the attribute that is encoded in the edge translucency is also used to encode the depth in one of the experiment depth functions (where edge depth is defined based on length). There is a risk of this adding a confounding factor and interfering with user performance as the curve depth will now also relate to its translucency. One way avoid such interferences would be to remove all alpha blending from the experiment graphs and compare the addition of stereoscopic depth to edges to flat graphs without edge shading. However bundling frequently uses shading information in many implementations, and there would be a risk of stacking the experiment in favour of stereoscopic depth by removing it. The colouring scheme used for nodes used is the same as for our previous experiments.

Display Setup

For our display we used a 55" Samsung LCD 3D television. The 3D effect is created using active shutter glasses. The television display frequency for the experiment was 60Hz, resulting in a frame rate of 30Hz for each eye. The display resolution was 1600 x 1200 pixels and we rendered in 3D using the OpenGL Quad-Buffered Approach. When graphs were displayed on-screen they did not occupy the full height of the display to reduce the effect of frame cancellation [Waro4], an effect by which the screen edge appears to occlude the graph edge, diminishing the stereoscopic depth effect.

Camera Setup

For our previous experiments we used a perspective projection with an aperture (also known as field of view) of 60 degrees and with a standard computer monitor aspect ratio. However as described in section 2.6.2 other parameters are required for stereoscopic 3D.

It is possible to view objects on a stereoscopic display such that the depth and protrusion from the screen are exaggerated in comparison to what a viewer might experience when examining a real world object at a similar scale. However establishing the sort of bounds in parameters that allow this type of hyper-stereo and its impact is beyond the scope of our research. Therefore we adopt stereoscopic viewing parameters that model the scene as if it is a physical object in the real world, in front of the user and occupying the same space as the display. This should also reduce any eye strain resulting from differences in vergence, image disparity and focal length.

Due to the consistent graph size across experiments the camera is always at the same distance from the graph plane. This also allows us to use a constant camera focal length, set to 140 units. This is equal to the distance from the user to the display, in centimetres. We wish for the graph to pop out of the screen with some negative parallax, therefore we display it at 120 units from the user. The graphs were scaled so that their full extents would be visible on screen at the viewing distance. To ensure that the edges recede far enough from the graph to induce positive parallax, we set the maximum edge control point depth to 50 units, resulting in a maximum depth of 30 units behind the focal length. This is within the practical viewing volume of -25% to +60% of the viewer to screen distance described by Williams and Parrish [WP90]. For the eye separation we use 6 units, approximately the average interpupillary distance of the human eye. These values were tested by the author and gave a satisfying 3D effect without any resulting eye-strain, headache or discomfort. The range of graphs used for the experiment have a consistent volume and did not require any adjustment of these parameters, so they remained static for the length of the experiment. Previous work by Ware et al. [WGP98] and Carvalho [CTD⁺11] offers approaches for dynamic adjustment of stereo parameters which may be useful if a wide range of graph sizes were to be used, or to make the stereoscopic effect more pronounced, but was not

necessary for our experiments.

Participants

24 participants volunteered for our path tracing experiments there were . A 25th participant was excluded due to issues viewing stereoscopic 3D. All participants, bar one, had a background in computer science and had some level of familiarity with node link graphs. Each participant viewed each combination of factors once during the experiment. This was a total of 72 graphs, in addition to 6 initial training graphs.

21 participants volunteered for our clustering connectivity experiment. 3 additional participants were dropped due to an error in the data logging for the experiment. As for the path tracing experiment all participants, bar one, had a background in computer science and had some level of familiarity with node link graphs. Each participant viewed each combination of factors twice and the time and score were averaged across each repetition of factors. Each participant viewed a total of 48 graph rendering, in addition to six training graphs.

5.3.3 Initial Experimental Results

Significant Results - Path Tracing

For this experiment we performed a 5 way repeated measures ANOVA with within-subjects factors: layout \times depth \times edge density \times bundling level \times path length.

All significant results for the three dimensional path tracing experiment can be seen in tables 5.6 and 5.7. Figure 5.17 shows the significant main effects.

As expected from our previous experiments, graph density and bundling are significant main effects. It is clear that bundling across all the graphs still has a significantly negative impact (as it did in the two dimensional edge bundling experiments in section 5.1). There is however a notable increase in the effectiveness of the straight line edges (figure 5.17a), when compared to our previous experiments. This difference is not related to the use of 3D, as it applies to all depth functions and can be attributed to the narrower range of graphs and densities used in this experiment.

It can also be seen that there is a large difference in performance between the low and medium density graphs (figure 5.17b). The low level of performance across the medium density graphs reduces the likelihood of significant results when analysing bundling performance as well as the stereoscopic depth performance. This can be seen clearly in figure 5.19 where the significant difference between each level of bundling for the different layout types changes from $p < 0.02$ across all graphs to a more significant value of $p = 0.001$ across the lower density graphs.

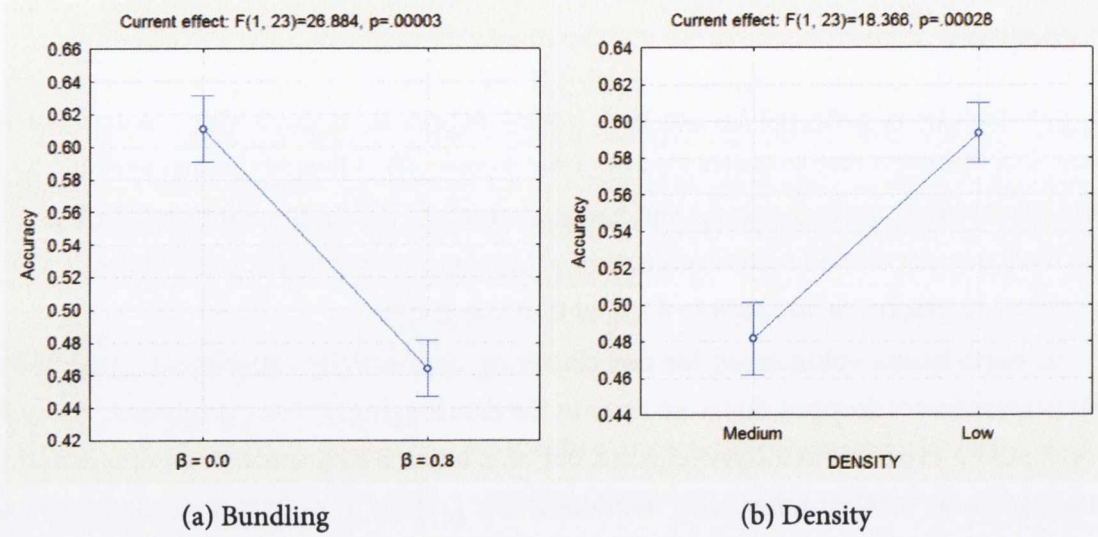


Figure 5.17: Unweighted means of the two significant single factors resulting from the Analysis Of VAriance for the path tracing experiment. The vertical error bars indicate + / - standard errors.

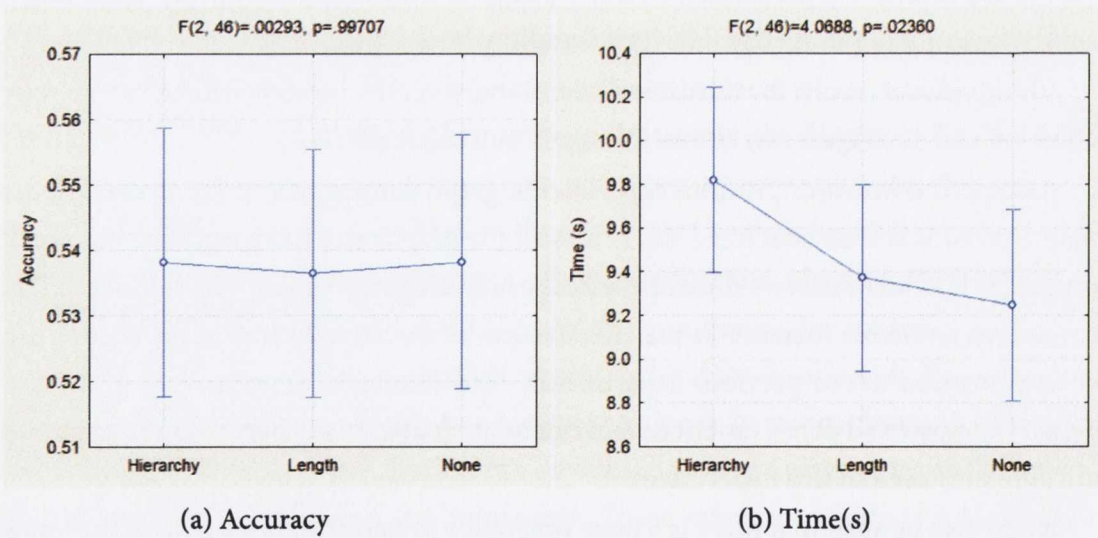


Figure 5.18: Unweighted means of the accuracy and user time taken for the depth factors, resulting form the Analysis Of VAriance for the path tracing experiment. The vertical error bars indicate + / - standard errors.

Effect	F-test	Comment
Bundling	$F(1,23)=26.884,$ $p=.00003$	As shown by the previous experiment bundling significantly impacted user accuracy at path tracing
Density	$F(1,23)=18.366,$ $p=.00028$	As expected density significantly impacts user performance at path tracing.
Layout , Depth	$F(2,46)=4.5614,$ $p=.01559$	The two different modes of depth (hierarchy and length based) are significantly different from each other, but not from graphs where no depth was used, see figure 5.19a.
Path Length , Bundling	$F(2,46)=5.5297,$ $p=.00705$	Newman keuls post hoc analysis showed that for answers of Path length 1 or 3 results became significantly worse if bundling was used. There was no significant difference for path length 2, however this length was significantly worse than the other path lengths for unbundled edges to begin with.
Bundling, Density	$F(1,23)=6.6957,$ $p=.01646$	The performance difference between the different levels of bundling was much smaller for the higher density graphs
Depth, Path Length, Bundling	$F(4,92)=3.6449,$ $p=.00840$	The impact of each depth function differed depending on path depth or bundling, however no clear conclusions can be drawn
Layout, Depth, Density	$F(2,46)=5.2946,$ $p=.00852$	As the large density graph impacted performance, the impact of layout combined shows as being significantly different between densities, see figure 5.19.

Table 5.6: Results for the ANOVA for the user performance at the path tracing task.

Effect	F-test	Comment
Bundling	F(1, 23)=6.7222, p=.01627	In addition to damaging user accuracy, bundling also means that user take longer on average to answer
Path Length	F(2, 46)=69.207, p=.00000	As previously seen, longer paths require a longer response time
Depth	F(2, 46)=4.0688, p=.02360	Post hoc analysis showed that the only statistical differences were between the hierarchical depth approach and length approach ($p = 0.039$) and the hierarchical depth and no depth ($p = 0.0252$). Hierarchical depth had a negative impact on user response time.
Density	F(1, 23)=5.9729, p=.02261	Density increased time response. This means not only did users more likely get the answers wrong for the more dense graphs, it took them longer to do so.
Bundling, Path Length	F(2, 46)=13.125, p=.00003	Bundling had a strong negative effect for path length of 1, a lesser but still negative effect of path length 2 and a small improvement for path length 3

Table 5.7: Results for the ANOVA for the user time taken at the path tracing task.

The Impact of Stereoscopic Depth

Performing an initial analysis of the the ANOVA results that the addition of stereoscopic depth has not resulted in a significant performance difference across graphs (as seen in figure 5.18a), nor is there and interaction effect with density ($p = 0.91251$). Due to the large error bounds as well as the extremely high p value, it looks as if there is no significant effect. In terms of time the hierarchy based approach significantly slowed down users by approximately half a second (see table 5.7 and figure 5.18b)

However due to the large number of factors there may be some cross-talk (i.e. different factors cancel each other out). The ANOVA analysis shows a significant effect for a combination of layout and depth type ($p = 0.01559$, see figure 5.19a). This significant effect is further exaggerated if we consider only the low density renderings ($p = 0.00065$, see figure 5.19b).

The previous lack of significant difference, when layout is not taken into consideration, is because the effects of each of the depth types (hierarchy and length) partially cancel each other out. The effect of no depth (none) is consistent across both layout types. We suspect that this occurs because optimising the layout indirectly impacts the depth function for one of the depth layout types, that is, when edge depth (edge distance from the graph plane) is set based on edge length. It is worth reiterating that regardless of the edge length the depths are scaled so that the furthest control point is 50 units behind the graph plane. Therefore

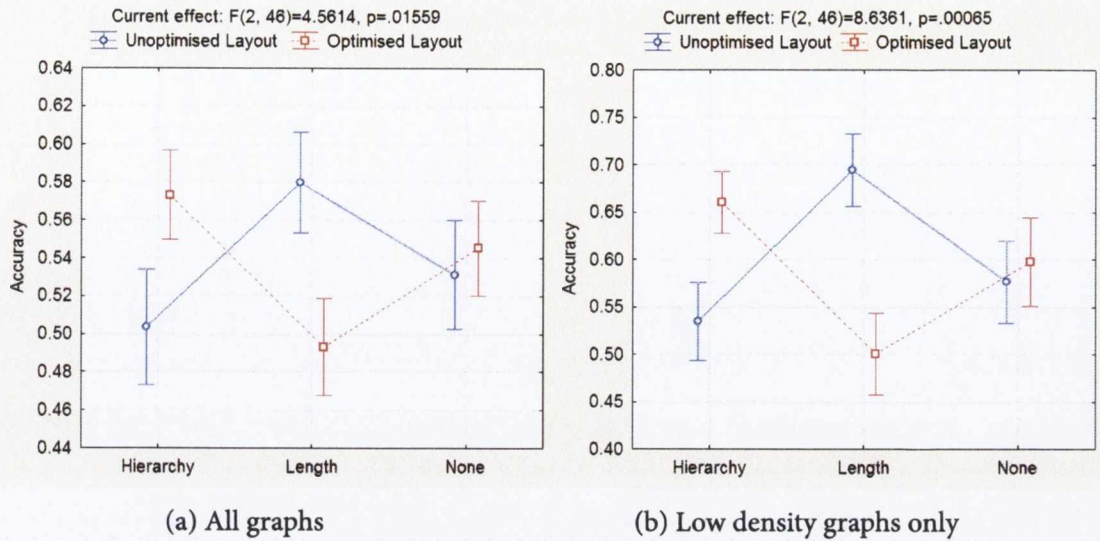


Figure 5.19: The depth types show significant results when analysed based on layout type. The significance is even greater when only low density graphs are considered. The vertical error bars indicate + / - standard errors.

Depth Function	Hierarchy	Length	None
Hierarchy		0.0260166069832669	0.42277286878992
Length	0.0260166069832669		0.114767729422493
None	0.42277286878992	0.114767729422493	

Table 5.8: Newman Keuls post-hoc analysis of path tracing accuracy for unoptimised low density graphs.

the range of depths is the same for both approaches, it is the distribution of depths in this range that changes. The hierarchy based approach has 3 distinct depths whereas the length based approach results in a wide range, in proportion from the length of the shortest to the length of the longest. Post-hoc Newman-Keuls analysis show that the only significant difference is between the two depth functions as can be see in tables 5.8 and 5.9. There were no significant differences between any of the layout types and the absence of 3D edge depth. There is also no significant interaction effect between depth and bundling ($p = 0.5742$). There were some further interaction effects for user response time which included density as an effect. Due to the negative impact of density on accuracy these effects are not considered (as tracing incorrect paths more quickly is of no benefit to users).

Significant Results - Clustering Connectivity

For this experiment experiment we performed a 5 way repeated measures ANOVA with within-subjects factors: layout \times depth \times edge density \times bundling level. Density was the only individual factor to provide a main effect, and it did so for both accuracy ($p = .00051$) and

Depth Function	Hierarchy	Length	None
Hierarchy		0.0260166069832671	0.231167958289438
Length	0.0260166069832671		0.247430025996758
None	0.231167958289438	0.247430025996758	

Table 5.9: Newman Keuls post-hoc analysis of path tracing accuracy for optimised low density graphs.

Factor	Effect	F-test	Comment
Acc.	Density	F(1, 20)=17.086, p=.00051	As for path tracing, density significantly impacted user accuracy
Acc.	Layout, Depth, Bundling	F(2, 40)=3.8383, p=.02986	
Acc.	Depth, Bundling, Density	F(2, 40)=3.9796, p=.02653,	
Time	Density	F(1, 20)=9.6431, p=.00558	Density increased time response. This means that not only did users more likely get the answers wrong for the more dense graphs, it took them longer to do so.

Table 5.10: Results for the ANOVA for the user accuracy and time taken at the Cluster Connectivity task.

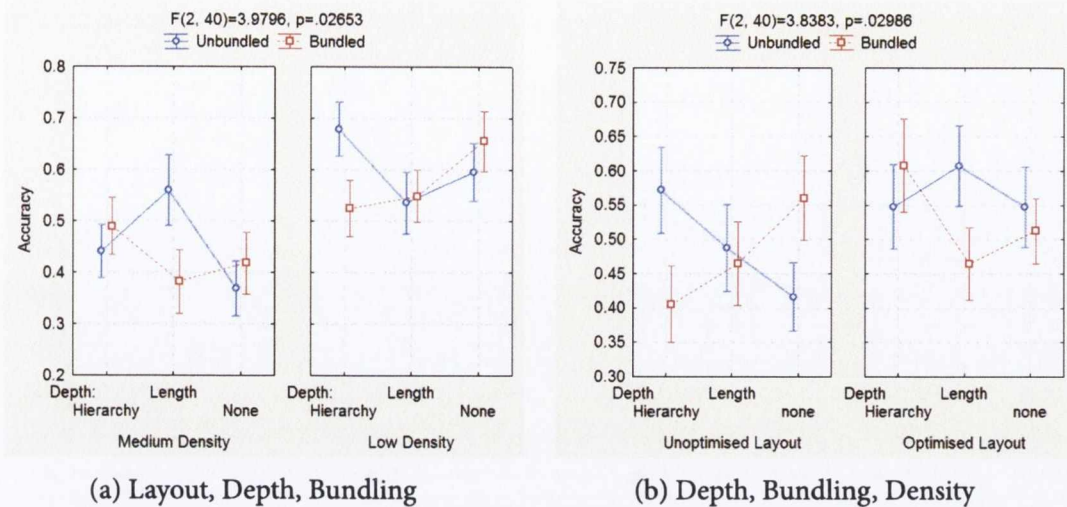


Figure 5.20: The significant interaction effects from the cluster connectivity experiment. The vertical error bars indicate \pm standard errors.

time taken ($p = .00558$). Unlike for our previous experiment, there was no significant main effect found for bundling improving user response time, however the value was approaching significance ($p = .08619$). This can be explained by the smaller number of graphs displayed to the user as well as possible interaction effects with other factors. Interaction effects were found between layout depth and bundling as well as between depth bundling and density see figure 5.20

Depth alone, nor interacting with another single factor, never has a significant impact on user performance. The fact that this performance depends on two other factors in such a complex manner suggests that it cannot draw any reliable conclusions about its benefit in visualising high level trends.

5.3.4 Follow On Path Tracing Experiment

The aim of introducing three dimensional depth was to mitigate the negative impact of bundling on low level path tracing tasks. Due to the result of our three dimensional path tracing experiment where there was an interaction effect between the two types of depth function, we decided to perform a follow on experiment to clarify the causes of the interaction. We aim to validate the effect of optimising layout and depth on user performance as was seen in the previous experiment (and can be seen in figure 5.19). As part of this goal we altered the experiment as described in the following subsection.

Experiment Changes

Graph Density Graph density is limited to only one type (low) as the negative impact of graph density is clear from the preceding experiments.

Three Dimensional Depth 3D depth is limited to only length based depth and no depth. This is due to the negative impact on time by hierarchy depth and is also to allow the user to view more rendering within the experiment in order to obtain more statistically significant results.

Edge Shading Edge shading was not considered a factor in our previous experiment. However, as we are using edge length as a measure to control edge depth and as this is the measure used to control the alpha blending of edges, we believe that it might have an impact on user performance. We therefore display shaded and unshaded edges to the user. An example of an experiment graph both shaded and unshaded can be seen in figure 5.21.

Experiment Graphs A limited set of random graphs was used in our previous experiment and rotated versions of the graphs were displayed to the user. For our follow on experiment we created a large set of random graphs structured using the same approach as before. The graph set contains two layouts of each graph: one optimised and one unoptimised. The random nature of the graph generation means that the optimisation process can have a wide range of improvements for graphs. Some graphs have a very large edge crossing reduction, some have little or none. The average edge crossing reduction across all graphs was 55.43% of edge intersections with a standard deviation of 27.41%. The average reduction of inter-cluster edge crossings was 17.08%, with a standard deviation of 11%. In addition to increasing the number of random graphs, due to participant feedback the trial time-out was extended to 30 seconds and participants were allowed enter an answer once the time expired.

Participants 14 participants took part in this experiment, all of whom had a background in computer science and had some level of familiarity with node link graphs. Each participant was shown each combination of factors 3 times, resulting in a total of 144 separate graph renderings, in addition to 6 training graphs.

Hypothesis

Our main hypothesis is as for our previous stereoscopic path tracing experiment. We hypothesised that adding depth information to the edge bundles in a direction perpendicular to the graph plane will improve user performance at a path tracing task when viewing the graph in stereoscopic 3D. If we add depth, perpendicular to the graph plane, to the graph edges and view the graph stereoscopically, user performance for path tracing of bundled graphs will be improved. We also want to investigate the impact of edge shading, particularly as edge depth is also based on edge length. Therefore we hypothesise that edge shading will have a significant impact on the users perception of the bundled edges, resulting in an improved level of accuracy, when compared to unshaded edges.

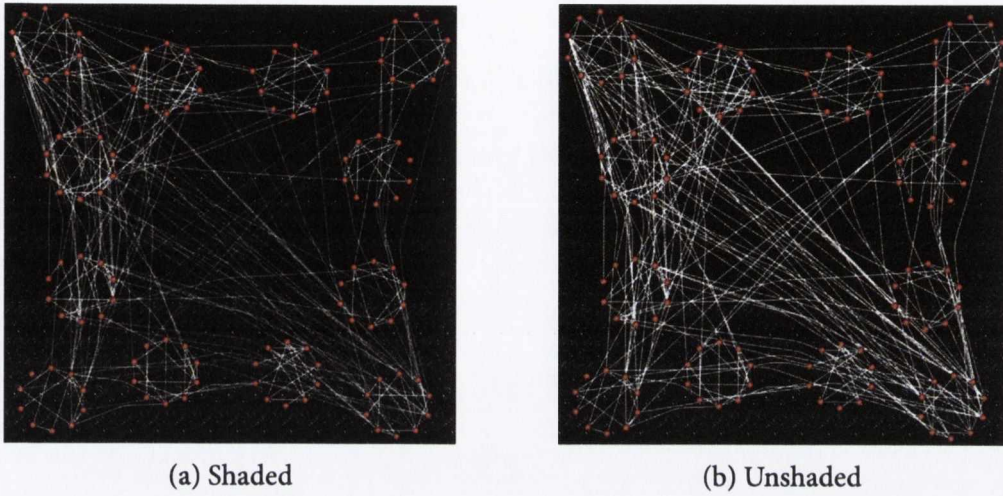


Figure 5.21: An experiment random graph shown with and without edge shading.

Factor	Count	Values
Layout	2	Un-optimised, Optimised
Depth	3	length , None
Path Length	3	1,2,3
Bundling	2	$\beta = 0.0$, $\beta = 0.8$
Shading	2	Unshaded , Shaded

Table 5.11: A summary of the various factors considered for the ANOVA analysis of the follow-on path tracing experiment.

Effect	F-test	Comment
Bundling	$F(1, 13)=112.67$, $p=.00000$	As for previous experiment bundling significantly negatively impacts path tracing accuracy.
Bundling, Path Length	$F(2, 26)=4.3225$, $p=.02395$	As for previous experiments, there was an interaction effect between bundling and path length.
Layout, Bundling, Depth	$F(1, 13)=4.9136$, $p=.04509$	The interaction effect due to depth is not very strong. Newman Keuls post-hoc analysis showed that depth is never significantly different for the same levels of bundling and layout. Bundling and layout alone approach significance ($p = 0.068$), so we can say depth is not contributing much as part of this interaction
Bundling, Depth, Path Length	$F(2, 26)=6.4849$, $p=.00519$	The only combination of other factors that depth is significant for is for bundled paths of length 3. The addition of depth results in accuracy being significantly worse ($p = 0.0363$)

Table 5.12: Significant results for the ANOVA for the participant accuracy at the second path tracing experiment.

5.3.5 Results

For this experiment we average the accuracy and time results across the three repetitions and performed a 5 way repeated measures ANOVA with within-subjects factors: layout \times depth \times edge density \times bundling level \times path length. The main significant results are summarised in tables 5.12 and 5.13. It is clear from the results that the addition of depth to edges does not improve path tracing, whether or not the edges are bundled, as it provided no significant difference in user score ($p = 0.951$). While there were some interaction effects which included depth, (see figures 5.22, and 5.23), its impact as part of the interaction appears to be quite small. Depth also shows a significantly negative impact on time, with users averaging 1.37 seconds longer ($p = .00467$). Edge shading did not play a part in any significant effects for accuracy or time. Therefore both of our hypotheses for this experiment have been proven false. We discuss why the addition of stereoscopic depth failed to improve participant accuracy at path tracing for our test graphs in our conclusions.

Effect	F-test	Comment
Depth	$F(1, 13)=11.614,$ $p=.00467$	Three dimensional depth slowed down users by 1.37 seconds on average
Path Length,	$F(2, 26)=49.615,$ $p=.00000$	As for all previous experiments, the larger the geodesic path, the larger the time taken.
Bundling Path, Length	$F(2, 26)=23.174,$ $p=.00000$	As for our initial stereoscopic experiment bundling had a strong negative effect for path length of 1, a lesser but still negative effect of path length 2 and a small improvement for path length 3

Table 5.13: Significant results for the ANOVA for the participant time taken at the second path tracing experiment.

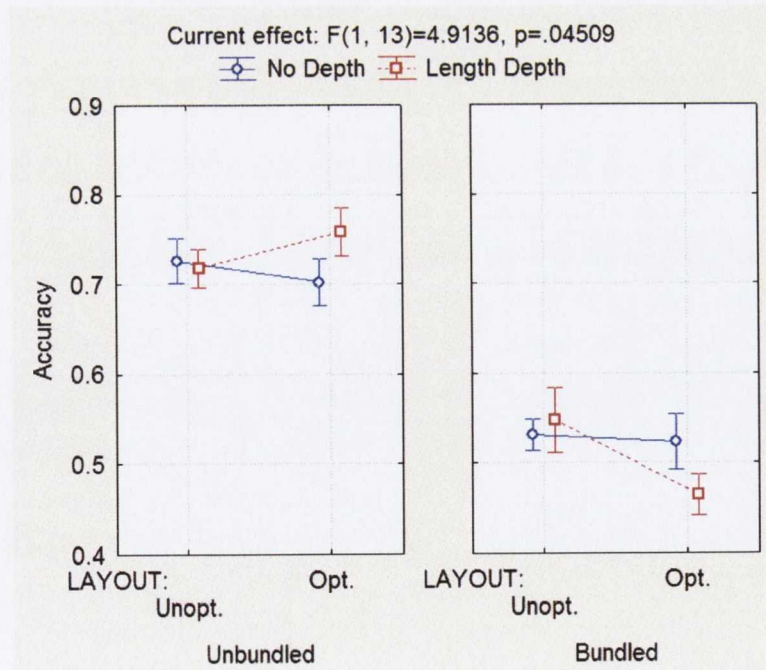


Figure 5.22: The significant interaction effect between bundling, layout and depth for the follow on path tracing experiment. The vertical error bars indicate + / - standard errors.

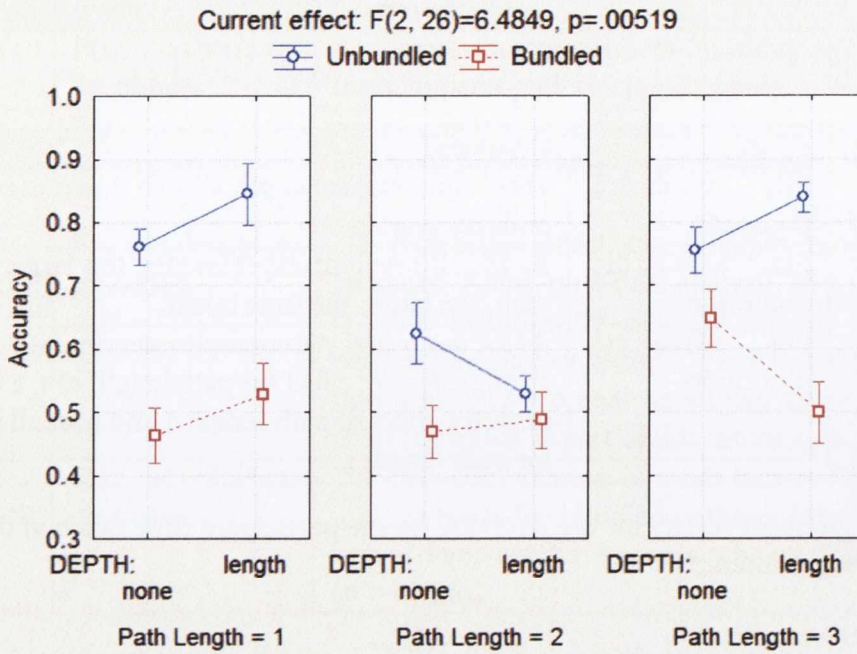


Figure 5.23: The significant interaction effect between bundling, depth and path length for the follow on path tracing experiment. The vertical error bars indicate + / - standard errors.

5.4 Conclusions

5.4.1 Edge Bundling

Path Tracing

From our experimental results it is clear that, within the context of our experimental setup, bundling does not aid users. It actually hinders them at path tracing tasks, not only in terms of accuracy but also in terms of time taken to complete the task. An important caveat for this result is that no visual enhancements for the bundling other than translucency were used. As previously mentioned, colour is often used in edge bundles. This may have a significant effect when it comes to distinguish individual edges within bundles even if Holten and van Wijk's [HvW09] user study on how best to render directed edges indicated colouring was not the best option, as this was not in the context of bundled edges. The use of colour and depth to aid in the perception of bundles is certainly a topic worthy of further work.

Furthermore, the positioning of nodes within clusters relative to bundles and their associated control points may have a significant impact. Within the experiments, no consideration was given to the routing of bundles around clusters or nodes in order to avoid confounding factors between the bundles and straight line graphs. Adjustment to the cluster hierarchy control points, algorithmically or with user intervention, could further reduce visual ambiguity that results from this.

Cluster Connectivity Path Tracing

For recognising high level patterns, bundling offered no significant improvement for intermediate levels of bundling, and is a hindrance when the tightest possible bundles are used. However bundling does produce a significant positive effect on the time taken to answer, without negatively impacting accuracy. Therefore it can be recommended when understanding high level connectivity is important. In terms of the level of bundling used we found no significant differences between the $\beta = 0.7$ and $\beta = 0.9$ levels of bundling for path tracing based task or recognising higher level patterns. While the $\beta = 1.0$ does provide a clearer view of the graph nodes, it is of no benefit in terms of graph comprehension.

5.4.2 Three Dimensional Stereoscopic Edge Bundling

It is clear from our results that our approach for adding depth to edges did not benefit the user, and specifically did not counter the negative impact of edge bundling as was originally expected. Depth frequently played a role in interaction effects, but never had a significant impact as a measure alone, nor in conjunction with edge bundling. In our initial three dimensional path tracing experiment, there was an interaction with layout, in terms of edge crossings. To investigate the relationship further we ran an additional experiment, where every graph was randomly generated and edge shading was added as a factor. The fact that no significant effect was found indicates that the interaction between depth and layout in the initial experiment was derived from characteristics of the graph other than the layout optimisation. It is clear from this, as well as the fact that depth often interacted with multiple other factors, that the role of stereoscopic depth when perceiving edges is subtle and relies on characteristics of a graph other than the ones used to define our experiment graphs. Another potential source for the lack of significance was the fact that we used quite straight forward measures to determine the depth of edges, however a more complex depth function may lead to different results.

Previous work has shown the stereoscopic rendering of three dimensional graphs to be beneficial [WF96, WMo8, SM93, HHL10]. While our work may appear to contradict these findings, there is one key characteristic about our approach which differentiates our work from the existing literature, and also explains why we saw no benefit where others did. Our nodes remained in a flat plane, only the edges were extended into three dimensions. We avoided the the positioning of nodes in three dimensions to allow us to evaluate the performance of three dimensional edges independently of three dimensional node positioning. However, clearly the position of nodes in three dimensions plays an important role in the traceability of paths. The continuity of edges in a path being traced is an important graph aesthetic [WPCMo2]. The fact that our three dimensional edges began and ended perpendicular to the graph plane would have resulted in a lack of three dimensional continuity of the path being traced (in addition to the lack of continuity in the two dimensional graph

plane that bundling can cause).

One potential solution to this would be a clustered graph layout where the clusters lie at different stereoscopic depths. This would allow for more continuity between edges and the varying positions of the nodes in three dimensions should make the edges joining them easier to perceive. Such a layout would have to consider many problems not encountered in our experiments, such as the occlusion of nodes, the layout of clusters (as a circular layout of cluster nodes may not be useful within a fully three dimensional layout), as well as a new method to extend bundling into 3D.

Chapter 6

Conclusions and Future Work

This chapter summarises the conclusions of our research and provides directions for future work on visualising dense small world graphs, using agglomerative clustering around nodes of interest.

6.1 Conclusions

Graph visualisation is an extremely broad field, covering many subtopics. There is more to the visualisation of a graph than an aesthetically pleasing positioning of nodes. To this end we examined many different aspects of the field. We examined the clustering of a graph around nodes of interest to the user. We showed how such a clustering can be used as a basis for laying out a graph. Once a graph node has been positioned the routing of edges plays a large role in comprehensibility of a graph, so we also examined edge routing techniques providing a thorough evaluation based on user performance at low level tasks. For our clustering and layout we predominantly considered small world graphs, as these characteristically reflect many real-world graphs and feature a level of structure which we believed to be useful for graph clustering. It is also possible to procedurally generate small world graphs, where other characteristics such as node count and edge density could be altered, to allow for thorough experimentation and validation.

6.1.1 Graph Clustering

In chapter 3 we introduced our approach to agglomerative clustering of small world graphs around nodes of interest. An important difference between our clustering approach and many existing approaches is that we use a set of input nodes specified by the user as the basis for building the clustering. Each node of interest specified by the user forms a basis or a cluster, which is grown agglomeratively.

We suggested that average clustering coefficient of clusters could be an effective heuristic for our agglomerative clustering, testing its use on a mix of procedurally generated and

real world graphs. We then refined our clustering approach further and evaluated it using a large range of procedurally created graphs. We also investigated the effects of our clustering approach using a benchmark dataset describing papers in the field of information visualisation and the citations between them.

We found that clustering coefficient makes an effective heuristic for agglomeratively clustering small world graphs around nodes of interest. We also demonstrated this using an example of clustering around four papers of interest in the information visualisation data set. For our example case, using clustering coefficient as a heuristic produced clusters which were more aligned with the classification of papers by keyword (provided as part of the dataset), than when modularity was used as a heuristic. However, modularity, a metric relating to the number of edges between clusters, was still a useful heuristic. It provided effective results for procedurally generated small world graphs, and also provided effective results, compared to other heuristics, for procedurally generated graphs which lacked the structure of small world graphs.

6.1.2 Graph Layout

Clustering graphs, using our agglomerative approach, groups nodes together around the nodes of interest specified as an input. However this data must still be presented to the user. In chapter 4 we discussed approaches to graph layout using a clustering hierarchy. We demonstrated how clusters can be laid out using circular layouts and demonstrated how inter-cluster edge crossings can be reduced using intelligent initial placement of nodes and cluster rotation for hierarchically clustered circle layouts.

In section 4.4 we utilised our agglomerative clustering approach to develop a multi-level layout. We showed how the relationships between the initial top level clusters can be replicated throughout the generated graph hierarchy. We demonstrated two approaches to layout, hierarchical and multilevel, that utilised the hierarchical clustering. The layouts reinforced the relationships between the clusters formed around the nodes of interest, positioning sub-clusters close to the higher level clusters that they were more strongly related to. Of the two layout approaches, multilevel layout gave a more appealing result in terms of cluster spacing. We also showed how the hierarchy can be utilised to generate hierarchical edge bundles, to help clarify the relationships between the clusters.

6.1.3 Edge Routing

Graph layout algorithms primarily position nodes, often giving consideration to the relationships between nodes, defined by the graph edges. However, most algorithms do not directly consider edge routing. This results in edges being a source of clutter for many high density graphs. Edge bundling is a recently popular clutter reduction technique. In chap-

ter 5 we examined the impact of edge bundling using user experiments, which focused on user perception of low level node connectivity, as well as higher level edge trends. The hierarchical nature of our layout, described in chapter 4, naturally lends itself to Holten's approach of hierarchical edge bundling. We developed an approach for generating hierarchically clustered graphs for user experiments. In these compound graphs, inter-cluster edges were distributed using a power-law and intra-cluster edges were given a flat probability of distribution. We used our test graphs to validate edge bundling over a range of densities and node counts. We showed that while edge bundling negatively impacts a user's ability to trace paths at a low level, it allowed users to recognise higher level trends more quickly.

Given that previous work has shown that stereoscopic rendering of graphs laid out in three dimensions aids users at a low level path tracing task, we performed experiments to determine if stereoscopic viewing could be used to counteract the negative effects associated with edge bundling on path tracing. We developed approaches for rendering edges with stereoscopic depth for a graph drawn in a flat two dimensional plane, by adding depth perpendicular to the graph plane to edges. We performed user experiments, utilising a stereoscopic display and active shutter glasses. Our experiments examined the impact of different approaches to depth, as well as the impact of shading and layout, with respect to the reduction of edge crossing. The resulting data showed no statistically significant benefit to using depth when viewed under stereoscopic conditions. Our results illustrate the difficulty of utilising three dimensional viewing, in that it's difficult to characterise any performance impact alone based on stereoscopy, independent from other factors such as characteristics of the graph.

6.2 Future Work

6.2.1 Graph Clustering

We have demonstrated the effectiveness of our agglomerative approach to graph clustering, however there are many variations and enhancements possible for future work. Our current approach begins with a single node of interest per cluster. It would be possible to allow for multiple nodes to be assigned per cluster initially, allowing users to have more control over clustering. We have provided a detailed evaluation, however further evaluation is possible. We utilised some established heuristics from the literature and practical considerations restricted us from including others. Other metrics are available and new ones will most likely be introduced in the future.

A higher level user qualitative evaluation, based around real data and a high level task and a user survey, such as that of Ridsen *et al.* [RCMCoo], would help gauge how useful subject matter expert users find different metrics to guide the agglomeration process. It

would also help definitively determine how users would use the ability to rearrange graphs around nodes of interest, to aid in a task. Such an evaluation is a very significant undertaking, in terms of time and resources, and was beyond the scope of this thesis.

Another possible improvement would be to automate the selection of nodes of interest to create a more general approach, for when the user does not have nodes of interest in mind prior to graph analysis. For our experiments in section 3.5, we automated the selection of nodes of interest by selecting nodes based on node degree. This approach was purely practical, ensuring that the similar nodes were selected for each run of the algorithm for different graphs. There are other centralities such as vertex betweenness or node clustering coefficient which could be used to specify an initial node set for the agglomerative clustering. These centralities could possibly be used in conjunction with an independent set filtration, as used by the GRIP layout algorithm[GK01], to ensure a distribution of nodes across the diameter of the graph. Any approach used to generate an initial node set would need to be thoroughly evaluated experimentally against a wide range of input graphs and varying agglomerative clustering input node sets, to be sure that resulting clusterings are of high quality and stable across a range of input graphs. The performance of our agglomerative clustering with different sizes of automated initial nodes sets, on graphs with different characteristics such as size density and edge distribution could also prove significant in terms of automating the input node set.

6.2.2 Graph Layout

Our graph layout approaches are an initial step, and not a complete solution. As stated in section 4.6 further work is required on the weighting of graph nodes for force directed layout. Further analysis is needed on the usefulness of generating a layout using a multilevel hierarchy of more than a few levels deep. We also need to examine the impact of much larger scales of graph, as well as the practicality of other layouts (such as some form of force directed layout) for the nodes of leaf clusters.

Graph layout is one of the largest sub-fields in graph visualisation, with many publications focusing solely on the drawing of graphs. There are a wide range of possible layouts, many of which could be alternatives to the approaches described in this thesis. There exist comparisons of graph layouts such as that by Hachul and Jünger [HJ06] and Bartel *et al.* [BGKM11]. None of the algorithms compared are based around specifying an input set of nodes, therefore a direct comparison to our approach would be difficult. The successful development of an automated node selection set, as just described in section 6.2.1, would allow a proper comparison. The extension of edge routing into 3D failed to provide us with a significant increase in performance. However, adopting our layout into a 3D approach, where nodes and clusters are placed at different depths may allow us to reap the benefits described by Mitchell and Ware[WMO8].

6.2.3 Edge Routing

There are many different approaches to bundling, for our evaluation we utilised the Holten's hierarchy based approach [Holo6], which popularised the technique. Holten's approach was suitable for our evaluation as our hierarchical layout approach provides the hierarchy structure required for routing edges. Other approaches exist and may fare differently under evaluation. Our evaluation of edge routing did not consider node avoidance resulting in some nodes being drawn over edges that they are not connected to. It may be possible to develop Holten's technique further, through the introduction of extra control points, so that edges bundles avoid nodes that are not part of their source or destination.

Our evaluation of edge bundling with stereoscopic viewing showed that using three dimensional depth had no significant impact on user performance. Our approach focused on using three dimensional depth for the edges between nodes. Further experimentation on utilising edge bundling between clusters at different depths as part of a full three dimensional graph layout may produce different results.

Bibliography

- [ACJM03] D. Auber, Y. Chiricota, F. Jourdan, and G. Melancon. Multiscale visualization of small world networks. In *Information Visualization, 2003. INFOVIS 2003. IEEE Symposium on*, pages 75–81, 2003.
- [AGB96] B.J. Rogers A. Glennerster and M.F. Bradshaw. Stereoscopic depth constancy depends on the subject's task. *Vision Research, Elsevier*, 36(21):3441 – 3456, 1996.
- [AK95] Charles J. Alpert and Andrew B. Kahng. Recent directions in netlist partitioning: a survey. *Integration, the VLSI Journal*, 19(1-2):1 – 81, 1995.
- [AMA07] D. Archambault, T. Munzner, and D. Auber. Topolayout: Multilevel graph layout by topological features. *Visualization and Computer Graphics, IEEE Transactions on*, 13(2):305 –317, march-april 2007.
- [AT12] AT and T. The at and t graph collection, 2012. available at <http://www.graphdrawing.org/>.
- [BB00] André M. S. Barreto and Helio J. C. Barbosa. Graph layout using a genetic algorithm. In *SBRN '00: Proceedings of the VI Brazilian Symposium on Neural Networks (SBRN'00)*, page 179, Washington, DC, USA, 2000. IEEE Computer Society.
- [BB05] Michael Baur and Ulrik Brandes. Crossing reduction in circular layouts. In Juraj Hromkovic, Manfred Nagl, and Bernhard Westfechtel, editors, *Graph-Theoretic Concepts in Computer Science*, volume 3353 of *Lecture Notes in Computer Science*, pages 332–343. Springer Berlin / Heidelberg, 2005.
- [BD07] M. Balzer and O. Deussen. Level-of-detail visualization of clustered graph layouts. In *Visualization, 2007. APVIS '07. 2007 6th International Asia-Pacific Symposium on*, pages 133–140, 2007.
- [BG09] R.A. Bittencourt and D. Guerrero. Comparison of graph clustering algorithms for recovering software architecture module views. In *Software Main-*

tenance and Reengineering, 2009. CSMR '09. 13th European Conference on, pages 251–254, march 2009.

- [BGKM11] Gereon Bartel, Carsten Gutwenger, Karsten Klein, and Petra Mutzel. An experimental evaluation of multilevel layout methods. In Ulrik Brandes and Sabine Cornelsen, editors, *Graph Drawing*, volume 6502 of *Lecture Notes in Computer Science*, pages 80–91. Springer Berlin / Heidelberg, 2011. 10.1007/978-3-642-18469-7.
- [BH04] F. Boutin and M. Hascoet. Cluster validity indices for graph partitioning. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 376–381, july 2004.
- [BPG00] Mark F Bradshaw, Andrew D Parton, and Andrew Glennerster. The task-dependent use of binocular disparity and motion parallax information. *Vision Research, Elsevier*, 40(27):3725–3734, 2000.
- [Bra01] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177, 2001.
- [CF09] Du Cai-Feng. High clustering coefficient of computer networks. In *Information Engineering, 2009. ICIE '09. WASE International Conference on*, volume 1, pages 371–374, 2009.
- [Che05] C. Chen. Top 10 unsolved information visualization problems. *Computer Graphics and Applications, IEEE*, 25(4):12–16, 2005.
- [CHH⁺08] Weiwei Cui, Zhou Hong, Qu Huamin, Wong Pak Chung, and Li Xiaoming. Geometry-based edge clustering for graph visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1277–1284, 2008.
- [Chu97] Fan R. K. Chung. *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92)*. American Mathematical Society, February 1997.
- [CJM03] Y. Chiricota, F. Jourdan, and G. Melancon. Software components capture using graph clustering. In *Program Comprehension, 2003. 11th IEEE International Workshop on*, pages 217–226, may 2003.
- [CK95] J. Carriere and R. Kazman. Research report. interacting with huge hierarchies: beyond cone trees. In *Information Visualization, 1995. Proceedings.*, pages 74–81, oct. 1995.
- [CM83] Thomas F. Coleman and Jorge J. Mor. Estimation of sparse jacobian matrices and graph coloring problems. *SIAM Journal on Numerical Analysis*, 20(1):pp. 187–209, 1983.

- [CTD⁺11] F. Carvalho, D.R. Trindade, P.F. Dam, A. Raposo, and I.H.F. dos Santos. Dynamic adjustment of stereo parameters for virtual reality tools. In *Virtual Reality (SVR), 2011 XIII Symposium on*, pages 66–72, may 2011.
- [Del05] B. Delaney. Forget the funny glasses [auto-stereoscopic display systems]. *Computer Graphics and Applications, IEEE*, 25(3):14–19, may-june 2005.
- [Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. 10.1007/BF01386390.
- [Ead84] P Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.
- [ED07] G. Ellis and A. Dix. A taxonomy of clutter reduction for information visualisation. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1216–1223, nov.-dec. 2007.
- [EF97] Peter Eades and Qing-Wen Feng. Multilevel visualization of clustered graphs. In Stephen North, editor, *Graph Drawing*, volume 1190 of *Lecture Notes in Computer Science*, pages 101–112. Springer Berlin Heidelberg, 1997.
- [EHP⁺11] O. Ersoy, C. Hurter, F.V. Paulovich, G. Cantareiro, and A. Telea. Skeleton-based edge bundling for graph visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2364–2373, dec. 2011.
- [ER59] P. Erdős and A. Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.
- [ET07] Niklas Elmqvist and Philippas Tsigas. Citewiz: a tool for the visualization of scientific citation networks. *Information Visualization*, 6(3):215–232, 2007.
- [ETNG⁺08] N. Elmqvist, Do Thanh-Nghi, H. Goodell, N. Henry, and J. D. Fekete. Zame: Interactive large-scale graph visualization. In *Visualization Symposium, 2008. PacificVIS '08. IEEE Pacific*, pages 215–222, 2008.
- [FLM95] Arne Frick, Andreas Ludwig, and Heiko Mehldau. A fast adaptive layout algorithm for undirected graphs. In *GD '94: Proceedings of the DIMACS International Workshop on Graph Drawing*, pages 388–403, London, UK, 1995. Springer-Verlag.
- [Flo62] Robert W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345–, June 1962.

- [FR91] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience, John Wiley and Sons, Ltd.*, 21(11):1129–1164, 1991.
- [FT07] Y. Frishman and A. Tal. Multi-level graph layout on the gpu. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1310–1319, 2007.
- [FvNo6] Ham Frank van and Krishnan Neeraj. Ask-graphview: A large scale graph visualization system. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):669–676, 2006. 1187842 Member - James Abello.
- [GFC04] M. Ghoniem, J.-D. Fekete, and P. Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 17–24, 0-0 2004.
- [GHGH09] Apeksha Godiyal, Jared Hoberock, Michael Garland, and John Hart. Rapid multipole graph drawing on the gpu. In Ioannis Tollis and Maurizio Patrignani, editors, *Graph Drawing*, volume 5417 of *Lecture Notes in Computer Science*, pages 90–101. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-642-00219-9.
- [GK01] P. Gajer and S.G. Kobourov. Grip: Graph drawing with intelligent placement. *LNCS Graph Drawing 2000*, 1984:222–238, 2001.
- [GK07] Emden R. Gansner and Yehuda Koren. Improved circular layouts. In *GD'06: Proceedings of the 14th international conference on Graph drawing*, pages 386–398, Berlin, Heidelberg, 2007. Springer-Verlag.
- [GKN05] Emden Gansner, Yehuda Koren, and Stephen North. Graph drawing by stress majorization. In Jnos Pach, editor, *Graph Drawing*, volume 3383 of *Lecture Notes in Computer Science*, pages 239–250. Springer Berlin / Heidelberg, 2005.
- [HEH08] Weidong Huang, Peter Eades, and Seok-Hee Hong. Beyond time and error: a cognitive approach to the evaluation of graph drawings. In *BELIV '08: Proceedings of the 2008 conference on BEYond time and errors*, pages 1–8, New York, NY, USA, 2008. ACM.
- [HFM07] N. Henry, J.-D. Fekete, and M.J. McGuffin. Nodetrix: a hybrid visualization of social networks. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1302–1309, nov.-dec. 2007.

- [HH10] Weidong Huang and Maolin Huang. Exploring the relative importance of crossing number and crossing angle. In *Proceedings of the 3rd International Symposium on Visual Information Communication, VINCE '10*, pages 10:1–10:8, New York, NY, USA, 2010. ACM.
- [HHL10] Djamel Hassaine, Nicolas S. Holliman, and Simon P. Liversedge. Investigating the performance of path-searching tasks in depth on multiview displays. *ACM Trans. Appl. Percept.*, 8(1):8:1–8:18, November 2010.
- [HJ05] Stefan Hachul and Michael Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In Jnos Pach, editor, *Graph Drawing*, volume 3383 of *Lecture Notes in Computer Science*, pages 285–295. Springer Berlin Heidelberg, 2005.
- [HJ06] Stefan Hachul and Michael Jünger. An experimental comparison of fast algorithms for drawing general large graphs. In Patrick Healy and Nikola S. Nikolov, editors, *Graph Drawing*, volume 3843 of *Lecture Notes in Computer Science*, pages 235–250. Springer Berlin Heidelberg, 2006.
- [HK01] David Harel and Yehuda Koren. A fast multi-scale method for drawing large graphs. In Joe Marks, editor, *Graph Drawing*, volume 1984 of *Lecture Notes in Computer Science*, pages 235–287. Springer Berlin / Heidelberg, 2001.
- [HK02] David Harel and Yehuda Koren. Graph drawing by high-dimensional embedding. In Michael T. Goodrich and Stephen G. Kobourov, editors, *Graph Drawing*, volume 2528 of *Lecture Notes in Computer Science*, pages 207–219. Springer Berlin Heidelberg, 2002.
- [HKKS03] John Hopcroft, Omar Khan, Brian Kulis, and Bart Selman. Natural communities in large linked networks. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '03*, pages 541–546, New York, NY, USA, 2003. ACM.
- [HMM00] I. Herman, G. Melancon, and M.S. Marshall. Graph visualization and navigation in information visualization: A survey. *Visualization and Computer Graphics, IEEE Transactions on*, 6(1):24–43, jan-mar 2000.
- [HN07] Pawan Harish and P. Narayanan. Accelerating large graph algorithms on the gpu using cuda. In Srinivas Aluru, Manish Parashar, Ramamurthy Badrinath, and Viktor Prasanna, editors, *High Performance Computing fb HiPC 2007*, volume 4873 of *Lecture Notes in Computer Science*, pages 197–208. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-77220-0.

- [Holo6] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):741–748, sept.-oct. 2006.
- [HVWo8] Danny Holten and Jarke J. Van Wijk. Visual comparison of hierarchically organized data. *Computer Graphics Forum*, 27(3):759–766, 2008.
- [HvWo9] Danny Holten and Jarke J. van Wijk. A user study on visualizing directed edges in graphs. In *Proceedings of the 27th international conference on Human factors in computing systems, CHI '09*, pages 2299–2308, New York, NY, USA, 2009. ACM.
- [HW79] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.
- [HW09] Danny Holten and Jarke J. van Wijk. Force-directed edge bundling for graph visualization. *Computer Graphics Forum*, 28(3):983–990, 2009.
- [Joh77] Donald B. Johnson. Efficient algorithms for shortest paths in sparse networks. *J. ACM*, 24(1):1–13, January 1977.
- [JS91] B. Johnson and B. Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *Visualization, 1991. Visualization '91, Proceedings., IEEE Conference on*, pages 284–291, oct 1991.
- [KCH03] Y. Koren, L. Carmel, and D Harel. Drawing huge graphs by algebraic multi-grid optimization. *Multiscale Modelling and Simulation*, 1(4):645–673, 2003.
- [KK89] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.*, 31(1):7–15, 1989.
- [KLo8] Ma Kwan-Liu. Stargate: A unified, interactive visualization of software projects. In *Visualization Symposium, 2008. PacificVIS '08. IEEE Pacific*, pages 191–198, 2008.
- [LBA10a] A. Lambert, R. Bourqui, and D. Auber. 3d edge bundling for geographical data visualization. In *Information Visualisation (IV), 2010 14th International Conference*, pages 329–335, july 2010.
- [LBA10b] A. Lambert, R. Bourqui, and D. Auber. Winding roads: Routing edges into bundles. *Computer Graphics Forum, Wiley*, 29(3):853–862, 2010.

- [LFR08] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78(4):046110, Oct 2008.
- [Lib12] BOOST Libraries. Boost libraries, 2012. available at <http://www.boost.org/>.
- [LLCM12] Sheng-Jie Luo, Chun-Liang Liu, Bing-Yu Chen, and Kwan-Liu Ma. Ambiguity-free edge-bundling for interactive graph visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 18(5):810–821, may 2012.
- [Lov10] Mike Love. Geneology of influence, 2010. available at <http://www.mike-love.net/genealogy/>.
- [LWH10] Lijuan Luo, Martin Wong, and Wen-mei Hwu. An effective gpu implementation of breadth-first search. In *Proceedings of the 47th Design Automation Conference, DAC '10*, pages 52–55, New York, NY, USA, 2010. ACM.
- [MD10] Fintan McGee and John Dingliana. An Evaluation of the use of Clustering Coefficient as a Heuristic for the Visualisation of Small World Graphs. pages 167–174, Sheffield, United Kingdom, 2010. Eurographics Association.
- [MD12a] Fintan McGee and John Dingliana. An empirical study on the impact of edge bundling on user comprehension of graphs. In *Proceedings of the International Working Conference on Advanced Visual Interfaces, AVI '12*, pages 620–627, New York, NY, USA, 2012. ACM.
- [MD12b] Fintan McGee and John Dingliana. Visualising small world graphs: Agglomerative clustering of small world graphs around nodes of interest. In *Proceedings of the International Conference on Information Visualisation Theory and Applications 2012 ()*, IVAPP 2012, 2012.
- [Melo6] Guy Melancon. Just how dense are dense graphs in the real world?: a methodological note. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, BELIV '06, pages 1–7, New York, NY, USA, 2006. ACM.
- [Mil67] S Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.
- [MKNT87] S. Masuda, t. Kashiwabara, K Nakajima, and Fujisawa T. On the np-completeness of a computer network layout problem. In *Proc. IEEE Intl. Symp. Circuits and Systems*,, pages 292–295, 1987.

- [MMo8] C. Muelder and Kwan-Liu Ma. Rapid graph layout using space filling curves. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1301 – 1308, nov.-dec. 2008.
- [MMCG99] S. Mancoridis, B.S. Mitchell, Y. Chen, and E.R. Gansner. Bunch: a clustering tool for the recovery and maintenance of software system structures. In *Software Maintenance, 1999. (ICSM '99) Proceedings. IEEE International Conference on*, pages 50 –59, 1999.
- [MMO05] Jonathan McPherson, Kwan-Liu Ma, and Michael Ogawa. Discovering parametric clusters in social small-world graphs. In *Proceedings of the 2005 ACM symposium on Applied computing, SAC '05*, pages 1231–1238, New York, NY, USA, 2005. ACM.
- [MMR⁺98] S. Mancoridis, B.S. Mitchell, C. Rorres, Y. Chen, and E.R. Gansner. Using automatic clustering to produce high-level system organizations of source code. In *Program Comprehension, 1998. IWPC '98. Proceedings., 6th International Workshop on*, pages 45 –52, jun 1998.
- [Mun98] T. Munzner. Exploring large graphs in 3d hyperbolic space. *Computer Graphics and Applications, IEEE*, 18(4):18–23, 1998.
- [MV09] L. Moussiades and A. Vakali. Benchmark graphs for the evaluation of clustering algorithms. In *Research Challenges in Information Science, 2009. RCIS 2009. Third International Conference on*, pages 197 –206, april 2009.
- [NEHH11] Quan Nguyen, Peter Eades, Seok-Hee Hong, and Weidong Huang. Large crossing angles in circular layouts. In Ulrik Brandes and Sabine Cornelsen, editors, *Graph Drawing*, volume 6502 of *Lecture Notes in Computer Science*, pages 397–399. Springer Berlin / Heidelberg, 2011.
- [New04] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69(6):066133, Jun 2004.
- [New10] Mark Newman. *Networks: An Introduction*. Oxford University Press, Inc., New York, NY, USA, 2010.
- [NG04] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004.
- [Plao4] Catherine Plaisant. The challenge of information visualization evaluation. In *Proceedings of the working conference on Advanced visual interfaces, AVI '04*, pages 109–116, New York, NY, USA, 2004. ACM.

- [PMCC01] Helen C. Purchase, Matthew McGill, Linda Colpoys, and David Carrington. Graph drawing aesthetics and the comprehension of uml class diagrams: an empirical study. In *APVis '01: Proceedings of the 2001 Asia-Pacific symposium on Information visualisation*, pages 129–137, Darlinghurst, Australia, Australia, 2001. Australian Computer Society, Inc.
- [PNBH12] Sergey Pupyrev, Lev Nachmanson, Sergey Bereg, and Alexander Holroyd. Edge routing with ordered bundles. In Marc van Kreveld and Bettina Speckmann, editors, *Graph Drawing*, volume 7034 of *Lecture Notes in Computer Science*, pages 136–147. Springer Berlin / Heidelberg, 2012. 10.1007/978-3-642-25878-7-14.
- [PPP12] H.C. Purchase, C. Pilcher, and B. Plimmer. Graph drawing aesthetics - created by users, not algorithms. *Visualization and Computer Graphics, IEEE Transactions on*, 18(1):81–92, jan. 2012.
- [Pur97] Helen C. Purchase. Which aesthetic has the greatest effect on human understanding? In *Proceedings of the 5th International Symposium on Graph Drawing, GD '97*, pages 248–261, London, UK, 1997. Springer-Verlag.
- [Pur98] H.C. Purchase. The effects of graph layout. In *Computer Human Interaction Conference, 1998. Proceedings. 1998 Australasian*, pages 80–86, nov-4 dec 1998.
- [QE01] Aaron Quigley and Peter Eades. Fade: Graph drawing, clustering, and visual abstraction. In Joe Marks, editor, *Graph Drawing*, volume 1984 of *Lecture Notes in Computer Science*, pages 197–210. Springer Berlin Heidelberg, 2001.
- [QZW07] Huamin Qu, Hong Zhou, and Yingcai Wu. Controllable and progressive edge clustering for large networks. In Michael Kaufmann and Dorothea Wagner, editors, *Graph Drawing*, volume 4372 of *Lecture Notes in Computer Science*, pages 399–404. Springer Berlin / Heidelberg, 2007.
- [RCMCoo] Kirsten Ridsen, Mary P. Czerwinski, Tamara Munzner, and Daniel B. Cook. An initial examination of ease of use for 2d and 3d information visualizations of web content. *International Journal of Human-Computer Studies, Elsevier*, 53(5):695–714, 2000.
- [RDLC12] Nathalie Henry Riche, Tim Dwyer, Bongshin Lee, and Sheelagh Cpendale. Exploring the design space of interactive link curvature in network diagrams. In *Proceedings of the International Working Conference on Advanced Visual Interfaces, AVI '12*, pages 506–513, New York, NY, USA, 2012. ACM.

- [RLM]05] Ruth Rosenholtz, Yuanzhen Li, Jonathan Mansfield, and Zhenlan Jin. Feature congestion: a measure of display clutter. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 761–770, New York, NY, USA, 2005. ACM.
- [RMC91] George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone trees: animated 3d visualizations of hierarchical information. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, CHI '91, pages 189–194, New York, NY, USA, 1991. ACM.
- [Scho07] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27 – 64, 2007.
- [SM93] Randy L. Sollenberger and Paul Milgram. Effects of stereoscopic and rotational displays in a three-dimensional path-tracing task. *Human Factors*, Sage, 35:483 – 499, 1993.
- [ST99] Janet Six and Ioannis Tollis. Circular drawings of biconnected graphs. In Michael Goodrich and Catherine McGeoch, editors, *Algorithm Engineering and Experimentation*, volume 1619 of *Lecture Notes in Computer Science*, pages 662–662. Springer Berlin / Heidelberg, 1999.
- [ST04] Alkiviadis Symeonidis and Ioannis Tollis. Visualization of biological information with circular drawings. In Jos Barreiro, Fernando Martn-Snchez, Vctor Maojo, and Ferran Sanz, editors, *Biological and Medical Data Analysis*, volume 3337 of *Lecture Notes in Computer Science*, pages 468–478. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-30547-7.
- [STT81] Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiro Toda. Methods for visual understanding of hierarchical system structures. *Systems, Man and Cybernetics, IEEE Transactions on*, 11(2):109 –125, feb. 1981.
- [SW05] Thomas Schank and Dorothea Wagner. Approximating clustering coefficient and transitivity. *Journal of Graph Algorithms and Applications*, 9:2005, 2005.
- [Tam87] R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.
- [TDoG13] University of Cologne University of Sydney TU Dortmund, Osnabrck University and oreas GmbH. The open graph drawing framework (ogdf), 2013. available at <http://www.ogdf.net/>.

- [Tufo1] Edward R. Tufte. *The Visual Display of Quantitative Information, 2nd edition*. Graphics Press, 2 edition, May 2001.
- [vDoo] S. M. van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, The Netherlands, 2000.
- [vHo4] F. van Ham. Case study: Visualizing visualization. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages r5–r5, 2004.
- [vHWo8a] F. van Ham and M. Wattenberg. Centrality based visualization of small world graphs. In A. Vilanova, A. Telea, G. Scheuermann, and T. Müller, editors, *Eurographics/ IEEE-VGTC Symposium on Visualization 2008*, Eindhoven, Holland, 2008.
- [VHWo8b] F. Van Ham and M. Wattenberg. Centrality based visualization of small world graphs. *Computer Graphics Forum*, 27(3):975–982, 2008.
- [vLKS⁺11] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J.J. van Wijk, J.-D. Fekete, and D.W. Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011.
- [vSvDZS⁺10] Boris W. van Schooten, Elisabeth M. A. G. van Dijk, Elena Zudilova-Seinstra, Avan Suinesiaputra, and Johan H. C. Reiber. The effect of stereoscopy and motion cues on 3d interpretation task performance. In *Proceedings of the International Conference on Advanced Visual Interfaces, AVI '10*, pages 167–170, New York, NY, USA, 2010. ACM.
- [WAB93] Colin Ware, Kevin Arthur, and Kellogg S. Booth. Fish tank virtual reality. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems, CHI '93*, pages 37–42, New York, NY, USA, 1993. ACM.
- [Wal01] C. Walshaw. A multilevel algorithm for force-directed graph drawing. In Joe Marks, editor, *Graph Drawing*, volume 1984 of *Lecture Notes in Computer Science*, pages 31–55. Springer Berlin / Heidelberg, 2001. 10.1007/3-540-44541-2.
- [Wal12] Christopher Walshaw. The graph partitioning archive, 2012. available at <http://staffweb.cms.gre.ac.uk/wco6/partition/>.
- [War04] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

- [Wato3] D.J. Watts. *Small worlds: the dynamics of networks between order and randomness*. Princeton studies in complexity. Princeton University Press, 2003.
- [WCo7] Nelson Wong and Sheelagh Carpendale. Supporting interactive graph exploration using edge plucking. In *Proceedings of Visualization and Data Analysis (VDA) 2007, SPIE*, volume 6495, pages 649508–649508–12, 2007.
- [WCGo3] N. Wong, S. Carpendale, and S. Greenberg. Edgelens: an interactive method for managing edge congestion in graphs. In *Information Visualization, 2003. INFOVIS 2003. IEEE Symposium on*, pages 51–58, oct. 2003.
- [WF96] Colin Ware and Glenn Franck. Evaluating stereo and motion cues for visualizing information nets in three dimensions. *ACM Trans. Graph.*, 15(2):121–140, April 1996.
- [WGP98] C. Ware, C. Gobrecht, and M.A. Paton. Dynamic adjustment of stereo display parameters. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 28(1):56–65, jan 1998.
- [Whe38] Charles Wheatstone. Contributions to the physiology of vision. part the first. on some remarkable, and hitherto unobserved, phenomena of binocular vision. *Philosophical Transactions of the Royal Society of London*, Vol. 128:371–394., 1838.
- [WHH05] J. Wu, A.E. Hassan, and R.C. Holt. Comparison of clustering algorithms in the context of software evolution. In *Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference on*, pages 525–535, sept. 2005.
- [Wil97] Graham J Willis. Nicheworks - interactive visualization of very large graphs. In *Proceeding of the 5th international Symposium on Graph Drawing GD '97*, pages 403–414, Rome, 1997. Springer.
- [WMo5] Colin Ware and Peter Mitchell. Reevaluating stereo and motion cues for visualizing graphs in three dimensions. In *Proceedings of the 2nd symposium on Applied perception in graphics and visualization, APGV '05*, pages 51–58, New York, NY, USA, 2005. ACM.
- [WMo8] Colin Ware and Peter Mitchell. Visualizing graphs in three dimensions. *ACM Trans. Appl. Percept.*, 5(1):1–15, 2008. 1279642.
- [WP90] Steven P. Williams and Russell V. Parrish. New computational control techniques and increased understanding for stereo 3-d displays. volume 1256, pages 73–82. SPIE, 1990.

- [WPCMo2] Colin Ware, Helen Purchase, Linda Colpoys, and Matthew McGill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110, 2002.
- [WRMW95] John P. Wann, Simon Rushton, and Mark Mon-Williams. Natural problems for stereoscopic depth perception in virtual environments. *Vision Research, Elsevier*, 35(19):2731 – 2736, 1995.
- [WS98] D. Watts and S. Strogatz. Collective dynamics of "small-world" networks. *Nature*, (393):440–442, 1998.
- [WSHEo8] Huang Weidong, Hong Seok-Hee, and P. Eades. Effects of crossing angles. In *Visualization Symposium, 2008. PacificVIS '08. IEEE Pacific*, pages 41–46, 2008.
- [YuFo8] Grinstein G. Plaisant C. YuFekete, J.-D. The history of infovis. *IEEE InfoVis 2004 Contest*, 14(6):1285–1292, 2008.
- [Zac77] Wayne W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research, University of New Mexico*, 33(4):452–473, 1977.
- [ZYC+o8] Hong Zhou, Xiaoru Yuan, Weiwei Cui, Huamin Qu, and Baoquan Chen. Energy-based hierarchical edge clustering of graphs. In *Visualization Symposium, 2008. PacificVIS '08. IEEE Pacific*, pages 55 –61, march 2008.