

A METHOD FOR GRAPH DRAWING UTILISING PATTERNS

A THESIS SUBMITTED TO
THE UNIVERSITY OF KENT
IN THE SUBJECT OF COMPUTER SCIENCE
FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY.

By
Robert Baker
July 2017

Abstract

This thesis describes a novel method for the layout of undirected graphs. It works by identifying certain patterns within the graph and drawing these in a consistent manner. For graphs to be useful and of benefit to a user, the result must be clear and easy to understand. This process attempts to draw graphs in such a manner.

Firstly, a background of graph problems and graph drawing is introduced, before the benefits of patterns are explained. Following this, there is an in-depth discussion of a number of existing graph drawing techniques, perceptual theories and methods for subgraph isomorphism.

This pattern-based method is then explained in great detail. Firstly, the patterns required are defined and examples given. Then, there is an explanation of the methodology involved in identifying these patterns within a graph. Following on from this, the order in which patterns are drawn based on their connection types to those already drawn is detailed, before a detailed description of each drawing method.

Evaluation of this method follows, starting with analysis mainly based on three real world data sources. This is in the form of side-by-side comparisons of graphs drawn with this method and a force-directed method. Following this, a metric based evaluation compares the two methods on edge crossings and occlusion, while also detailing some pattern based metrics.

Further evaluation continues in the form of an empirical study. The methodology of this study is detailed before results are displayed. Analysis of these results follows, with conclusions drawn.

Finally, potential further work is detailed and possible implementations discussed. All study materials and results are provided in the Appendix for those who wish to repeat the study or analysis.

Acknowledgements

My first thanks must go to my supervisor, Dr. Peter Rodgers, who has been a source of great support and advice throughout my Ph.D. I have learnt useful research and writing skills from him and his invaluable guidance has led me to both this research topic and the completion of this thesis. Thanks are also due to the other two members of my supervisory panel, Simon Thompson and David Barnes who have provided useful feedback, both within and outside of this role.

I must also thank my fiancée, Susannah, for her unwavering faith, support and encouragement throughout my doctoral studies, and to my parents for both their constant help and guidance, as well as creating in me a love of learning and an interest in science. I also thank my grandparents, who were always keen to hear how the research was going, despite never understanding the reply!

Acknowledgements must also go to Kai Xu and Rick Walker for many useful discussions at the beginning of this research project and to those other academics who showed interest in my work and provided valuable advice and feedback, including Helen Purchase, Daniel Archambault, Gem Stapleton, Bilal Alsallakh, and the attendees of Diagrams 2016. Thanks must also go to the members of SW104 in the autumn of 2012, who all provided lots of useful tips, encouragement and a welcoming atmosphere as I was starting my studies.

Contents

Abstract	ii
Acknowledgements	iv
Contents	v
List of Tables	xi
List of Figures	xii
List of Algorithms	xxiii
1 Introduction	1
1.1 History of Graph Problems and Graph Drawing	2
1.2 Why Patterns?	4
1.3 Summary of Chapters	5
2 Related Work	7
2.1 Graph Drawing Techniques	7
2.1.1 Barycentre Approach	9
2.1.2 Force-directed methods	9
2.1.3 Simulated annealing	13
2.1.4 Other Drawing Techniques	14

2.2	Graph Aesthetics and Perceptual Theories	17
2.3	Subgraph Isomorphism	25
2.3.1	Motifs	28
2.4	Summary	32
3	Description of Drawing Method	33
3.1	Introduction to the Drawing Method	33
3.2	Definition of Patterns	34
3.2.1	Circle	35
3.2.2	Clique	36
3.2.3	Star	36
3.2.4	Path	37
3.2.5	Triangle	38
3.2.6	Other Notation	38
3.3	Identification of patterns	39
3.3.1	Circle	40
3.3.2	Clique	42
3.3.3	Star	46
3.3.4	Path	47
3.3.5	Triangle	49
3.4	Determining the drawing order	50
3.5	Drawing each pattern	60
3.5.1	Share One Edge	61
3.5.2	Share Two Edges	68
3.5.3	Share One Node	73
3.5.4	Share Two Nodes	80
3.5.5	Connected by an Edge	83
3.5.6	Connected by Multiple Edges	89

3.5.7	No connections	90
3.5.8	Paths & Routes	92
3.5.9	Loose nodes	94
3.6	Methods Used in Numerous Drawing Techniques	99
3.6.1	Drawable Areas	99
3.6.2	Symmetry Score	104
3.6.3	Occlusion & Edge Crossing Score	107
4	Analysis of Generated Layouts	109
4.1	Introduction	109
4.1.1	Force Directed Method	109
4.2	Data Collection	110
4.2.1	Academy Award Nominees	110
4.2.2	Formula One Teammates	111
4.2.3	Character Relations in Novels	112
4.3	Examples of Drawing	113
4.3.1	Academy Award Nominees	114
4.3.2	Formula One Teammates	118
4.3.3	Character Relations in Novels	123
4.3.4	Other Data Sources	127
4.4	Quantitative Analysis of Examples	135
4.5	Summary	149
5	Formal Empirical Study	150
5.1	Introduction to Study	150
5.2	Methodology	151
5.2.1	Methodology Details	151
5.2.2	Task types	154

5.2.3	Survey and Preference Information	155
5.2.4	Participant Information	155
5.2.5	Data Sources	155
5.3	Results	157
5.4	Analysis	162
5.5	Threats to Validity	163
5.5.1	Internal Validity	164
5.5.2	Construct Validity	164
5.5.3	External Validity	165
5.6	Summary	166
6	Conclusion and Further Work	167
6.1	Contributions	167
6.2	Further Work	168
6.2.1	Increased Number of Defined Patterns	169
6.2.2	Increased Number of Connection Types	169
6.2.3	Tailoring to a Particular Domain	169
6.2.4	Multi-level approach	170
6.2.5	Flexible Drawing Order	171
6.2.6	Custom Patterns and Layouts	172
6.2.7	Hybrid System	174
	Bibliography	176
A	Full Analysis of Generated Layouts	194
A.1	Size of graphs	194
A.2	Time taken	203
A.3	Density of graphs	206

A.4	Size of Largest Clique	215
A.5	Patterns by Type	217
A.6	Discarded and Drawn Patterns	226
A.7	Discarded and Drawn Nodes	235
A.8	Node-node Occlusion	244
A.9	Node-edge Occlusion	252
A.10	Edge Crossings	261
A.11	Time taken	270
B	Empirical Study Data	279
B.1	Information Provided to Participants	279
B.1.1	Introduction	280
B.1.2	Practice Questions	281
B.1.3	Conclusion	285
B.2	Examples Used	286
B.2.1	Practice	286
B.2.2	Task Type 1	291
B.2.3	Task Type 2	296
B.2.4	Task Type 3	301
B.2.5	Task Type 4	307
B.3	Questions and Answers	312
B.3.1	Practice Questions	312
B.3.2	Main Study	315
B.4	Preference Sheet	322
B.4.1	Pair 1	322
B.4.2	Pair 2	323
B.4.3	Pair 3	324
B.5	Results	325

B.5.1 Main Study 325
B.5.2 Preferences 367

List of Tables

2.1	A summary of various graph drawing aesthetics	19
4.1	Rounds competed by Lotus-Renault drivers in the 2012 and 2013 Formula One seasons [89, 90]	112
5.1	Statistical Analysis of Force vs Pattern	157
5.2	χ^2 values from Friedman Analysis for Time and Accuracy	158
5.3	p -values for Force vs Pattern when grouped by Task Type	159
5.4	p -values for Force vs Pattern when grouped by Size	160
B.1	Practice Questions Asked	312
B.2	Questions Asked	316
B.3	Main study results	325
B.4	Preference results	367

List of Figures

1.1	Euler’s Seven Bridges of Königsberg	2
1.2	Key Players and Notable Relationships in the Middle East	3
1.3	MusicRoamer: Suggested artists for some of the Rat Pack	4
2.1	A graph drawn using Tutte’s Barycentre approach	9
2.2	Example of the forces used in Peter Eades’s method	10
2.3	Example of Local Optima	11
2.4	Example of the same graph drawn using two different force-directed methods	11
2.5	Example of the same graph drawn using a force-directed method and simulated annealing	15
2.6	Good and Bad Examples of Centrally Placing High Degree Nodes	18
2.7	Good and bad examples of clustering similar nodes	19
2.8	Good and bad examples of distributing nodes evenly	20
2.9	Good and bad examples of node-edge occlusion	20
2.10	Good and bad examples of node-node occlusion	20
2.11	Good and bad examples of minimising edge crossings	21
2.12	Good and bad examples of minimising edge variance	21
2.13	Good and bad examples of minimising the length of the largest edge	21
2.14	Good and bad examples of minimising the total length of edges .	22

2.15	Good and bad examples of maximising the minimum angle of edges leaving a node	22
2.16	Good and bad examples of consistent aspect ratio	23
2.17	Good and bad examples of minimising area	23
2.18	Good and bad examples of maximising symmetry	23
2.19	Example of subgraph isomorphism	26
2.20	Example of various motifs and glyphs	30
2.21	Glyphs and patterns representing graphs of two different sizes . .	31
2.22	Example of an obscured clique	31
3.1	Example of circles of size 6, 5 and 4	35
3.2	Example of cliques of size 6, 5 and 4	36
3.3	Example of stars of size 6, 5 and 4	37
3.4	Example of paths of size 4 and 5	38
3.5	Example of a triangle	38
3.6	Other Nodes	39
3.7	Four identical circles	40
3.8	A clique within a clique	43
3.9	Example of the clique scoring identification method	44
3.10	Graph with 10 nodes, but with 8 patterns	52
3.11	Clique with 1 edge shared	54
3.12	Clique with 1 node shared	54
3.13	Star with 1 outside node shared	54
3.14	Star with 1 edge shared	54
3.15	Star with centre node shared	55
3.16	Circle with 1 edge shared	55
3.17	Circle with 2 edges shared	55
3.18	Circle with 2 nodes shared	55

3.19	Circle with 1 node shared	56
3.20	Path with 2 edges shared	56
3.21	Path with 1 edge shared	56
3.22	Path with 2 nodes shared	56
3.23	Path with 1 node shared	56
3.24	Clique with connecting edges	56
3.25	Star with connecting edges	57
3.26	Circle with connecting edges	57
3.27	Path with connecting edges	57
3.28	Triangle with 1 edge shared	57
3.29	Triangle with 2 nodes shared	58
3.30	Triangle with 1 node shared	58
3.31	Triangle with connecting edges	58
3.32	Clique with no connection	58
3.33	Star with no connection	58
3.34	Circle with no connection	58
3.35	Path with no connection	59
3.36	Triangle with no connection	59
3.37	Comparison of 1 edge sharing cliques	59
3.38	Ideal Layouts for a Clique, Circle, Star & Triangle	60
3.39	Comparison of a drawn pattern with and without scaling	64
3.40	Comparison of a drawn pattern before and after reflection	65
3.41	Examples of the One Edge Sharing algorithm for Circles & Cliques	66
3.42	Example of the Edge Sharing algorithm for Stars	67
3.43	Possible drawing methods for patterns sharing two edges	68
3.44	Examples of the Multiple Edge Sharing algorithm	70
3.45	Example of Two Shared Non-Consecutive Edges	72

3.46	Examples of the Shared Node algorithm for Cliques, Circles, and Stars (outside node)	77
3.47	Example of Shared Node algorithm for Stars (centre node)	79
3.48	Example of two circles sharing two nodes	80
3.49	Example of a triangle sharing two nodes	82
3.50	Example of the grid based search algorithm	86
3.51	Example of the Edge Connected algorithm	87
3.52	Example of patterns connected by multiple edges	90
3.53	Example of two cliques having no connection	91
3.54	Example of various possible path locations	93
3.55	Example of Loose Nodes - Multiples	96
3.56	Examples of Loose Nodes - Singles	96
3.57	Examples of Area Intersections	100
3.58	The invalid areas and the best valid area when drawing pattern sB	101
3.59	Poor symmetry (1)	105
3.60	Poor symmetry (2)	105
3.61	Good symmetry	106
3.62	Symmetry Example	107
4.1	Yahoo’s suggested “People also search for” for Steven Seagal [4] .	111
4.2	1998 Best Leading and Supporting Actor and Actress Academy Award Nominees	114
4.3	1997 Best Leading and Supporting Actor and Actress Academy Award Nominees	115
4.4	2007 Best Leading Actor and Actress Academy Award Nominees	116
4.5	2006 Best Leading and Supporting Actor and Actress Academy Award Nominees	116

4.6	2004 Best Leading and Supporting Actor and Actress Academy Award Nominees	117
4.7	Formula One team-mates, 1996–1998	118
4.8	Formula One team-mates, 2006–2008	119
4.9	Formula One team-mates, 1995–1997	119
4.10	Formula One team-mates, 2013–2015	120
4.11	Formula One team-mates, 2002–2004	120
4.12	Formula One team-mates, 2009–2011	121
4.13	Formula One team-mates, 2005–2007	122
4.14	Formula One team-mates, 1998–2000	122
4.15	<i>The Jungle Book</i> , by Rudyard Kipling (1894)	123
4.16	<i>Pride and Prejudice</i> , by Jane Austen (1813)	124
4.17	<i>Strange Case of Dr Jekyll and Mr Hyde</i> , by Robert Louis Stevenson (1886)	125
4.18	<i>Great Expectations</i> , by Charles Dickens (1860-1861)	126
4.19	Graph 693.16	128
4.20	Graph 4037.35	128
4.21	Graph 6028.36	129
4.22	Graph 6952.39	130
4.23	Graph 11669.39	132
4.24	Crew of 1960s & 1970s NASA Space Launches	133
4.25	Skeletal Layout of Aspirin	134
4.26	Skeletal Layout of Citric Acid	134
4.27	Average Number of Nodes and Edges	138
4.28	Average Edge Density	139
4.29	Summary of Time Taken (s)	140
4.30	Time Taken (s) vs Number of Nodes	141

4.31	Time Taken (s) vs Number of Edges	142
4.32	Average Number of Each Type of Pattern	143
4.33	Average Percentage of Patterns Drawn or Discarded	144
4.34	Average Percentage of Nodes in Drawn or Discarded Patterns or in No Pattern	145
4.35	Average Number of Instances of Node-Node Occlusion	146
4.36	Average Number of Instances of Node-Edge Occlusion	147
4.37	Average Number of Edge Crossings	148
5.1	Screenshot of Study Software	153
5.2	Study Room	153
5.3	Data grouped by Method	157
5.4	Data grouped by Task	159
5.5	Data grouped by Size	160
5.6	Mean Time for Method vs Size vs Time	161
6.1	Skeletal Formula of vitamin C (ascorbic acid)	170
6.2	Example of a potential multi-level approach	171
6.3	Suggestion for the definition of a user defined pattern	173
6.4	Potential Force-Directed Placement of Patterns	174
6.5	Example of a potential multi-level approach	175
A.1	Number of Nodes and Edges (Books)	195
A.2	Number of Nodes and Edges (Formula One)	196
A.3	Number of Nodes and Edges (Academy Awards) (1)	197
A.4	Number of Nodes and Edges (Academy Awards) (2)	198
A.5	Number of Nodes and Edges (Rome) (1)	199
A.6	Number of Nodes and Edges (Rome) (2)	200
A.7	Number of Nodes and Edges (Rome) (3)	201
A.8	Number of Nodes and Edges (Rome) (4)	202

A.9	Time Taken (s) vs Number of Nodes By Dataset	204
A.10	Time Taken (s) vs Number of Edges By Dataset	205
A.11	Edge Density (Books)	207
A.12	Edge Density (Formula One)	208
A.13	Edge Density (Academy Awards) (1)	209
A.14	Edge Density (Academy Awards) (2)	210
A.15	Edge Density (Rome) (1)	211
A.16	Edge Density (Rome) (2)	212
A.17	Edge Density (Rome) (3)	213
A.18	Edge Density (Rome) (4)	214
A.19	Size of Largest Clique (Books)	216
A.20	Number of patterns by type (Books)	218
A.21	Number of patterns by type (Formula One)	219
A.22	Number of patterns by type (Academy Awards) (1)	220
A.23	Number of patterns by type (Academy Awards) (2)	221
A.24	Number of patterns by type (Rome) (1)	222
A.25	Number of patterns by type (Rome) (2)	223
A.26	Number of patterns by type (Rome) (3)	224
A.27	Number of patterns by type (Rome) (4)	225
A.28	Percentage of Patterns Drawn & Discarded (Books)	227
A.29	Percentage of Patterns Drawn & Discarded (Formula One)	228
A.30	Percentage of Patterns Drawn & Discarded (Academy Awards) (1)	229
A.31	Percentage of Patterns Drawn & Discarded (Academy Awards) (2)	230
A.32	Percentage of Patterns Drawn & Discarded (Rome) (1)	231
A.33	Percentage of Patterns Drawn & Discarded (Rome) (2)	232
A.34	Percentage of Patterns Drawn & Discarded (Rome) (3)	233
A.35	Percentage of Patterns Drawn & Discarded (Rome) (4)	234

A.36 Percentage of Nodes Drawn, in Discarded Patterns and in No Pattern (Books)	236
A.37 Percentage of Nodes Drawn, in Discarded Patterns and in No Pattern (Formula One)	237
A.38 Percentage of Nodes Drawn, in Discarded Patterns and in No Pattern (Academy Awards) (1)	238
A.39 Percentage of Nodes Drawn, in Discarded Patterns and in No Pattern (Academy Awards) (2)	239
A.40 Percentage of Nodes Drawn, in Discarded Patterns and in No Pattern (Rome) (1)	240
A.41 Percentage of Nodes Drawn, in Discarded Patterns and in No Pattern (Rome) (2)	241
A.42 Percentage of Nodes Drawn, in Discarded Patterns and in No Pattern (Rome) (3)	242
A.43 Percentage of Nodes Drawn, in Discarded Patterns and in No Pattern (Rome) (4)	243
A.44 Instances of Node-Node Occlusion (Books)	245
A.45 Instances of Node-Node Occlusion (Formula One)	246
A.46 Instances of Node-Node Occlusion (Academy Awards)	247
A.47 Instances of Node-Node Occlusion (Rome) (1)	248
A.48 Instances of Node-Node Occlusion (Rome) (2)	249
A.49 Instances of Node-Node Occlusion (Rome) (3)	250
A.50 Instances of Node-Node Occlusion (Rome) (4)	251
A.51 Instances of Node-Edge Occlusion (Books)	253
A.52 Instances of Node-Edge Occlusion (Formula One)	254
A.53 Instances of Node-Edge Occlusion (Academy Awards) (1)	255
A.54 Instances of Node-Edge Occlusion (Academy Awards) (2)	256

A.55 Instances of Node-Edge Occlusion (Rome) (1)	257
A.56 Instances of Node-Edge Occlusion (Rome) (2)	258
A.57 Instances of Node-Edge Occlusion (Rome) (3)	259
A.58 Instances of Node-Edge Occlusion (Rome) (4)	260
A.59 Instances of Edge Crossings (Books)	262
A.60 Instances of Edge Crossings (Formula One)	263
A.61 Instances of Edge Crossings (Academy Awards) (1)	264
A.62 Instances of Edge Crossings (Academy Awards) (2)	265
A.63 Instances of Edge Crossings (Rome) (1)	266
A.64 Instances of Edge Crossings (Rome) (2)	267
A.65 Instances of Edge Crossings (Rome) (3)	268
A.66 Instances of Edge Crossings (Rome) (4)	269
A.67 Time taken (s) (Books)	271
A.68 Time taken (s) (Formula One)	272
A.69 Time taken (s) (Academy Awards) (1)	273
A.70 Time taken (s) (Academy Awards) (2)	274
A.71 Time taken (s) (Rome) (1)	275
A.72 Time taken (s) (Rome) (2)	276
A.73 Time taken (s) (Rome) (3)	277
A.74 Time taken (s) (Rome) (4)	278
B.1 p1 - Practice 1	287
B.2 p2 - Practice 2	287
B.3 p3 - Practice 3	288
B.4 p3a - Practice 4	288
B.5 p4 - Practice 5	289
B.6 p5 - Practice 6	289
B.7 p6 - Practice 7	290

B.8	p7 - Practice 8	290
B.9	t1sf - 1998 Best Leading and Supporting Actor and Actress Nominees	291
B.10	t1sp - 1998 Best Leading and Supporting Actor and Actress Nominees	292
B.11	t1mf - <i>The Jungle Book</i> , by Rudyard Kipling (1894)	293
B.12	t1mp - <i>The Jungle Book</i> , by Rudyard Kipling (1894)	294
B.13	t1lf - <i>A Tale of Two Cities</i> , by Charles Dickens (1859)	295
B.14	t1lp - <i>A Tale of Two Cities</i> , by Charles Dickens (1859)	295
B.15	t2sf - Formula One team-mates, 1997-1999	296
B.16	t2sp - Formula One team-mates, 1997-1999	297
B.17	t2mf - 2000 Best Leading Actor and Actress Nominees	297
B.18	t2mp - 2000 Best Leading Actor and Actress Nominees	298
B.19	t2lf - <i>Jane Eyre</i> , by Charlotte Brontë (1847)	299
B.20	t2lp - <i>Jane Eyre</i> , by Charlotte Brontë (1847)	300
B.21	t3sf - Formula One team-mates, 2009-2011	301
B.22	t3sp - Formula One team-mates, 2009-2011	302
B.23	t3mf - 2002 Best Leading and Supporting Actor and Actress Nominees	303
B.24	t3mp - 2002 Best Leading and Supporting Actor and Actress Nominees	304
B.25	t3lf - Formula One team-mates, 2004-2006	305
B.26	t3lp - Formula One team-mates, 2004-2006	306
B.27	t4sf - Formula One team-mates, 2010-2012	307
B.28	t4sp - Formula One team-mates, 2010-2012	307
B.29	t4mf - Formula One team-mates, 1996-1998	308
B.30	t4mp - Formula One team-mates, 1996-1998	309

B.31 t4lf - 2005 Best Leading and Supporting Actor and Actress Nominees	310
B.32 t4lp - 2005 Best Leading and Supporting Actor and Actress Nominees	311
B.33 Preference Pair 1, Pattern - Formula One team-mates, 2002-2004	322
B.34 Preference Pair 1, Force - Formula One team-mates, 2002-2004	323
B.35 Preference Pair 2, Pattern - Formula One team-mates, 2005-2007	323
B.36 Preference Pair 2, Force - Formula One team-mates, 2005-2007	324
B.37 Preference Pair 3, Pattern - 2007 Best Leading Actor and Actress Nominees	324
B.38 Preference Pair 3, Force - 2007 Best Leading Actor and Actress Nominees	325

List of Algorithms

3.1 Overall Algorithm	34
3.2 Circle identification	42
3.3 Clique and Triangle identification	45
3.4 Star identification	47
3.5 Path identification	48
3.6 Edge Swapping (Share One Edge)	62
3.7 Share One Edge - Circle, Clique and Triangle	65
3.8 Share One Edge - Star	67
3.9 Share Two Edges - Consecutive Shared Edges	70
3.10 Share Two Edges — Non-Consecutive Shared Edges	71
3.11 Node Swapping (Share One Node)	74
3.12 Share One Node - Circle, Clique, Triangle, Star (Outside)	77
3.13 Share One Node - Star (Centre Node)	79
3.14 Share Two Nodes - Circle	81
3.15 Share Two Nodes - Triangles	82
3.16 Node Swapping (Connected By an Edge)	84
3.17 Connected By ≥ 1 Edges - Clique, Circle, Triangle, Star	87
3.18 No connections	91
3.19 Path and Route Adjustment	93
3.20 Loose Nodes Identification	96

3.21 Drawing Loose Nodes (Multiples)	97
3.22 Drawing Loose Nodes (Singles)	98
3.23 Join Drawing Areas	102
3.24 Find Best Drawing Areas	103
3.25 Symmetry Score	106
3.26 Occlusion and Edge Crossing Score	108

Chapter 1

Introduction

Graphs can be used to represent various forms of data. For these graphs to be of benefit to a user, it is important that they are drawn in an aesthetically pleasing manner. The process of creating a visual representation of a graph is known as *graph drawing*. This work proposes a new method of drawing undirected graphs using straight edges by identifying subgraphs (patterns) and drawing those in a consistent manner. An empirical study was also conducted in order to answer the research question: “**Is a pattern based layout more effective than a force based layout?**” and the results of this are analysed and discussed.

There are several processes involved with this method: identifying patterns, determining a drawing order, and drawing each pattern. Although there has been considerable research in the field of graph drawing (See Chapter 2), this method is a new approach to the problem. This thesis provides an introduction to graph drawing and existing techniques before describing the new method in detail in Chapter 3. An examination of the effectiveness of the new method will follow in Chapter 4, and an empirical study will be described in Chapter 5. The results of this study will be analysed and and conclusions made.

1.1 History of Graph Problems and Graph Drawing

Graph problems have existed for many centuries. The first such problem was considered by Leonhard Euler in 1736 when he attempted to find a path through the city of Königsberg by crossing each of the 7 bridges once and only once [44]. By imaging the two islands and the two river banks as nodes and the bridges as edges, a more general problem was developed. Euler proved that no path could be found as some of the nodes had an odd number of edges.

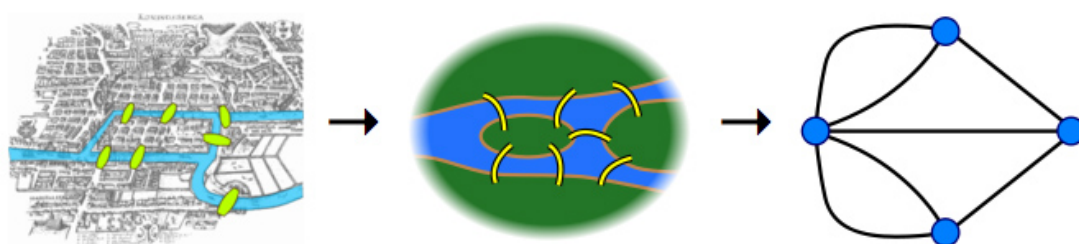


Figure 1.1: Euler's Seven Bridges of Königsberg

Graph theory further developed with a textbook by Dénes Kőnig in 1936 [71] and by Frank Harary in 1969 [52]. These were both considered to be seminal pieces in the field [121]. Most graph problems are of a theoretical nature and, until the development of computer technology, several (most notably the Four Colour Problem¹) remained unsolved. An introductory summary of graph theory is provided by Chartrand [24].

Graph drawing was started by William Tutte who in 1963 created a pioneering barycentre approach [120] (See Section 2.1.1). This was the first graph drawing method and paved the way for many further developments, such as Peter Eades' 1984 spring embedder [38]. There have been numerous developments in the field of graph drawing since then, and a number of these are described in

¹Separating a plane into contiguous regions (a map), no more than four colours are needed to colour each region so that no two adjacent regions share the same colour.

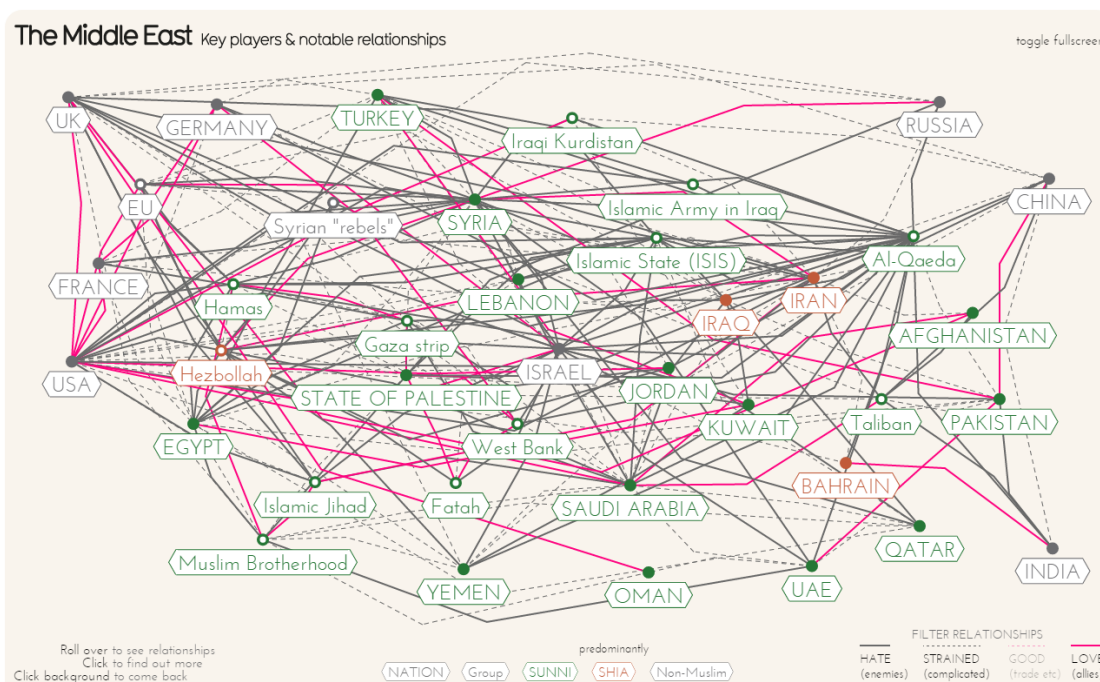


Figure 1.2: Key Players and Notable Relationships in the Middle East [81]

Chapter 2.

However, graph drawing has not just remained in the field of academia; it has begun to appear in mass media too. For example, a visualization (Figure 1.2 [81]) appeared on a number of news websites detailing the complicated relationships between several of the interested parties in conflicts in the Middle East. Also, a popular website [23] was developed to allow users to discover new artists they may enjoy by drawing graphs of their current favourite artists. In the example of Figure 1.3, if a user were to search for Frank Sinatra, Dean Martin and Sammy Davis Jr., they would discover other similar artists, such as Nat King Cole and Bing Crosby.

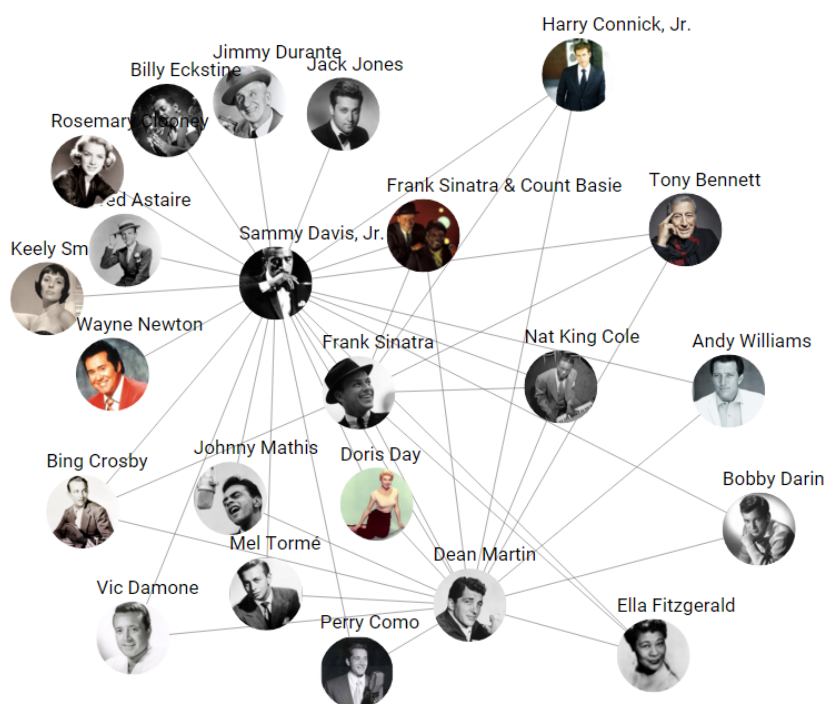


Figure 1.3: MusicRoamer: Suggested artists for some of the Rat Pack [23]

1.2 Why Patterns?

Despite there being a large number of different graph drawing techniques, none so far have identified patterns within a graph and drawn them in a consistent manner. This section will describe the reasoning behind choosing to identify and draw patterns.

The patterns used in this research are all drawn as regular polygons wherever possible (see Chapter 3 for details of the implementation). Most children learn to recognise basic shapes such as circles, triangles and rectangles before they start school [27] and thus regular polygons should be familiar to most of the population. For example, when processing an image, humans segment that image into simple geometric shapes [14]. It is therefore sensible to attempt to draw subgraphs in a manner that is familiar to many: using regular shapes.

Although regular shapes are familiar and common, that does not necessarily mean they improve understanding of a graph. Dunne and Shneiderman [36] found that highlighting particular subgraphs “can help biologists spot the locations of particular processes, but does little to reduce the clutter of a complex network visualization and can even reduce readability”. However, they did note that “the spatial layout of a node-link diagram can have a profound impact on the detection of communities in the network and the perceived importance of actors”. McGrath et al. [83] concur with this by discovering that “the number of perceived groups changes when nodes are spatially clustered to hide or highlight a clique”. This shows that a user’s perception of the underlying data may be affected by the emphasis of features or groups within the layout, even if the graph is not drawn to emphasise such features.

1.3 Summary of Chapters

This thesis is divided into a number of chapters:

Chapter 2 This chapter details existing work in the fields of graph drawing, graph aesthetics and subgraph isomorphism. Various graph drawing methods are described in detail, with comparisons between them identified. An in-depth comparison of this work and that of Dunne and Schniederermann [36] is also provided. Detailed analysis of previous work in many areas of graph aesthetics are considered. This enables the discussion of what makes a graph layout effective, and justifies the decisions taken in the implementation and the design of the empirical study.

Chapter 3 The research is described in detail in this chapter. Patterns will be defined, before their identification is explained. The determination of the

drawing order follows, before detailed explanations of the many drawing algorithms. Examples and pseudocode for each drawing type are provided.

Chapter 4 An analysis of generated layouts is performed in this chapter. An explanation of various datasets begins the chapter, before examples from various datasets drawn with this pattern based system and a force directed layout are compared for interesting visual artefacts. Following this, various metrics are used to quantitatively compare the pattern based and force directed layouts.

Chapter 5 This chapter describes the formal empirical study, conducted in order to answer the research question: “**Is a pattern based layout more effective than a force based layout?**”. The methodology will be detailed before the results are discussed and analysed. Any potential threats to validity are also detailed.

Chapter 6 Conclusions from the research are present in this chapter. The research is summarised and the effectiveness is discussed. Potential avenues for future work are also discussed.

Chapter 2

Related Work

There has been much research in both graph drawing, aesthetics and subgraph isomorphism (the process of identifying a subgraph within another graph). In this chapter, related work in graph drawing and subgraph isomorphism is discussed. Previous work on graph aesthetics is also discussed as this work can be used as a basis for implementing and evaluating the results of a particular drawing method.

2.1 Graph Drawing Techniques

Graph drawing is a well researched area of data visualization. There are a number of existing reviews and surveys of current drawing techniques. The *Handbook of Graph Drawing and Visualization*, edited by Tamassia [115] in 2013, details a large number of algorithms and research on a broad range of topics. Gibson et al. [51] surveyed graph layout techniques, but only detailed methods for the drawing of graphs in two-dimensions. Di Battista et al. [34] also described a large number of techniques, including examples, in their survey. They did

not focus on one particular area of graph drawing, including both application-oriented and theoretical work, however the age of this research (having been published in 1994) does mean a number of more recent techniques are missing. Díaz et al. [35] also compiled a review of literature in the field of graph drawing, but heavily focus on the algorithms involved rather than any empirical work. Herman et al. [56] created a review of methods for visualising graphs. Unlike other work, this survey is only focussed on “structured data (i.e. where graphs are the fundamental structural representation of the data)”. They also focus on more particular issues that are relevant for visualisations, such as usability and aesthetics. Kobourov [69] produced a thorough review of a number of graph drawing techniques that use force directed methods. This starts with Tutte’s method [120] and continues with descriptions of methods up to the present day. Von Landesberg et al. [125] and Hu [60] both created reviews of drawing methods with particular emphasis on techniques that are designed or optimised for large graphs. In addition to these reviews, Chen [25], Spence [110] and Ware [127] have all written detailed books on the subject of Information Visualisation, of which graph drawing is a part.

There are a number of existing methods for drawing graphs, and these include the barycentre approach, force-directed methods, simulated annealing, constraint based layouts and others. Many techniques for drawing graphs optimise for a particular domain and restricted graph type, such as planar graphs, however the methods detailed below are for drawing graphs with fewer restrictions.

2.1.1 Barycentre Approach

Tutte was the first to propose a method for drawing graphs in 1963 [120]. The method first creates a boundary polygon before using a system of linear equations to place nodes at the barycentre of their neighbouring nodes. Tutte proves that this method can draw a crossing-free, straight line drawing for any 3-node connected planar graph. This automatic layout is a big advantage, however it does limit the type of graphs that can be drawn to planar graphs and is therefore not suitable for drawing more general graphs. Some nodes must also be fixed in location before drawing can begin. There is also the advantage of having a global solution, rather than the potential to get stuck in local optima, as can be the case with force-directed methods.

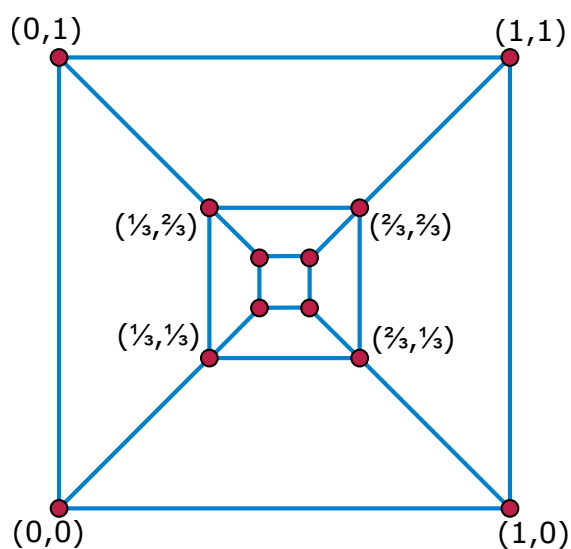


Figure 2.1: A graph drawn using Tutte's Barycentre approach

2.1.2 Force-directed methods

Many graph drawing techniques are based on force-directed methods. This approach was developed by Eades [38], and models nodes as charged particles

which have a repulsive force against all other nodes. Edges are represented as springs charged with attractive forces between the nodes they connect, and both these forces relate to the distance between the respective nodes. In Eades's model, the attraction force is $k \log d$, and the repulsive force is $\frac{k}{d^2}$, where k is an experimentally found constant and d is the distance. These forces move their respective nodes and when they reach an equilibrium, or a certain number of iterations have been completed, a final layout is created.

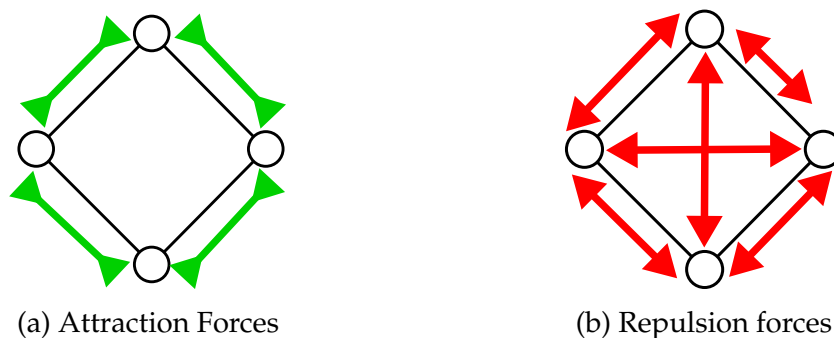


Figure 2.2: Example of the forces used in Peter Eades's method [38]

Graphs drawn using this technique often have aesthetically pleasing layouts (and symmetrical layouts [40]), but there can be some issues such as occlusion and local optima. On particularly dense graphs, occlusion can be problematic and force-directed methods often struggle to overcome local optima. This means that a node cannot move to a less favourable position before eventually reaching a better position. For example, as Figure 2.3a demonstrates, nodes 2 and 3 should be swapped, but the repulsive forces acting on each node prevent them from moving closer together.

While Eades's original principle is still used, it has been modified and improved. Fruchterman and Reingold [48] introduced improvements such as simplifying the calculations of the forces, as well as limiting the repulsive forces to

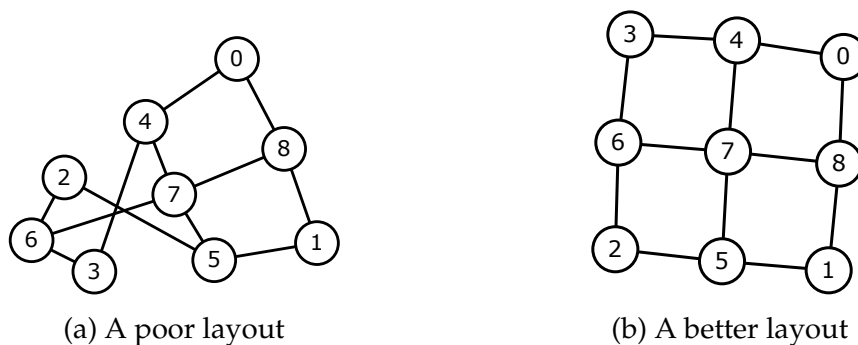


Figure 2.3: Example of Local Optima

act only on nodes nearby, rather than the entire graph. In the simplified calculations, attraction is linearly proportional to the distance (following Hooke’s Law¹) whereas repulsion has a linear inverse proportion to the distance. This differs to Eades’ work where computationally expensive logarithmic calculations are used.

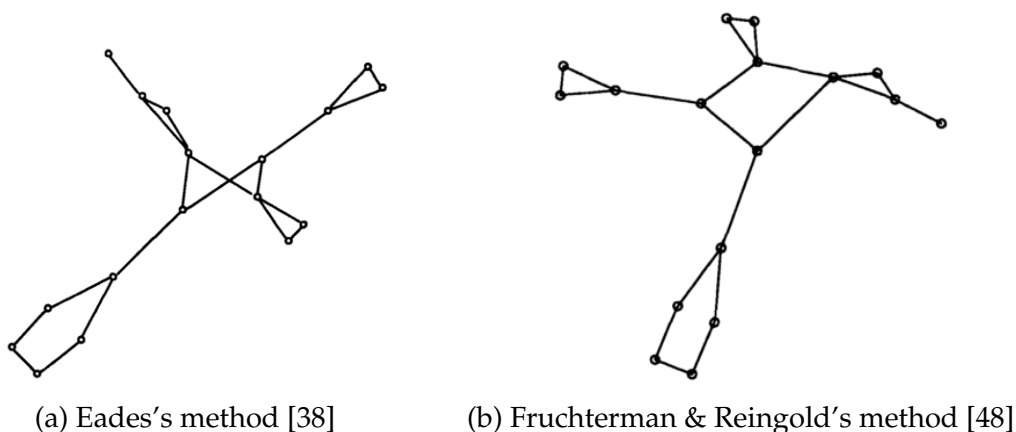


Figure 2.4: Example of the same graph drawn using two different force-directed methods (Taken from Figures 75 and 76 in Fruchterman and Reingold [48])

¹Hooke’s Law states that: “The force needed to stretch a spring is proportional to the extension of the spring from its natural length” [20]. Eades used his own formulae to calculate the spring forces, whereas Kamada and Kawai maintained a closer relationship to Hooke’s Law.

Kamada and Kawai [68] use partial differential equations to achieve an optimisation of this, as well as introducing the concept of an ideal distance between two non-connected nodes. Harel and Koren [54] adapted Kamada and Kawai's technique to approximate the final layout and allow nodes to deviate from this approximation by a decreasing amount at each iteration. Gansner et al [50] modified Kamada and Kawai's technique by changing the energy function using a process known as stress majorisation. Sugiyama and Misue [113] additionally apply magnetic forces to nodes to attempt to orientate edges in a particular manner. Lin and Yen [76] also developed a modification to the basic force-directed technique where edges also repel other nearby edges. This avoids edge occlusion, but unfortunately results in node occlusion instead. Lipp et al. [77] created an algorithm that draws large graphs in $O(V \log V + E)$ time complexity rather than $O(V^2)$ time complexity taken by Fruchterman and Reingold [48]. To achieve this, nodes are grouped and the barycentre of this group is used in the force calculations for all nodes within the group. This enables the algorithm to perform faster "without a significant influence on the quality of the drawings (in terms of the number of crossings and deviation in edge lengths)".

Hierarchical approaches based on force-directed methods have also been developed. For example, Walshaw [126] suggested to cluster nodes together to form smaller graphs. The smaller graphs are drawn with a force-directed method and then expanded. Additional nodes are then drawn using a force-directed method and the graph expanded again. This continues until the graph has returned to the original topology. Barnes and Hut [9] created a hierarchical approach that ran in $\mathcal{O}(V \log V)$ time (where V represents the set of nodes), which was extended by Hu [59]. Gajer et al. [49] also developed a hierarchical force-directed approach, but draw the graph in many dimensions, before projecting to 2 or 3 dimensions. Arleo et al [7] developed a system to draw graphs

using a multi-level force-directed technique, but with the algorithm running over a distributed system. This method is able to draw graphs with a million edges in around an hour.

It is also possible to cluster edges as Holten and van Wijk [58] discovered. Through bundling edges that are similar, it is possible to clearly identify common connections. Tunkelang [119] created a version of a force directed layout based on more complex physical systems (such as those in astrophysics) and through numerical optimisation. Coleman and Parker [28] combined Davidson and Harel's [32] method with Fruchterman and Reingold's [48] method to create a new layout mechanism that, depending on the domain, drew graphs according to a number of aesthetics, for example avoiding nodes placed too close to edges, and maintaining consistent edge lengths.

Force-directed methods can also be combined with other visualization techniques. For example, combining force-directed methods and Euler diagrams [8, 104] enables an aesthetically pleasing graph to be drawn, as well as highlighting groupings between nodes. In this method, the forces are similar to those in Fruchterman and Reingold's work [48]. However, an extra repulsive force is applied which is proportional to the distance a node is from the boundary of its group. This ensures that nodes remain in their correct groups and do not cluster on the edges.

Further analysis on a collection of graph drawing algorithms was undertaken by Brandenburg et al. [19], where a variety of different drawing methods are compared.

2.1.3 Simulated annealing

Although force directed methods can sometimes produce aesthetically pleasing graph layouts, they are susceptible to local optima. A local optimum is where

an algorithm cannot move to a less favourable result in order to progress to an overall better result. Therefore, force-directed methods may not always generate the best possible graph layout. Simulated annealing, however, is not so severely affected by local optima.

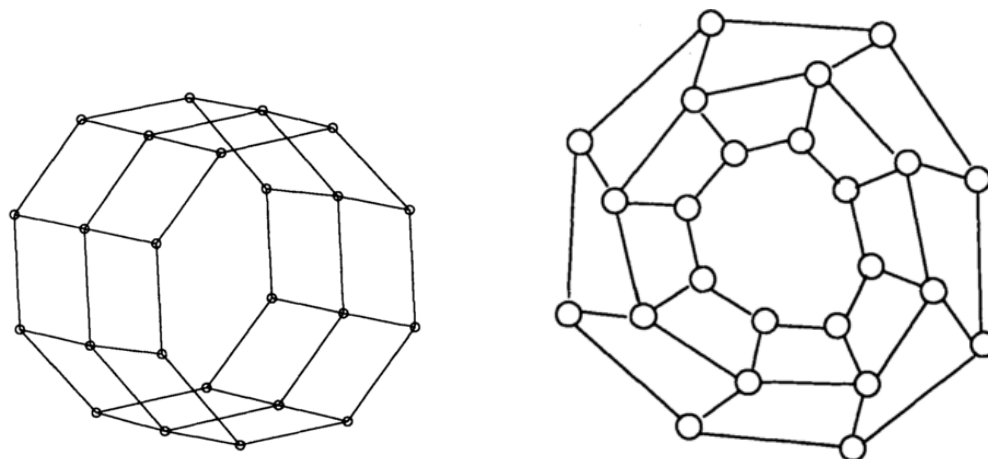
Simulated annealing is analogous to annealing in metallurgy, where metals are cooled slowly to improve their structural layout. Similarly, in simulated annealing the graph initially moves large distances which are then “cooled” slowly to allow smaller movements. This helps the system to avoid local optima. Davidson and Harel [32] used this method to draw graphs and their results compared favourably with manually drawn graphs. However, simulated annealing is computationally more expensive than force-directed methods, and in order to compensate Davidson and Harel allow their algorithm to find solutions that are only close to optimal. Although this does not produce the best possible layouts, the results are aesthetically pleasing. Simulated annealing is also used in other fields, such as materials science [10], map generalization [129] and power generation [134].

2.1.4 Other Drawing Techniques

2.1.4.1 Constraint-based Layout

Force-directed methods can also be used in conjunction with other constraints. Tamassia [114], for example, suggested that constraints such as pinning nodes so they cannot move, clustering nodes, and the alignment of edges could be used in conjunction with force-directed methods.

Tamassia implemented these constraints by adding new forces, or by creating “dummy” nodes that effect forces as any other node would, but do not appear in the final layout. Tamassia concluded that force-based methods lend



(a) Fruchterman & Reingold's force-directed method [48]

(b) Davidson and Harel's simulated annealing method [32]

Figure 2.5: Example of the same graph drawn using a force-directed method and simulated annealing (Taken from Figures 54 and 55 in Fruchterman and Reingold [48])

themselves well to additional constraints. This allows the use of force-directed methods for a number of tasks with only slight modifications and extra constraints.

Böhringer and Paulisch [16] also modified a number of existing techniques to allow a user to specify constraints in order to achieve a better layout.

2.1.4.2 Incremental Technique

Tunkelang [118] developed a new drawing technique based on an incremental search. In this technique, nodes are ordered based on a minimal height spanning tree. For each node placing, 16 locations are checked and for each one an aesthetic cost is calculated. This cost is based on attraction and repulsion forces as well as the number of edge crossings and edge lengths. The location with the lowest cost is then chosen for the node placement. After each individual node has been inserted the layout is improved. This is done by checking the cost of

neighbouring nodes.

Tunkelang compared this algorithm with both Fruchterman and Reingold's method [48] and Davidson and Harel's method [32] on three quality measures:

- Edge lengths
- Node distribution
- Number of edge crossings

The algorithm performed favourably on both sparse and dense graphs, although the other methods are not optimised for these test conditions. Computational time analysis was not performed.

2.1.4.3 Graph Embedder

A method known as the Graph Embedder was proposed by Frick et al [46]. This method utilises cooling methods used in previous work [32, 48] but instead of applying a temperature to the whole graph, it applies a different temperature to each node.

The temperature calculated for each node reflects the temperature of previous iterations as well as the possibility of an oscillating or rotating node. If the algorithm detects that the node is a larger distance from its intended destination, the temperature can rise. Force is also applied to the node to ensure that it is attracted to the centre of the graph; this helps to minimise the surface area of the graph.

Although this method successfully resolves foldings and minimizes edge crossings, local optima can result in drawings with poor aesthetics. However, on smaller graphs this method performs marginally better than that proposed by Fruchterman and Reingold [48], and Davidson and Harel [32]. This is not true for larger graphs where this method results in visually poorer layouts.

2.1.4.4 Other Drawing Methods

Hobbs and Rodgers [57] created a genetic algorithm that drew a graph based on a number of aesthetic criteria. Utech et al. [123] also used an evolutionary algorithm to draw a directed graph, by layering and ordering nodes before drawing them in turn.

Archambault et al [5, 6] created a multi-level drawing algorithm that identifies topological features within a graph, and condenses them into a single node. This continues until all features have been found, when each feature is then drawn in an appropriate manner. Unlike the pattern based system described in Chapter 3, this is multi-level and does not identify features on a single level.

2.2 Graph Aesthetics and Perceptual Theories

There has been considerable research into graph aesthetics and perceptual theories in order to identify the features that improve a graph's usability and aesthetics.

There are a number of standard measures used to quantitatively measure the aesthetics of a graph. Eades et al [41] suggested a number of measures, as did Tamassia et al [116] who also proposed centrally placing high degree nodes (Figure 2.6). Further work on graph aesthetics was performed by Purchase [91, 97, 101] and Herman et al [56].

There are a number of perceptual theories that are the basis for aesthetic principles. The Gestalt principle of proximity suggests that "objects that are close to one another appear to form groups". This means that similar nodes should be clustered (so they appear as one group, but not so that they overlap), that other edge lengths should be maximised (so that unrelated nodes do not appear too close) and that nodes should not overlap [133]. Edge crossings



Figure 2.6: Good and Bad Examples of Centrally Placing High Degree Nodes

should be minimised (so that each edge can easily be identified) based on the same principle, although one method [106] uses the Gestalt principle of closure to place gaps where edges overlap, yet still maintain the usability of the visualisation. Keeping the edge lengths uniform helps with the principle of similarity, where stimuli that resemble each other are perceived as part of the same group [133]. This principle also suggests that symmetry and clustering similar nodes are important aesthetics. The physical limitation of human eyes is such that edge angles should be maximised and nodes should not be located too close together [31].

Bennett et al [13] described a large number of aesthetic principles and a modified version of their table (to remove unnecessary columns and rows inapplicable to general undirected graphs) is shown as Table 2.1. The authors who proposed and evaluated various metrics are detailed, along with an example of each. Only those metrics that are relevant to drawings of general undirected straight-line graphs are displayed.

Heuristic	Proposed By	Evaluated By	Example
<i>Node metrics</i>			
Cluster similar nodes	[116, 117]	[65]	Fig. 2.7
Distribute nodes evenly	[32, 53, 116, 117]	—	Fig. 2.8
Node-edge occlusion	[32, 53]	—	Fig. 2.9
Node-node occlusion	[132]	—	Fig. 2.10
<i>Edge metrics</i>			
Minimise edge crossings	[15, 32, 53, 116, 117]	[65, 96, 98, 128]	Fig. 2.11
Keep edge lengths uniform	[15, 32, 117]	—	Fig. 2.12
Minimise edge length	[53, 116, 117]	—	Fig. 2.13 & 2.14
Maximise edge angles	[31, 101, 117]	—	Fig. 2.15
<i>Overall Layout Metrics</i>			
Keep correct aspect ratio	[117]	—	Fig. 2.16
Minimise area	[116, 117]	[96]	Fig. 2.17
Maximise symmetry	[15, 101, 116, 117]	[96, 100] (local)	Fig. 2.18

Table 2.1: A summary of various graph drawing aesthetics (Modified from Bennett et al [13])

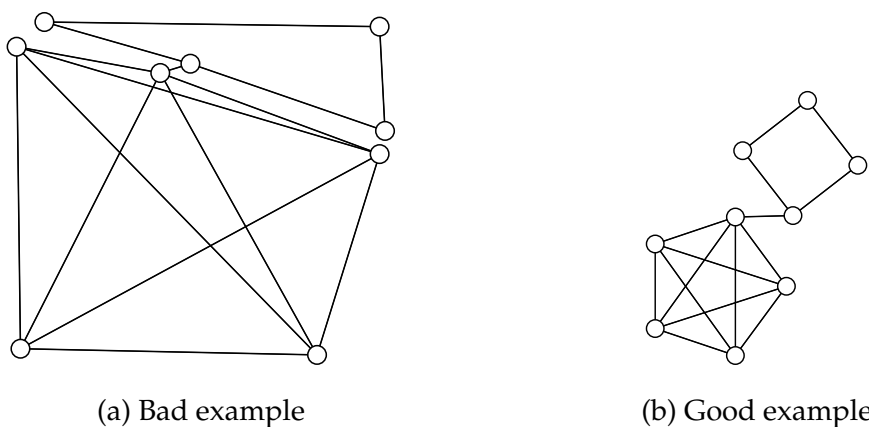


Figure 2.7: Good and bad examples of clustering similar nodes

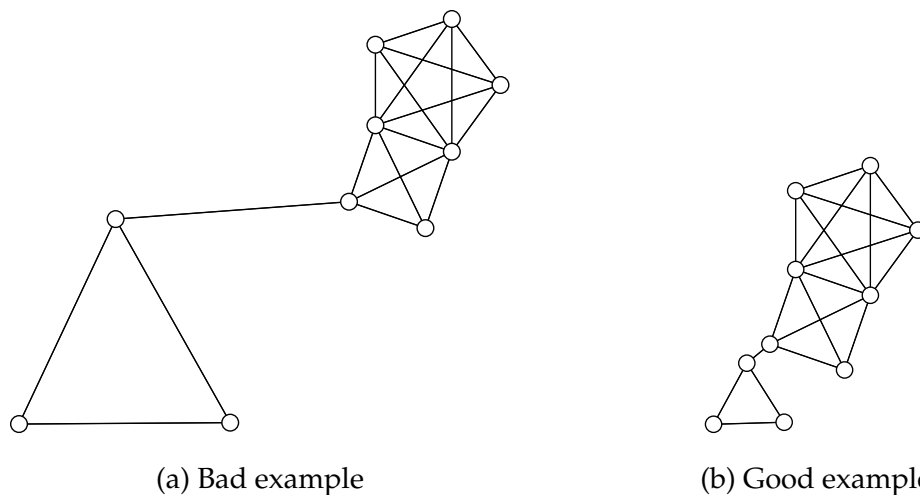


Figure 2.8: Good and bad examples of distributing nodes evenly



Figure 2.9: Good and bad examples of node-edge occlusion



Figure 2.10: Good and bad examples of node-node occlusion



Figure 2.11: Good and bad examples of minimising edge crossings



Figure 2.12: Good and bad examples of minimising edge variance

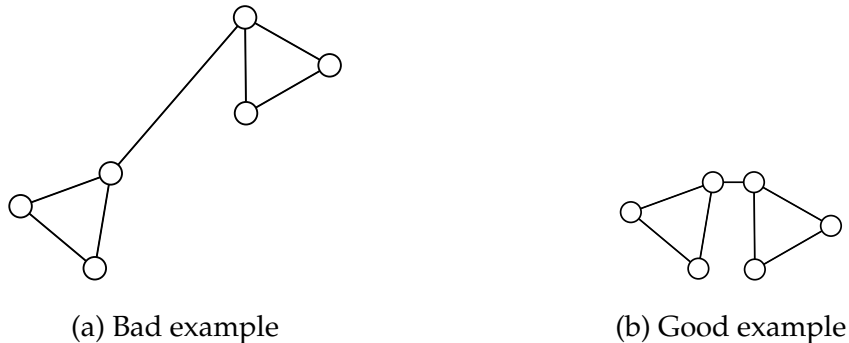


Figure 2.13: Good and bad examples of minimising the length of the largest edge

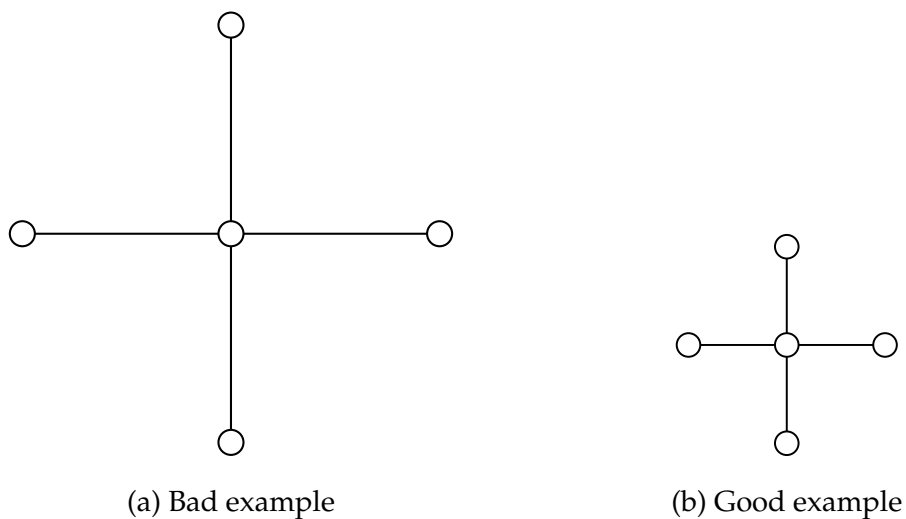


Figure 2.14: Good and bad examples of minimising the total length of edges



Figure 2.15: Good and bad examples of maximising the minimum angle of edges leaving a node

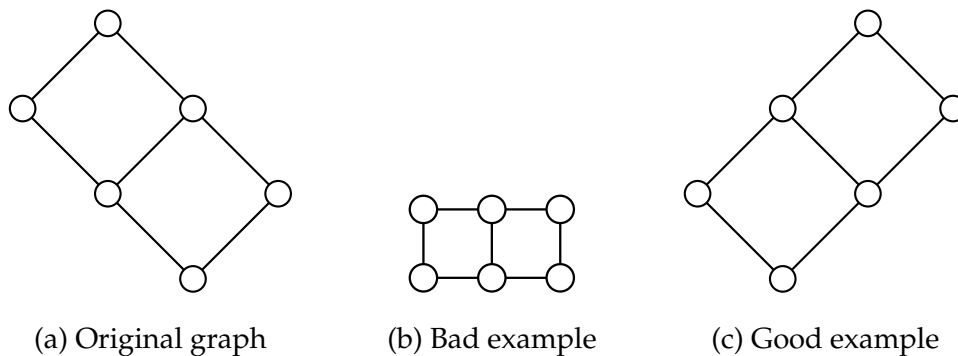


Figure 2.16: Good and bad examples of consistent aspect ratio

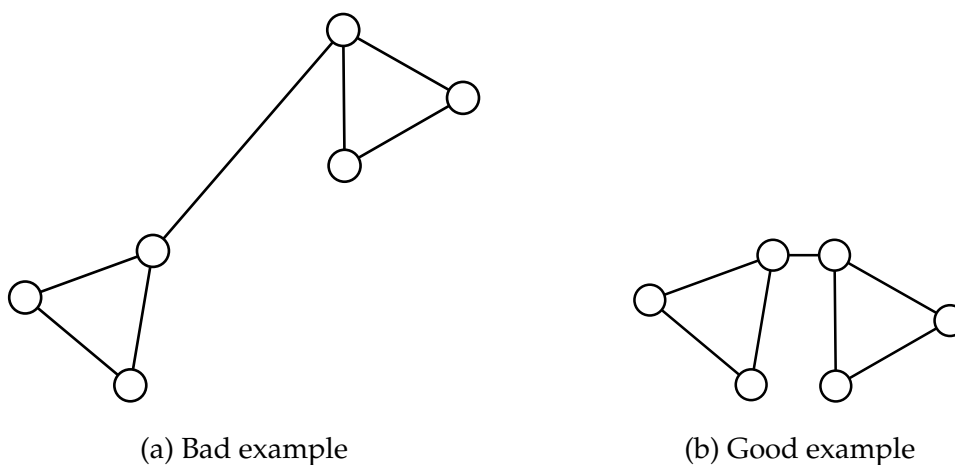


Figure 2.17: Good and bad examples of minimising area

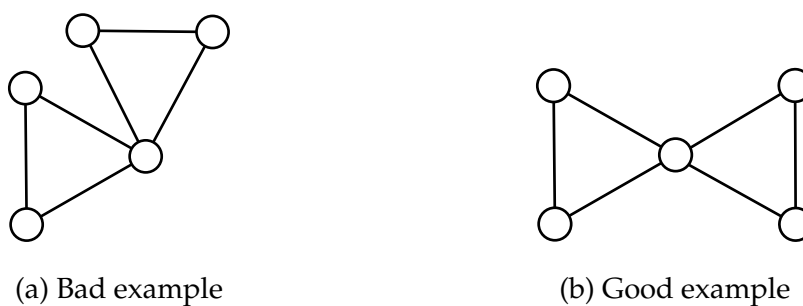


Figure 2.18: Good and bad examples of maximising symmetry

Formal methods to calculate these metrics were then developed. However, it is made clear that these only focus on “good” numbers, not on how useful humans perceive these metrics. Purchase [92] performed a number of empirical studies on various graph layout mechanisms to determine which performs best with users. Purchase [93] continues to discuss the virtue of such measures stating “designers ... can benefit from ... this human-centred approach, by adapting their methods to focus on user concerns, rather than computational ones”. By taking a human-centric view of graph aesthetics, graph drawing methods can be optimised for user understanding, rather than just low scores in metrics. McGrath and Blythe [82] agree, adding that “visualization strategies must take into account how viewers are likely to decode the information presented”. It is, however, then necessary to perform user studies on the results. Additionally, Purchase [95] also discovered that these current aesthetic measures are not necessarily relevant for certain domains, for example UML diagrams (or, as Storrie [112] discovered, a combination of aesthetic measures may be required): it is possible that a graph layout may never be used by a human (for example in PCB manufacture [45, 103]). It is indeed possible that users may create better layouts than the best automated layout mechanisms [37]. There are also aesthetics to consider when creating dynamic visualisations. Bender-deMoll and McFarland [12] discussed a number of issues with dynamic visualisations, one of which is preserving the *mental map* [39, 99, 102]. The *mental map* is the concept that the user gains insight and information from a diagram and that sudden and large changes can destroy this [39].

While it is accepted that the measures above are useful in improving the usability of a graph, there has also been work to demonstrate this. Huang [62] performed eye tracking analysis on users in order to determine the effect of edge crossing on the understanding of a graph. Huang found that increasing the

angle at which edges cross will result in an improved layout. It is also important to study the cognitive load caused by performing a task on particular graph drawing layout. Huang et al [63, 64, 66] performed a number of studies in order to determine the effect of cognitive load on a user. While it is useful that the graph may appear “nice looking”, if the layout results in extra work from the user, it may not be an effective solution.

There are several metrics to quantitatively measure the aesthetics of a graph, however it is important that the performance of users in empirical studies is also considered. As Bennett et al [13] eloquently concluded: “creating aesthetically appealing graphs is more than a quest for the beautiful – it has the practical aim of revealing underlying meaning and structure”.

2.3 Subgraph Isomorphism

Subgraph isomorphism is a computational problem where one must decide if a subgraph can be found within a larger graph. More formally, it is determining whether a given graph, G , has a subgraph, g , that is isomorphic to another given graph, H . If and only if an edge exists between two nodes in g , then there must also be an edge existing between the corresponding two nodes in H . The detection of subgraph isomorphism is an NP-complete problem, however approximate subgraph isomorphism can be completed in polynomial time [33]. There has been considerable research conducted on subgraph isomorphism, including detecting subgraphs, and drawing large graphs which are based on previously defined or drawn subgraphs.

Yuan et al [135] developed a system in which subgraphs were drawn according to user input. In this system a human or non-human user defines and manually draws a subgraph.

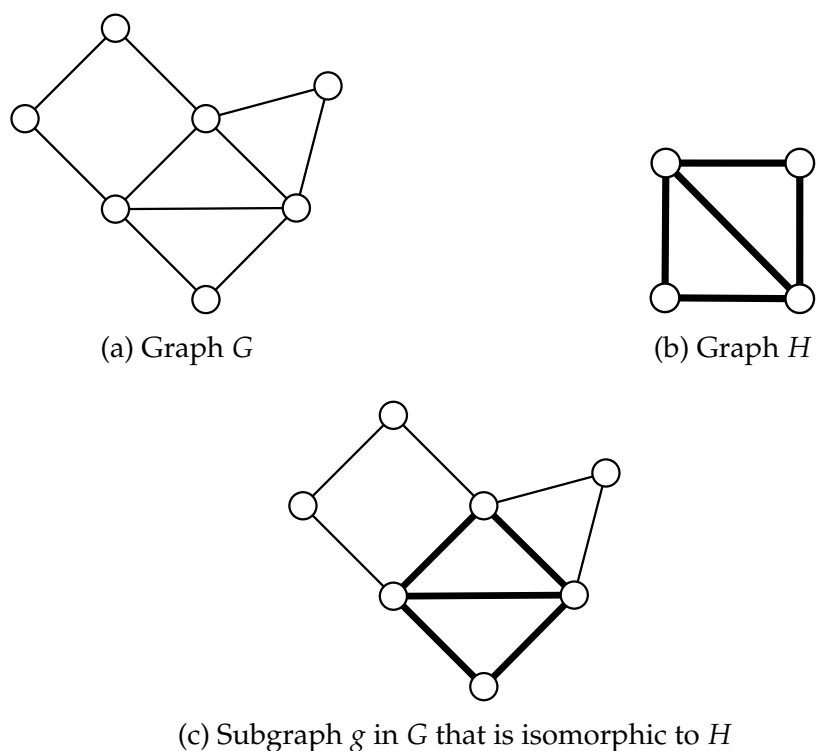


Figure 2.19: Example of subgraph isomorphism

The system then creates a layout for the rest of the graph. This layout is based around the subgraph and modifications to the user’s layout are kept to a minimum. This allows subgraphs to be drawn manually by humans (which often produces good layouts for small, sparse graphs) or by specific drawing algorithms (i.e. tree drawing algorithms). Despite the algorithm’s success, the “garbage in, garbage out” problem cannot be overlooked in that a poor layout from the user can result in a poor overall layout.

Another algorithm, developed by Ullmann [122], detects subgraph isomorphism. This algorithm uses adjacency matrices for the graph and subgraphs. It uses these matrices to detect isomorphism. Čibej and Mihelič [26] investigated the effectiveness of Ullmann’s technique on large datasets.

To overcome such issues as computational complexity, Messmer and Bunke

[84, 86] developed an algorithm that runs in quadratic time. This model uses a database to detect potential subgraphs and an arbitrary graph containing one or more of these subgraphs. Through this database, all potential subgraphs are converted into adjacency matrices and organised into a decision tree.

Messmer and Bunke's algorithm has a lower time complexity in comparison with Ullmann's model, but it is possible that the decision tree's size may grow exponentially with the size of the graphs. The authors propose a solution to this potential issue, but this comes at the cost of increased running time.

Messmer and Bunke created another method [85, 87] that decomposes graphs into smaller subgraphs. Once a set of small subgraphs has been identified, the main graph is examined to try to locate these subgraphs within it. If they do, one instance is noted and the higher level of graph is then compared. If all levels have been compared and exist, then the subgraph exists in the main graph. As each subgraph is only matched once, the complexity of the algorithm is limited. Performance rates for this algorithm are high but in some cases it can fail to deal with highly connected graphs.

Another model, developed by Huan et al [61], successfully identifies connected subgraphs within a larger graph. In this method, subgraphs are joined and extended with potential subgraphs within the main graph. In comparison with other isomorphism techniques, this system seems to perform favourably.

Cordella et al [30] implemented algorithms that test both subgraph and graph isomorphism with an emphasis on larger graphs. This algorithm's time complexity is independent of the number of nodes and performs favourably in comparison with other methods.

An algorithm to detect subgraph isomorphism in planar graphs has been developed by Eppstein [43]. Eppstein developed an algorithm that detects isomorphism in linear time by finding subgraphs in a tree from each possible node.

However, not all graphs are planar, so this method cannot be applied as a general solution.

Bonnici et al [17] developed a subgraph isomorphism test utilising a search based strategy, specifically to identify subgraphs in biochemical data. As with other methods, the authors use a tree based search approach to detect isomorphism. If there is a statistically significant increase in the number of subgraphs in a particular dataset compared to random graphs, then this set is of interest to the user. These subgraphs are known as motifs and are discussed further in Section 2.3.1.

Lambert et al [73] developed a method to identify patterns within graphs and highlight them, as well as identify overlaps between these patterns. The authors, however, make no attempt to draw the graph using these discovered patterns. Lischka and Karl [78] developed a method to draw a graph using subgraph isomorphism and various constraints, using a backtracking algorithm. Maier and Minas [80] developed an editing tool for visual languages that detects various reusable patterns and allows the user to specify a layout for these.

Subgraph isomorphism has been used in various fields of research aside from graph drawing and visualisation, as well as in a number of real world applications. Such examples include: identifying molecular structures [105], Chinese character recognition [79], interpreting schematic diagrams [22], semantic networks with graph grammars [42], seal verification [74], and image processing [11].

2.3.1 Motifs

There is a large interest in the identification and drawing of motifs. Motifs are subgraphs which appear within a large graph more often than random probability would suggest. The identification of motifs is relevant to both subgraph

isomorphism and this work. However, most motif research has been in the biological domain, with many implementations either simply highlighting the subgraphs or replacing them with a glyph².

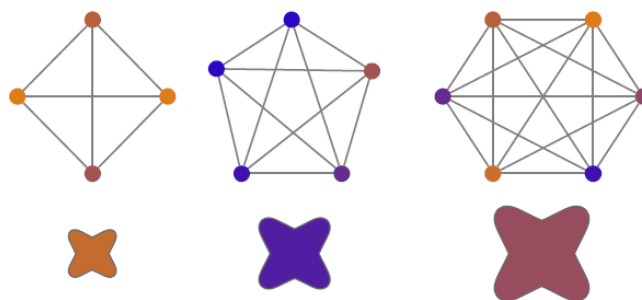
Motif detection can be a useful tool to identify interesting subgraphs in complex biological networks [88, 107, 108, 131], such as amino acid mutation graphs [75], metabolic networks [72] and E. coli networks [109]. Vehlow et al. [124] created a method to highlight motifs within a graph, but did not draw these subgraphs. Huan et al. [61] developed a novel method for the identification of motifs. Koenig et al. [70] developed a method that identifies user defined subgraphs within a main graph, but it redraws the graph to fit within the user defined subgraph layout. A summary of a number of algorithms designed for “frequent subgraph” (i.e. motif) detection are described by Jiang et al [67].

2.3.1.1 Motif Simplification

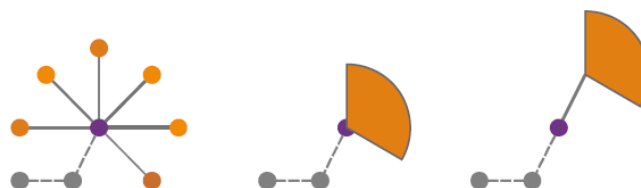
Dunne and Schneidermann [36] developed a method called *motif simplification* which identifies motifs and replaces these with glyphs in the final layout. They claim that using glyphs to replace motifs can require less screen space, are easier to understand in the context of the network, can reveal otherwise hidden relationships and preserve as much underlying information as possible. The authors consider three types of glyphs: cliques, fans (similar to stars in this work) and connectors. The glyphs grow larger as the number of nodes in the respective motif increases (see Figure 2.20).

Dunne and Schneidermann discuss a number of problems with this approach. One is that overlapping glyphs can be ambiguous and choosing one clique over another to prioritise in the drawing can impact the user’s perception of the

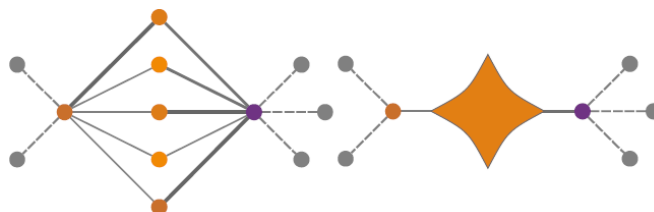
²A symbolic representation of a group of nodes, e.g. Dunne and Schneidermann [36] replace cliques with a petal shaped glyph



(a) Cliques of size 4, 5 and 6 and their glyphs (Taken from Figure 6)



(b) Fan motif and two glyph variants (Taken from Figure 4)



(c) Connector motif and glyph (Taken from Figure 5)

Figure 2.20: Example of various motifs and glyphs (Taken from Figures 4–6 in Dunne & Schneidermann [36])

graph. Instead, the largest non-overlapping clique is chosen, which is a similar technique to that used in the pattern based system described in this thesis.

One of the main differences between this work and that of Dunne and Schneidermann is that in this pattern based system, the underlying data is still always displayed. Users can still see the individual members of cliques, stars, etc as well as their full relationships.

There are also potential issues regarding a user judging the size of motifs and thus the number of nodes in the particular motif. Consider Figure 2.21a, where

two glyphs are shown. It is not clear whether these represent, for example, motifs of size 4 and 5, or 6 and 12. In the pattern based system, however, it is immediately obvious how many nodes exist within a pattern (see Figure 2.21b).



Figure 2.21: Glyphs and patterns representing graphs of two different sizes

This is because the patterns are drawn as regular polygons with the original graph layout still displayed. Even if part of a pattern were to be obscured, the Gestalt principle of good continuation suggests that a user could still determine the number of nodes in the pattern based on the angles that edges intercept a node (see Figure 2.22).

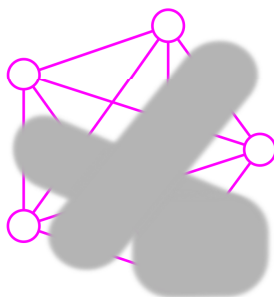


Figure 2.22: Example of an obscured clique

This ability to easily determine the size of a pattern while still displaying the original graph layout is a key difference between the work of Dunne and Schneidermann and the pattern based system described in this thesis.

2.4 Summary

This chapter has discussed relevant work in the fields of graph drawing, graph aesthetics, perceptual theories and subgraph isomorphism.

There are a large number of existing graph drawing techniques (see Section 2.1), from the popular force directed method developed by Eades [38], improved by Fruchterman and Reingold [48] and with many variations, to simulated annealing [32] and other methods. Each of these drawing techniques have their own advantages and weaknesses and many are tailored to particular types of graph or use cases.

Much research has also taken place into identifying the features that improve a graph drawing's usability and aesthetics (see Section 2.2). The Gestalt principles suggest various concepts which will improve a layout and authors [41, 116] have suggested various metrics to formally define a "good" layout. However, other authors [92] have performed numerous empirical evaluations and analysis in order to understand the effects that enable a usable and aesthetically pleasing graph layout.

The identification of patterns is closely related to the problem of subgraph isomorphism (see Section 2.3). Various authors [84, 86, 122, 135] have created methods of performing subgraph isomorphism, with some focussing on motifs. A similar piece of work creates the concept of *motif simplification* [36] (see Section 2.3.1.1, which identifies motifs within a graph and replaces these with glyphs.

Chapter 3

Description of Drawing Method

3.1 Introduction to the Drawing Method

This chapter describes the drawing method in great detail. In order to draw patterns in a consistent layout and achieve an aesthetically pleasing final result, several stages are required: definition of patterns (Section 3.2), identification of patterns (Section 3.3), determining a drawing order (Section 3.4) and the drawing of each pattern (Section 3.5). Each of these stages are described in their own section, with examples given of the various drawing methods for each connection type. Accompanying these descriptions are pseudocode algorithms which give a more formal definition of the process required. The overall method is described in Algorithm 3.1, and follows a number of graph drawing conventions, in this case a graph is a set of nodes and a set of edges¹. Internally within the software, each node has a list of edges, and each edge has two nodes – this redundant data structure allows easier implementation and expandability for

¹Often shown as $G \leftarrow (V, E)$. Functions in bold are self-explanatory (such as accessing or mutating properties of a node, edge or pattern), while those functions in all caps are detailed elsewhere.

other graph types. A pattern is considered to be a graph in itself, in that it has its own list of nodes and list of edges, although during identification these may be empty or the pattern is later discarded. Each node and edge also has a list of patterns that it is contained within.

Algorithm 3.1: Overall Algorithm

```

01: function OVERALLALGORITHM
02:    $G \leftarrow (V, E)$  ▷ Graph of Nodes and Edges
03:    $G \leftarrow \text{largestConnectedComponent}(G)$ 
04:    $P_u \leftarrow []$  ▷ Undrawn Patterns
05:    $P_d \leftarrow []$  ▷ Drawn Patterns
06:    $P_u \leftarrow P_u \cup \text{IDENTIFYCIRCLES}()$  ▷ See Algorithm 3.2
07:    $P_u \leftarrow P_u \cup \text{IDENTIFYCLIQUES}()$  ▷ See Algorithm 3.3
08:    $P_u \leftarrow P_u \cup \text{IDENTIFYSTARS}()$  ▷ See Algorithm 3.4
09:    $P_u \leftarrow P_u \cup \text{IDENTIFYPATHS}()$  ▷ See Algorithm 3.5
10:    $P_u \leftarrow \text{findDrawingOrder}(P_u)$  ▷ See Section 3.4
11:   for each Pattern  $G_p \in P_u$  do
12:     drawPattern( $G_p$ ) ▷ Draw pattern according to connection type, see algorithms in Section 3.5
13:      $P_u \leftarrow P_u \setminus \{G_p\}$ 
14:      $P_d \leftarrow P_d \cup \{G_p\}$ 
15:     for each Pattern  $\{G_p \mid G_p \in P_d, \text{isPath}(G_p) \text{ OR } \text{isRoute}(G_p)\}$  do
16:       ADJUSTPATHS( $G_p$ ) ▷ See Algorithm 3.19
17:     if  $P_d = \emptyset$  then
18:       Let drawn( $V[0]$ )  $\leftarrow \text{true}$ 
19:       ▷ If there are no patterns, then set one node to be drawn for the Loose Nodes method
20:     TIDYLOOSENODES( $G$ ) ▷ See Algorithm 3.20

```

3.2 Definition of Patterns

To draw patterns in a consistent layout, it is first essential to define what patterns are required. In this work, there are five types of subgraph that can be defined and identified within many graphs. These are cliques, circles, stars, paths and triangles. These patterns encompass the vast majority of nodes (as can be seen in Section A.7). One piece of further work described in Chapter 6 is

to define more patterns and potentially user-defined patterns. As terminology and definitions of these patterns (with the exception of cliques) are not concrete within the field, it is important to clarify the definitions used within this research. These definitions are as follows:

3.2.1 Circle

A circle:

- Must contain at least 4 nodes but no more than 8
- Each node must connect to exactly 2 other nodes in the pattern so that a closed path is formed
- Each node may connect to any number of other nodes not in the circle

There is a maximum limit imposed on a circle's size and this is to increase the performance of the system: searching for large circles is computationally slow and large circles are uncommon in graphs. The minimum limit is set at 4, as a circle of size 3 is classed as a triangle. In figures throughout this work, circles will be highlighted in blue and node labels will be prefixed with an o.

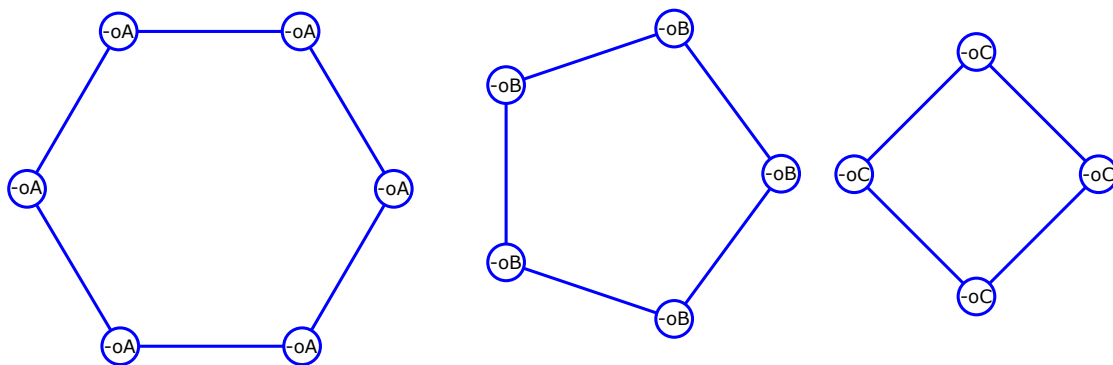


Figure 3.1: Example of circles of size 6, 5 and 4

3.2.2 Clique

A clique:

- Must contain at least 4 nodes
- All nodes are connected to the others within the clique
- All nodes can connect to nodes not within the clique

The minimum size limit is set at 4, as a clique with 3 sides is classed as a triangle. In figures throughout this work, cliques will be highlighted in pink and node labels will be prefixed with a c.

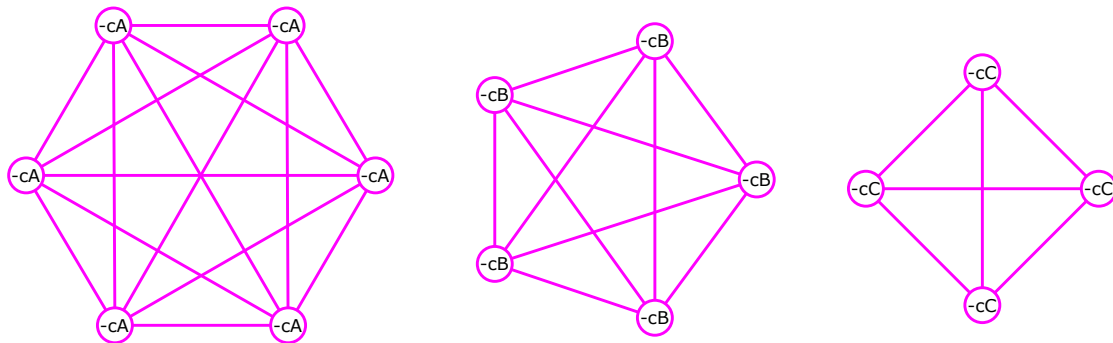


Figure 3.2: Example of cliques of size 6, 5 and 4

3.2.3 Star

A star:

- Must contain at least 5 nodes
- There is one central node which connects to all other nodes in the pattern
- All other nodes must only connect to the central node in this pattern
- Any node can connect to nodes outside the star

The minimum limit of 5 nodes is to allow 4 spokes and this retains consistency with the other pattern types. In figures throughout this work, stars will be highlighted in red and node labels will be prefixed with an s.

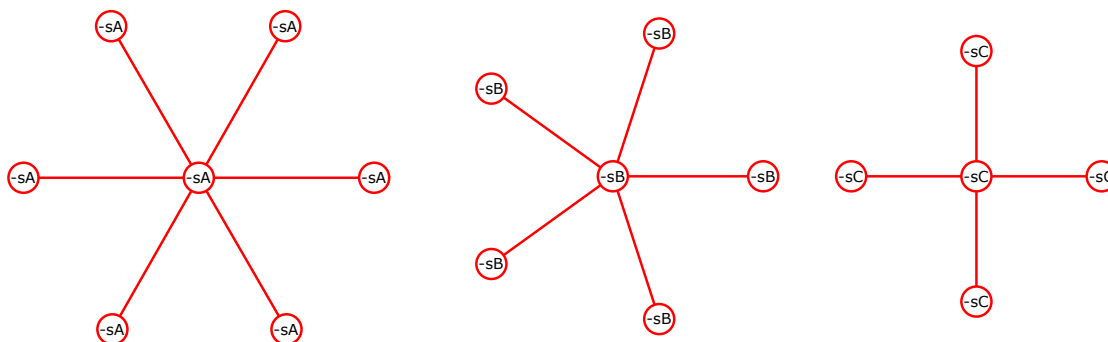


Figure 3.3: Example of stars of size 6, 5 and 4

3.2.4 Path

A path:

- Must contain at least 4 nodes
- The terminating nodes can connect to any other nodes outside of the pattern
- Other nodes in the pattern must connect to only 2 other nodes; the previous and next node in the path
- Any paths that are also circles are only treated as circles

The minimum limit of 4 nodes is set because a path made of 3 nodes is a far too common structure and can easily be drawn without a fixed method. In figures throughout this work, paths will be highlighted in orange or green and node labels will be prefixed with a p.

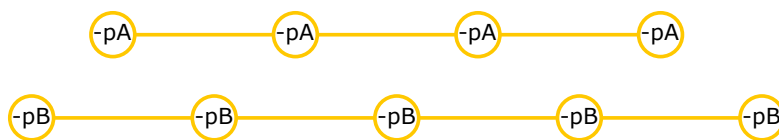


Figure 3.4: Example of paths of size 4 and 5

3.2.5 Triangle

A triangle:

- Must contain exactly 3 nodes
- All nodes must be connected to each other

Triangles are technically both small cliques and circles. However, for the purposes of this work, they are treated as a separate pattern. This is because triangles have a small number of nodes and therefore have a flexible layout. Triangles will be highlighted in cyan and node labels will be prefixed with a t .

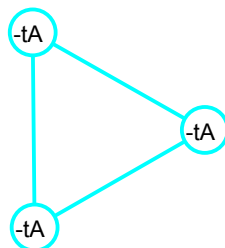


Figure 3.5: Example of a triangle

3.2.6 Other Notation

Within any graph there may be nodes that are not contained within any pattern (although these are quite rare). These are left highlighted in black or grey throughout figures in this work (See Figure 3.6).

Some nodes will have labels, consisting of pairs of letters. The first letter denotes the pattern type, as described above. The second letter is a unique

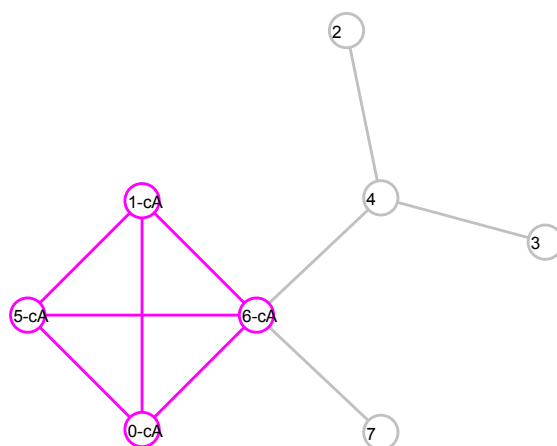


Figure 3.6: Other Nodes

identifier for that pattern, starting with A and continuing through the ASCII character set. For example, Figure 3.6 has one pattern: clique cA. In some examples, node labels may begin with a number - this is a unique identifier for each node. Figures in Chapters 4 and 5 have no highlighting, in order to make a fairer comparison with other drawing methods. In other examples, the list of patterns may be separated by a dash.

Many algorithms in this work have their time complexities displayed and discussed and the notation used in these is that, V represents the nodes within a the graph, E the edges within a graph, V_p the nodes within a pattern and E_p the edges within a pattern.

3.3 Identification of patterns

Once patterns have been defined, they are then identified within a given graph. Some graphs may have no patterns and not all patterns exist within all graphs. The processes used to identify each type of pattern are described below:

3.3.1 Circle

The first pattern to be identified is the circle, and the identification algorithm is displayed in Algorithm 3.2. During the identification process, checks are undertaken to ensure that no identical circles are found. For example, a circle of size 4 could be identified as 4 separate circles of the same size, depending on where the “start” is (see Figure 3.7, where the “start” is highlighted in pale blue). As all these patterns would be identical, it is necessary that only one is identified.

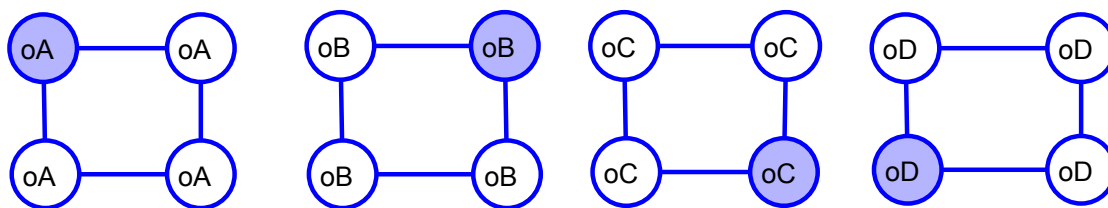


Figure 3.7: Four identical circles

Initially, the identification algorithm creates an empty array of valid circles. This is used to store identified circles. Then, for every node in the graph a potential circle is created with the current node as the starting point and stored in an array. Once this is complete, a breadth first search is performed over the potential circles to build a list of probable circles. This search ensures that there are potential circles in the array and the degree of the circle currently being searched for is less than or equal to the maximum permitted size.

A holding array of circles is created but left empty for later use. Then, iteration occurs over each potential circle and the start and end nodes are identified. Every edge that connects to the end node is then investigated and several checks are made.

The first check ensures that if the edge currently being investigated is already in the circle, then the algorithm is doubling back in its search trace, so this edge is ignored. If the node at the other end of the edge has less than 2

connections, then this search trace will come to an end, so again the edge is ignored. If, however, the edge's other connecting node (i.e. not the end node) is the same as the starting node then a circle has been found. A check is performed that ensures this circle is unique and is larger or equal in size to the minimum allowed length, and if so, that circle is added to the list of probable circles. Further checks are conducted to ensure that the connecting node is already part of the circle. If this the case, the edge is ignored. To reduce the number of potential circles, and to follow the definition of a circle, a check is performed to ensure that the connecting node does not connect to any other nodes within the circle; the start and end nodes are the exception. If additional connections are discovered, the edge is ignored. However, if all of these tests prove to be false, the connecting node and edge are added to the potential circle, and this circle is added to the holding array. Once all iterations through each potential circle are complete, the list of potential circles is cleared and made equal to the contents of the holding array. The length being searched for is increased and the main loop begins again.

Once the search has finished, either because there are no potential circles or the search length is too large, any probable circles with internal links are removed: those that remain are valid circles.

There is a maximum length on the size of circles allowed. This is mainly for performance reasons because searching for large circles is computationally time consuming and often yields very few results - in a number of random graphs tested, circles containing more than 8 nodes are exceptionally rare.

The process of identifying all circles is defined in Algorithm 3.2. The time complexity for this algorithm is $\mathcal{O}(V \cdot \log E \cdot \log V)$, as the algorithm must iterate through all nodes (V , Line 6), connecting edges ($\log E$, Line 13), and nodes in the existing circles ($\log V$, Line 25).

Algorithm 3.2: Circle identification

```

01: function IDENTIFYCIRCLES
02:    $G \leftarrow (V, E)$ 
03:    $minSize \leftarrow 4$ 
04:    $maxSize \leftarrow 8$ 
05:    $probableCircles \leftarrow \emptyset$  ▷ Used for storing circles for testing
06:    $potentialCircles \leftarrow \{([v], \emptyset) \mid v \in V\}$  ▷ A potentialCircle is created for every node in  $N$ 
07:    $length \leftarrow 1$ 
08:   while  $potentialCircles \neq \emptyset$  AND  $length \leq maxSize$  do
09:      $nextCircles \leftarrow \emptyset$ 
10:     for each  $(V_a, E_a) \in potentialCircles$  do
11:        $n_{start} \leftarrow \mathbf{firstNode}(V_a)$ 
12:        $n_{end} \leftarrow \mathbf{lastNode}(V_a)$ 
13:       for each Edge  $e \in \mathbf{connectingEdges}(v_{end})$  do
14:          $v_{connecting} \leftarrow \mathbf{oppositeEnd}(e, v_{end})$ 
15:         if  $e \in E_a$  then
16:           | Skip this edge ▷ Search is doubling back on itself
17:         if  $\mathbf{degree}(v_{connecting}) < 2$  then
18:           | Skip this edge ▷ Dead end
19:         if  $n_{connecting} = v_{start}$  then ▷ Circle potentially found
20:           | if  $|V_a| \geq minSize$  AND  $\mathbf{uniqueCircle}((V_a, E_a))$  then
21:             |  $probableCircles \leftarrow probableCircles \cup (V_a, E_a)$ 
22:             | Skip this edge
23:           | if  $v_{connecting} \in V_a$  then ▷ Node is already in the list
24:             | Skip this edge
25:           | for each Node  $v_a \in V_a$  do
26:             | if  $\mathbf{edgesBetween}(v_a, v_{connecting}) \neq \emptyset$  then
27:               | Skip this edge ▷ Check for internal connections
28:             |  $(V_b, E_b) \leftarrow (V_a \cup v_{connecting}, E_a \cup e)$ 
29:             |  $nextCircles \leftarrow nextCircles \cup G_b$ 
30:           |  $potentialCircles \leftarrow nextCircles$ 
31:          $length \leftarrow length + 1$ 
32:    $foundCircles \leftarrow \{G_f \mid G_f \in probableCircles, \mathbf{internalLinks}(G_f) = \emptyset\}$  ▷ Save all circles with no
internal links

```

3.3.2 Clique

Cliques are the next pattern to be identified. The algorithm attempts to find the largest possible clique within a graph, which is the same size as the node with the highest degree. If this cannot be found, cliques of decreasing size are

searched for. It is preferable to identify a few large cliques than several smaller ones. For example, a clique of size 5 also contains five cliques of size 4 (see Figure 3.8 where a clique of 4, in yellow, is contained within a clique of size 5). Therefore, the identification algorithm ignores cliques fully contained within others. In effect, this algorithm discovers all the maximal cliques in the graph [130].

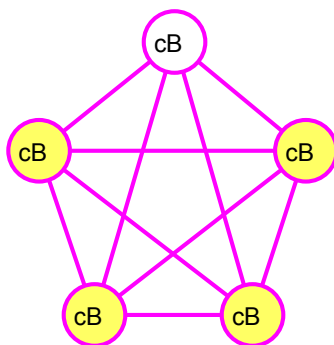


Figure 3.8: A clique within a clique

The algorithm considers every node as a potential starting point within a clique of given size. However, if this node has either too few connections (i.e. its degree is less than the required size of the clique minus 1) or is already in a clique (it has a visited flag set), it is ignored. If a node is considered to be a potential starting place then it is given a score of 1. Each child node of the starter is then investigated in turn and each child must have a degree greater than or equal to the required clique size minus 1 and the child must not have a visited flag set. If this is the case then this child is given a score of 2, but if not, this child is ignored as it is impossible that this node could form part of a clique. Any children that have not been ignored have their children (i.e. the grandchildren of the potential starter) investigated. These grandchildren must also have a degree greater than or equal to the required clique size minus 1 or have no visited flag set. If this is the case then their score is incremented by 1.

Once various nodes have been given scores determined by the number of times they have been visited, each node in the graph is checked to ensure it is valid for inclusion in a clique. The starting node's score is considered first and if this is equal to, or greater than, the required clique size, the node is added to the holding array. If this is not the case, all node scores are reset and another starting point is considered. If the node meets the specified requirements, its children are then tested in the same way. Again, the degrees of node's children must be at least the required clique size. If they are, they are also added to the holding array. This can be seen in Figure 3.9, where the nodes within the clique all have the correct score. The nodes in the star have either a score of 0 (because they do not have enough connections), or 1 where they are the child of the connecting node. Neither of these scores meet the size of clique being searched for and therefore those nodes cannot be considered part of the clique.

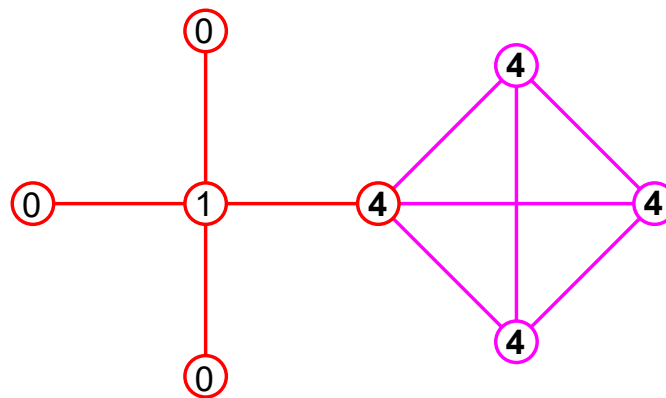


Figure 3.9: Example of the clique scoring identification method

After this process is complete, a comparison is made between the size of the holding array and the required clique size. If they are equal, a new clique is created and all nodes in the holding array and all edges that connect to another node in the clique are added. Each node is set to have been “visited” to prevent this clique being added multiple times.

The final check ensures that the number of edges is equal to

$$\frac{|V|(|V| - 1)}{2}$$

where $|V|$ represents the number of nodes in the clique. If the clique passes this final test it has been identified correctly and is added to the list of patterns in the graph. This process then repeats again with a search size 1 smaller, until a minimum size of 4 is reached.

The process of identifying all cliques and triangles is defined in Algorithm 3.3. The time complexity for this algorithm is $\mathcal{O}(V \cdot (\log V)^2)$, as the algorithm must iterate through all nodes (V , Line 9), connecting nodes ($\log V$, Line 14), and nodes that connect to those connecting nodes ($\log V$, Line 18).

Algorithm 3.3: Clique & Triangle identification

```

01: function IDENTIFYCLIQUES
02:    $G \leftarrow (V, E)$ 
03:    $searchSize \leftarrow |V|$ 
04:    $P_t \leftarrow \emptyset$ 
05:    $P_c \leftarrow \emptyset$ 
06:    $visited(G) \leftarrow \text{false}$ 
07:    $score(V) \leftarrow 0$ 
08:   while  $searchSize \geq 3$  do ▷ The minimum size allowed
09:     for each Node  $v_a \in V$  do
10:       if  $degree(v_a) < searchSize - 1$  OR  $visited(v_a)$  then
11:          $score(V) \leftarrow 0$ 
12:         Skip this node
13:        $score(v_a) \leftarrow 1$ 
14:       for each Node  $v_b \in \text{connectedTo}(v_a)$  do
15:         if  $degree(v_b) < searchSize - 1$  OR  $visited(v_b)$  then
16:           Skip this node
17:          $score(v_b) \leftarrow score(v_b) + 2$ 
18:         for each Node  $v_c \in \text{connectedTo}(v_b)$  do
19:           if  $degree(v_c) < searchSize - 1$  OR  $visited(v_c)$  then
20:             Skip this node
21:            $score(v_c) \leftarrow score(v_c) + 1$ 
22:        $V_p = \{v \mid score(v) = searchSize, v \in (v_a \cup \text{connectedTo}(v_a))\}$ 
23:        $E_p = \text{connectingEdges}(V_p)$ 

```

```

24: | | |  $G_p \leftarrow (V_p, E_p)$ 
25: | | | if  $|V_p| = searchSize$  AND  $|E_p| = \frac{|V_p|(|V_p|-1)}{2}$  then
26: | | |   visited( $V_p$ ) = true
27: | | |   if  $|V_p| = 3$  then
28: | | |      $P_t \leftarrow P_t \cup G_p$                                 ▷ Save as triangle
29: | | |   else
30: | | |      $P_c \leftarrow P_c \cup G_p$                                 ▷ Save as clique
31: | | |   else
32: | | |     score( $|V_p|$ )  $\leftarrow 0$ 
33: | | |     score( $V$ )  $\leftarrow 0$ 
34: | | |    $searchSize \leftarrow searchSize - 1$ 

```

3.3.3 Star

The third type of pattern identified is stars. Stars may also contain other stars. For example a star with 5 branches is also 5 stars with 4 branches each with the same centre, therefore only the largest is returned by the algorithm. The psuedocode for the star identification algorithm is displayed in Algorithm 3.4.

The algorithm starts by iterating through every node in the graph and checking that this node's degree is greater than or equal to the valid minimum size of a star. If this is the case, then this node is added to a list of potential centres.

Once complete, each potential centre is then investigated in turn and an array of potential branches is created from this node's connections. Every connecting edge of each branch is then investigated and if this connection is also a branch, then both the original potential branch and the connection are removed as potential branches. The size of the array of potential branches is then compared to the minimum size of stars permitted and if the array's size is greater than or equal to the minimum size, a star is created. All branches (including nodes and edges) are then added to the star and the next potential centre is considered.

Potential branches are removed because all nodes (with the exception of the

centre node, which must connect to all others) must only connect to the centre node. It is this relationship of connections that forms the star.

The process of identifying all stars is defined in Algorithm 3.4. The time complexity for this algorithm is $\mathcal{O}(V \cdot \log V)$, as the algorithm must iterate through all nodes (V , Line 6), and connecting nodes ($\log V$, Line 7).

Algorithm 3.4: Star identification

```

01: function IDENTIFYSTARS
02:    $G \leftarrow (V, E)$ 
03:    $P_s \leftarrow \emptyset$ 
04:    $minSize \leftarrow 4$  ▷ The smallest number of spokes allowed
05:    $V_{pc} = \{v \mid v \in V, \mathbf{degree}(v) \geq minSize\}$  ▷ Find potential centres
06:   for each Node  $v_a \in V_{pc}$  do
07:      $V_{branches} = \{v \mid v \in V, v \in \mathbf{connectedTo}(v_a), \mathbf{connectedTo}(v) \cap \mathbf{connectedTo}(v_a) = \emptyset\}$ 
08:     ▷ All nodes connected to  $v_a$ , but not those that are connected to other branches
09:     if  $|V_{branches}| \geq minSize$  then
10:        $V_p = v_a \cup V_{branches}$ 
11:        $E_p = \mathbf{connectingEdges}(V_p)$ 
12:        $G_p \leftarrow (V_p, E_p)$ 
13:        $P_s \leftarrow P_s \cup G_p$  ▷ Save as star

```

3.3.4 Path

Paths are the next pattern to be identified. Special care has been taken within the identification algorithm to ensure that closed paths are not found: they are instead represented as circles. As with other patterns, paths may contain additional paths. For example, a path of length 5 contains two paths of length 4, therefore only the longest path is found. The algorithm starts in the middle of a path and builds either side, rather than beginning the search at a start or end point. The pseudocode for the path identification algorithm is displayed in Algorithm 3.5.

The algorithm starts by iterating through every node within the graph. If

this node has been visited, or has a degree that is not equal to 2 then it is ignored. If this node does not, a path is created and this node is added to it. The algorithm then searches one direction through the path (notionally “forward”). The “forward” edge is added to the path, and the next node is set to be the starting point. A loop is then run which terminates when the next node’s degree is not 2, because this condition signals the end of the path. Inside the loop, the next node is set to be the opposite end of the edge (to the current node) and is added to the path. If this next node has a degree of 2, then the node is checked to ensure it has at least one unvisited connecting edge. If not, this is a path and the loop breaks. If the node satisfies this criteria, then the edge is set to be this unvisited edge (the node can have only one), marked as “visited” and added to the path.

When this loop breaks, it is run again, but this time searching the other direction (i.e. “backwards”). The algorithm follows an identical process, except nodes and edges are added to the start of the array to ensure the array order matches that of the path. Once complete, checks are conducted to ensure the first and last nodes are different. If they are the same, the last node is removed. The potential path is only added to the list of patterns if the path is at least as long as the minimum allowed length and is not identical to an already existing circle.

The process of identifying all paths is defined in Algorithm 3.5. The time complexity for this algorithm is $\mathcal{O}(V \cdot \log V)$, as the algorithm must iterate through all nodes (V , Line 5), and connecting nodes ($\log V$, Line 9).

Algorithm 3.5: Path identification

```

01: function IDENTIFYPATHS
02:    $G \leftarrow (V, E)$ 
03:    $minSize \leftarrow 4$  ▷ The smallest number of nodes allowed
04:    $visited(G) \leftarrow false$ 
05:   for each Node  $v \in V$  do

```

```

06: | | if degree( $v$ )  $\neq$  2 OR visited( $v$ ) then
07: | | | Skip this node
08: | | ( $V_p, E_p$ )  $\leftarrow$  ( $\{v\}, []$ ) ▷ Empty path for later use
09: | |  $pathFound \leftarrow$  BUILDPATH( $(V_p, E_p), n, minSize$ ) ▷ Build Path
10: | | if  $pathFound = true$  AND notACircle( $(V_p, E_p)$ ) then
11: | | | Save ( $V_p, E_p$ ) as a path
12:
13: function BUILDPATH( $(V_p, E_p), v, minSize$ )
14: | |  $E_a \leftarrow []$  ▷ Stores edges for use later
15: | |  $V_a \leftarrow []$  ▷ Stores nodes for use later
16: | | for  $i \in \{0, 1\}$  do ▷ Searching forwards (0) or backwards (1)
17: | | |  $e \leftarrow$  connectingEdges( $v$ )[ $i$ ] ▷ Start searching
18: | | | visited( $e$ )  $\leftarrow true$ 
19: | | |  $E_a \leftarrow E_a \cup e$ 
20: | | |  $v_{next} \leftarrow v$  ▷  $v$  will be used again later so overwriting is not possible
21: | | | if  $i = 0$  then
22: | | | | visited( $v_{next}$ )  $\leftarrow true$ 
23: | | | |  $V_a \leftarrow V_a \cup v_{next}$ 
24: | | | | while degree( $v_{next}$ ) = 2 do
25: | | | | |  $v_{next} \leftarrow$  oppositeEnd( $e, v_{next}$ )
26: | | | | | visited( $v_{next}$ )  $\leftarrow true$ 
27: | | | | |  $V_a \leftarrow V_a \cup v_{next}$ 
28: | | | | | if degree( $v_{next}$ ) = 2 then
29: | | | | | | if unvisitedConnectingEdges( $v_{next}$ ) =  $\emptyset$  then
30: | | | | | | | Break while ▷ Stop searching in this direction
31: | | | | | |  $e \leftarrow$  unvisitedConnectingEdges( $v_{next}$ )[0]
32: | | | | | | visited( $e$ )  $\leftarrow true$ 
33: | | | | | |  $E_a \leftarrow E_a \cup e$ 
34: | | | if  $V_a[0] = V_a[i]$  then ▷ If the first and last nodes in  $V_a$  are the same, then this is a closed path
35: | | | | Remove last node from  $V_a$ 
36: | | | if  $|V_a| < minSize$  then ▷ If the path is too small
37: | | | | return false
38: | | |  $V_p \leftarrow V_p \cup V_a$ 
39: | | |  $E_p \leftarrow E_p \cup E_a$ 
40: | | return true ▷ Path has been built

```

3.3.5 Triangle

The identification for triangles is performed at the same time as the identification of cliques (See Section 3.3.2). Whereas the clique method ensures that all patterns have more than 4 nodes, the triangle identification method ensures that the pattern contains exactly 3 nodes.

3.4 Determining the drawing order

Once all patterns have been identified, it is then necessary to determine the order in which they should be drawn. When determining the order, several considerations need to be made:

- The type of connection to the previously drawn patterns
 - One Node shared
 - * The pattern and the drawn set have one node in common.
 - Two Nodes shared
 - * The pattern and the drawn set have two nodes in common. This will often include circles and paths connected to non-consecutive nodes in circles. The pattern does not share any edges with the drawn set.
 - One Edge shared
 - * The pattern and the drawn set have one edge in common (and therefore also 2 nodes).
 - Two Edges shared
 - * The pattern and the drawn set have two edges in common (and therefore also 3 or 4 nodes).
 - Edge connection
 - * The pattern and the drawn set have one or more edges connecting them. This edge is in neither the drawn set, nor the pattern.
 - No connection
 - * The pattern and the drawn set have no nodes or edges in common, nor have any edges connecting them. This does not mean

the pattern is disconnected from the drawn set - it could be connected by a series of nodes and edges. This isn't considered to be a close connection, so this is treated as no connection.

- The amount of connectivity to the previously drawn patterns
 - Number of nodes shared (if sharing nodes)
 - Number of edges shared (if sharing edges)
- Degree of pattern
- Type of pattern

While it may appear that a pattern that shares an edge must also share two nodes, there is a distinction used within this work. Patterns which only share nodes (but not edges) with a drawn set are considered to be *node sharing*, while those that also share edges are considered to be *edge sharing*.

It is also necessary to discard a number of patterns that are identified, because many graphs have an unfeasibly large number of patterns or patterns which considerably overlap others. This is demonstrated in Figure 3.10, where the thickness of each edge is dependent on the number of patterns which contains this edge. Each individual pattern is shown in a distinctive colour, which has no relation to the colours previously defined. Edges with no patterns are shown as thin and in grey. This makes the process of drawing these patterns infeasible, and therefore only a subset of patterns are chosen for drawing. As a result of this, patterns which do not meet the connection type criteria above (i.e. the pattern shares more than 2 edges or more than 2 nodes (and no edges)) with the drawn set are disregarded.

The order for drawing has been identified based on the flexibility of patterns. Cliques are less flexible as they have a large number of edges compared

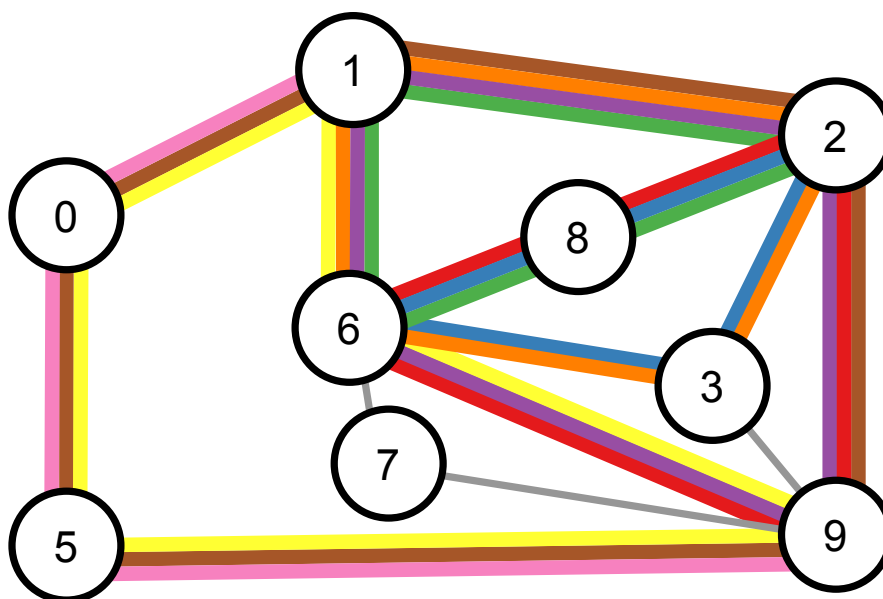


Figure 3.10: Graph with 10 nodes, but with 8 patterns

to other patterns, while stars are required to have every spoke connect back to the centre. Circles have increased flexibility in that they often have space inside themselves for drawing, while paths and triangles are very flexible. Patterns which share 1 or 2 edges are also very closely related and there is less flexibility with regards to location. They are therefore drawn earlier. A similar reasoning also draws patterns with 1 or 2 shared nodes following this. Patterns with only a connecting edge, or no connection at all are drawn last. This order can be changed, but this is a subject for further work and is discussed in Section 6.2.5.

The patterns are drawn in the following order:

Cliques

1. Clique with 1 edge shared (see Figure 3.11)
2. Clique with 1 node shared (see Figure 3.12)

Stars

3. Star with 1 outside node shared (see Figure 3.13)

4. Star with 1 edge shared (see Figure 3.14)
5. Star with centre node shared (see Figure 3.15)

Circles

6. Circle with 1 edge shared (see Figure 3.16)
7. Circle with 2 edges shared (see Figure 3.17)
8. Circle with 2 nodes shared (see Figure 3.18)
9. Circle with 1 node shared (see Figure 3.19)

Paths

10. Path with 2 edges shared (see Figure 3.20)
11. Path with 1 edge shared (see Figure 3.21)
12. Path with 2 nodes shared (see Figure 3.22)
13. Path with 1 node shared (see Figure 3.23)

Connecting Edges

14. Clique with connecting edges (see Figure 3.24)
15. Star with connecting edges (see Figure 3.25)
16. Circle with connecting edges (see Figure 3.26)
17. Path with connecting edges (see Figure 3.27)

Triangles

18. Triangle with 1 edge shared (see Figure 3.28)
19. Triangle with 2 nodes shared (see Figure 3.29)
20. Triangle with 1 node shared (see Figure 3.30)
21. Triangle with connecting edges (see Figure 3.31)

No connections

22. Clique with no connection (see Figure 3.32)
23. Star with no connection (see Figure 3.33)

- 24. Circle with no connection (see Figure 3.34)
- 25. Path with no connection (see Figure 3.35)
- 26. Triangle with no connection (see Figure 3.36)

For connections where patterns are connected by edges, the largest pattern with the most number of connections is chosen. For all other types of connection, simply the largest pattern is chosen.

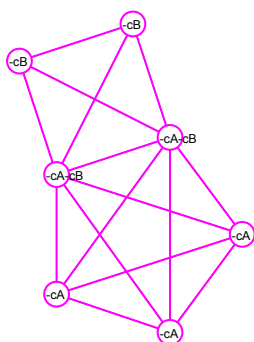


Figure 3.11: Clique with 1 edge shared

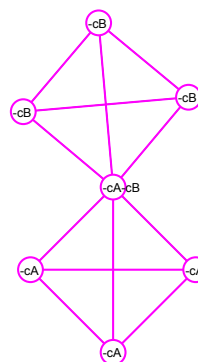


Figure 3.12: Clique with 1 node shared

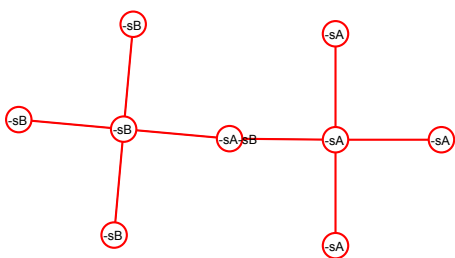


Figure 3.13: Star with 1 outside node shared

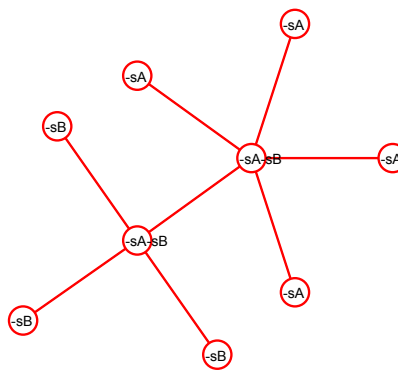


Figure 3.14: Star with 1 edge shared

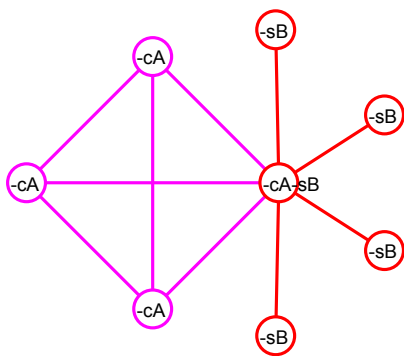


Figure 3.15: Star with centre node shared

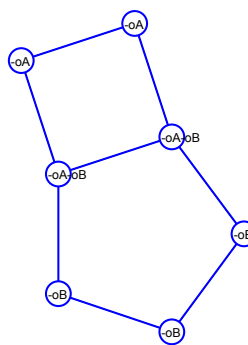


Figure 3.16: Circle with 1 edge shared

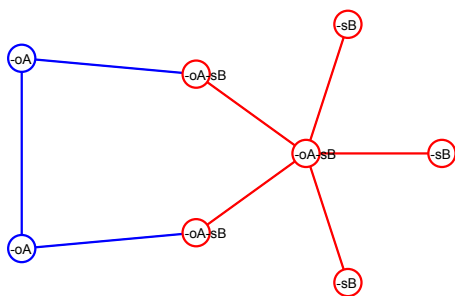


Figure 3.17: Circle with 2 edges shared

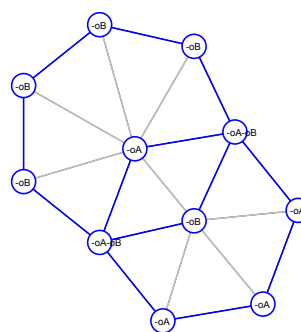


Figure 3.18: Circle with 2 nodes shared

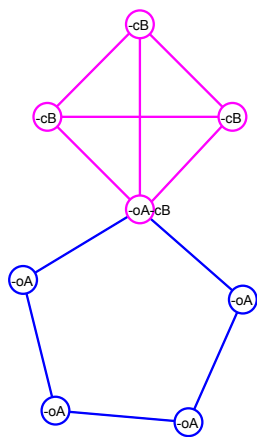


Figure 3.19: Circle with 1 node shared

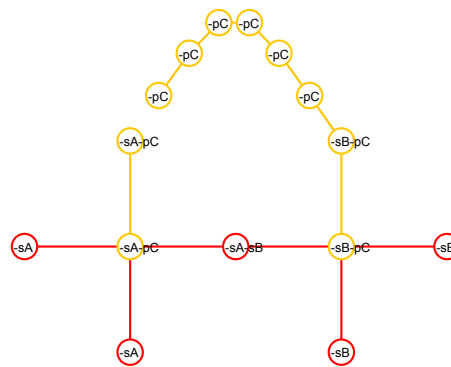


Figure 3.20: Path with 2 edges shared

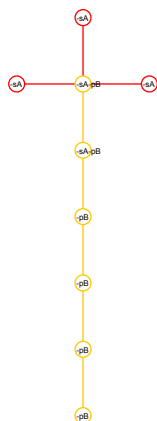


Figure 3.21: Path with 1 edge shared

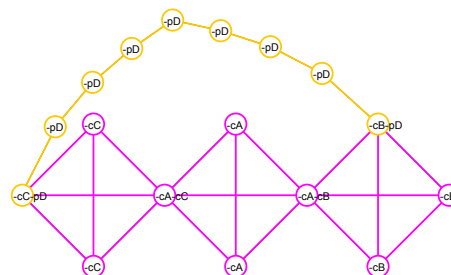


Figure 3.22: Path with 2 nodes shared

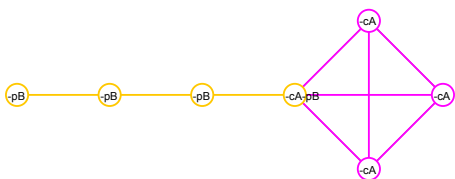


Figure 3.23: Path with 1 node shared

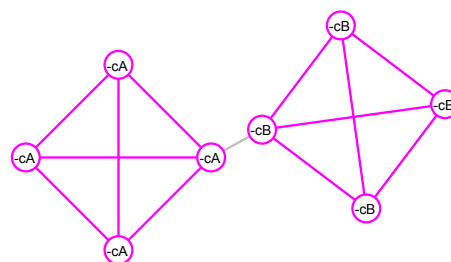


Figure 3.24: Clique with connecting edges

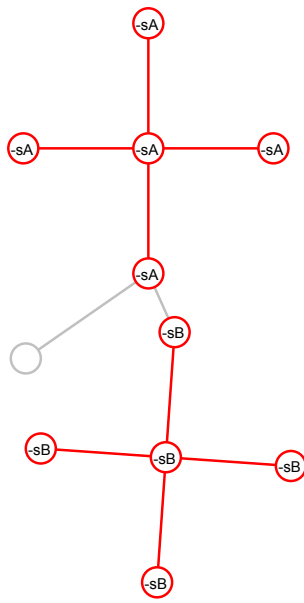


Figure 3.25: Star with connecting edges

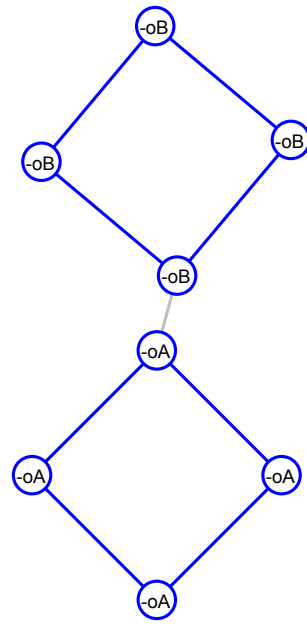


Figure 3.26: Circle with connecting edges

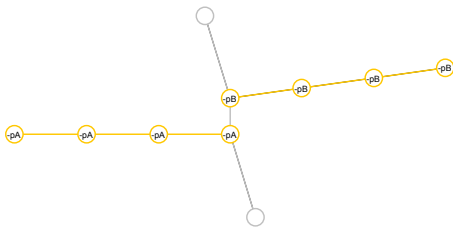


Figure 3.27: Path with connecting edges

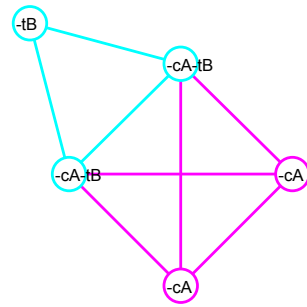


Figure 3.28: Triangle with 1 edge shared

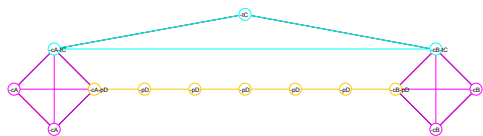


Figure 3.29: Triangle with 2 nodes shared

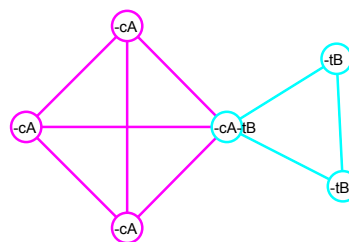


Figure 3.30: Triangle with 1 node shared

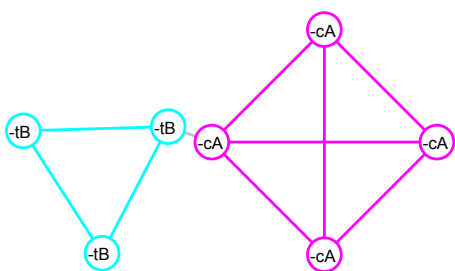


Figure 3.31: Triangle with connecting edges

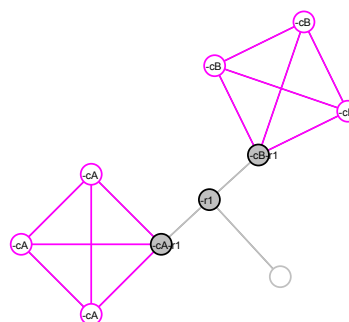


Figure 3.32: Clique with no connection

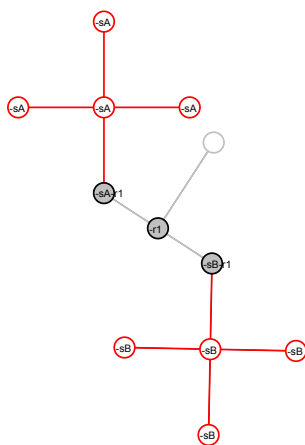


Figure 3.33: Star with no connection

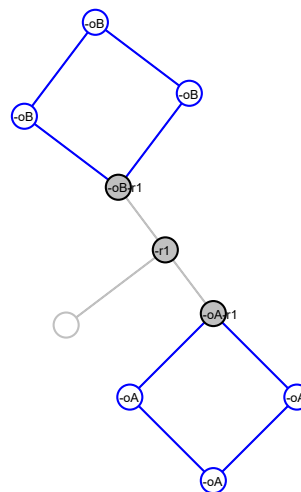


Figure 3.34: Circle with no connection



Figure 3.35: Path with no connection

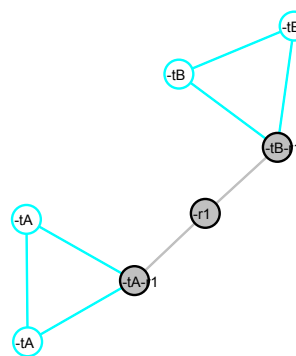
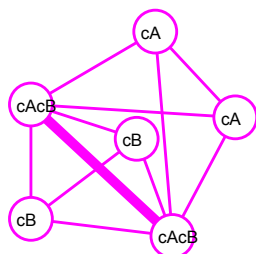
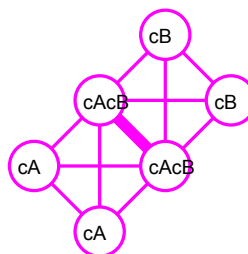


Figure 3.36: Triangle with no connection

Once the order of drawing is established, certain cliques are manipulated. This is necessary because it helps to improve the final layout. The nodes within cliques that have a shared edge require reordering to ensure that this particular edge is on the “outside” of the pattern. This helps avoid occlusion and to minimize edge crossings. For example, the layout of Figure 3.37b, where the shared edge is on the “outside”, is much preferable to the layout of Figure 3.37a where the share edge is an “inside” edge.



(a) Two cliques joined on an “inside” edge



(b) Two cliques joined on an “outside” edge

Figure 3.37: Comparison of 1 edge sharing cliques

3.5 Drawing each pattern

Once the drawing order has been established, the patterns can be drawn. The algorithms for drawing each pattern are different for each type of connection and, in some cases, the type of pattern. Some connection types (Shared One Node, Shared One Edge, Connected By One Edge) allow the possibility of swapping the order of nodes or edges within the pattern to allow a better layout. Other connection types do not allow this. While it is possible that swapping could be implemented for these, it was decided against doing this based on the time required for implementation and the relatively low occurrence of these connection types.

Cliques, circles, stars and triangles can be drawn according to an “ideal layout” which is the default layout of a pattern if it had no connections. The default layout also helps to ensure consistency when drawing patterns. Slight modifications may be required to enable the ideal layout to handle connections, but these are kept at a minimum to maximise consistency. The layout of paths is flexible. This means because that they do not have an ideal layout to follow, as they are usually drawn later. This enables the drawing method to take advantage of the available drawing space.

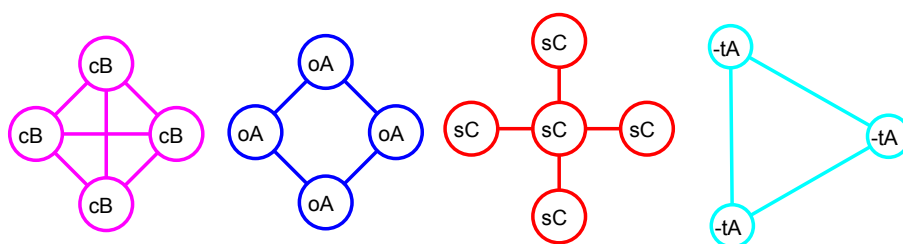


Figure 3.38: Ideal Layouts for a Clique, Circle, Star & Triangle

The first pattern drawn always follows its ideal layout. Patterns are drawn according to the techniques listed below:

3.5.1 Share One Edge

Circles, cliques and triangles share the same drawing algorithm when patterns share one edge within the drawn set (Section 3.5.1.2), whereas stars have a unique drawing method (Section 3.5.1.3). It is also possible for paths to share an edge with another pattern.

3.5.1.1 Edge Swapping

Edges may be swapped in the pattern to produce a better layout. For example, the order of nodes (and therefore edges) in a clique or spokes in a star is irrelevant and by changing this order, a better drawing result may be obtained. Therefore, a list of potential swaps is generated. If the shared edge has more than one drawn pattern connecting to it, then the current order is used. This prevents any already drawn nodes from being moved.

Both stars and cliques have similar methods for determining the potential edges to be swapped. Any edges in the pattern which meet the following conditions are considered to be potential swapping edges:

- Are in only one drawn pattern
- Both the start and finish nodes also are only in one drawn pattern
- (Cliques Only) Do not cross any of the other edges in the drawn pattern (i.e. on the outside)

For each of these potential swapping edges, the locations of the start and finish nodes are swapped with that of the shared edges. However, any nodes which are the centre of a drawn star remain in their original locations. If the shared and swapping edge happen to share a node, then the other two nodes are swapped. The pattern is then drawn according to the methods detailed in Sections 3.5.1.2,

3.5.1.3 and 3.5.8. Once drawn, a score is calculated based on node-node occlusion, edge-node occlusion, edge crossings and symmetry (Section 3.6.3). When all options have been tested, the swap with the best score is used for the final layout.

The process of swapping edges is described in Algorithm 3.6. The time complexity for this algorithm is dependent on the type of pattern being drawn. If the pattern to be drawn is a clique or triangle then the complexity is $\mathcal{O}(E_p + V_p)$, if the pattern is a star then the complexity is $\mathcal{O}(E_p + (V \cdot V_p))$, or if the pattern is a circle then the complexity is $\mathcal{O}(V \cdot V_p)$. This is due to the algorithm iterating through all edges in the pattern (E_p , Lines 9 and 14), if needed, and the time complexity of Algorithm 3.7 (V_p) or Algorithm 3.8 ($V \cdot V_p$).

Algorithm 3.6: Edge Swapping (Share One Edge)

```

01: function EDGESWAPPINGSHAREONEEDGE
02:    $G \leftarrow (V, E)$ 
03:    $G_p \leftarrow (V_p, E_p)$  ▷ The pattern about to be drawn
04:    $e_{shared} \leftarrow$  The Shared Edge
05:   if  $patterns(e_{shared}) = 2$  then ▷ Swapping can only occur if there's one other pattern sharing this
      edge ▷ The other pattern sharing the edge
06:      $G_{other} \leftarrow sharedPattern(e_{shared})$ 
07:      $E_{swaps} \leftarrow \emptyset$ 
08:     if  $isStar(G_{other})$  then
09:       for each Edge  $e \in E_p$  do
10:         if  $drawnPatterns(e) = 1$  AND  $e \neq e_{shared}$ 
11:         AND  $drawnPatterns(connectedTo(e)) = 1$  then ▷ If  $e$  has only one drawn pattern, is not the shared
            edge, and the nodes at either end also only have one drawn pattern, then this is a swap
12:           Let  $E_{swaps} \leftarrow E_{swaps} \cup e$ 
13:           else if  $isClique(G_{other})$  then
14:             for each Edge  $e \in E_p$  do
15:               if  $drawnPatterns(e) = 1$  AND  $e \neq e_{shared}$ 
16:               AND  $drawnPatterns(connectedTo(e)) = 1$  AND  $outsideEdge(e)$  then ▷ If  $e$  has only one
                drawn pattern, is not the shared edge, the nodes at either end also only have one drawn pattern, and
                this edge is on the outside of a clique, then this is a swap
17:                  $E_{swaps} \leftarrow E_{swaps} \cup e$ 
18:             else
19:               if  $isClique(G_p)$  OR  $isTriangle(G_p)$  then
20:                 DRAWPATTERN( $G_p$ ) ▷ Draw pattern (Clique or Triangle. See Algorithm 3.7)
21:             else

```

```

22: | | | DRAWPATTERN( $G_p$ )                                ▷ Draw pattern (Star. See Algorithm 3.8)
23: | | if  $E_{swaps} \neq \emptyset$  then
24: | |    $bestScore \leftarrow \infty$ 
25: | |    $bestSwap \leftarrow null$ 
26: | |   for each  $e \in E_{swap}$  do
27: | |     if isStar( $G_{other}$ ) then
28: | |       swapNodes(outsideNode( $e$ ), outsideNode( $e_{shared}$ )) ▷ Don't swap the centre of a
star | |
29: | |     else
30: | |       swapEdges( $e, e_{shared}$ )                                ▷ Swap the edges of a clique
31: | |     if isClique( $G_p$ ) OR isTriangle( $G_p$ ) then
32: | |       DRAWPATTERN( $G_p$ )                                ▷ Draw pattern (Clique or Triangle. See Algorithm 3.7)
33: | |     else
34: | |       DRAWPATTERN( $G_p$ )                                ▷ Draw pattern (Star. See Algorithm 3.8)
35: | |     Let  $currentScore \leftarrow$  CALCULATESCORE( $drawnPatterns$ ) ▷ See Algorithm 3.26
36: | |     if  $currentScore < bestScore$  then
37: | |        $bestScore \leftarrow currentScore$ 
38: | |        $bestSwap \leftarrow (e, e_{shared})$ 
39: | |     Swap back
40: | |   if isStar( $G_{other}$ ) then                                ▷ Swap Best Pairing
41: | |     swapNodes(outsideNodes( $bestSwap$ ))
42: | |   else
43: | |     swapEdges( $bestSwap$ )
44: | |   if isClique( $G_p$ ) OR isTriangle( $G_p$ ) then
45: | |     DRAWPATTERN( $G_p$ )                                ▷ Draw pattern (Clique or Triangle. See Algorithm 3.7)
46: | |   else
47: | |     DRAWPATTERN( $G_p$ )                                ▷ Draw pattern (Star. See Algorithm 3.8)
48: | | else
49: | |   if isClique( $G_p$ ) OR isTriangle( $G_p$ ) then
50: | |     DRAWPATTERN( $G_p$ )                                ▷ Draw pattern (Clique or Triangle. See Algorithm 3.7)
51: | |   else
52: | |     DRAWPATTERN( $G_p$ )                                ▷ Draw pattern (Star. See Algorithm 3.8)
53: | | else
54: | |   if isClique( $G_p$ ) OR isTriangle( $G_p$ ) then
55: | |     DRAWPATTERN( $G_p$ )                                ▷ Draw pattern (Clique or Triangle. See Algorithm 3.7)
56: | |   else
57: | |     DRAWPATTERN( $G_p$ )                                ▷ Draw pattern (Star. See Algorithm 3.8)

```

3.5.1.2 Circle, Clique & Triangle

Circles, cliques and triangles currently share the same algorithm for drawing. In this algorithm, the pattern is first drawn in its ideal layout, centred on the origin of the graph. The pattern is then scaled so all outside edges match the same length as the shared edge. This ensures the pattern does not distort the currently drawn set and that this pattern is drawn in a consistent manner. For example, in Figure 3.39, the clique is still drawn as a square, rather than being stretched by the circle into a trapezium.

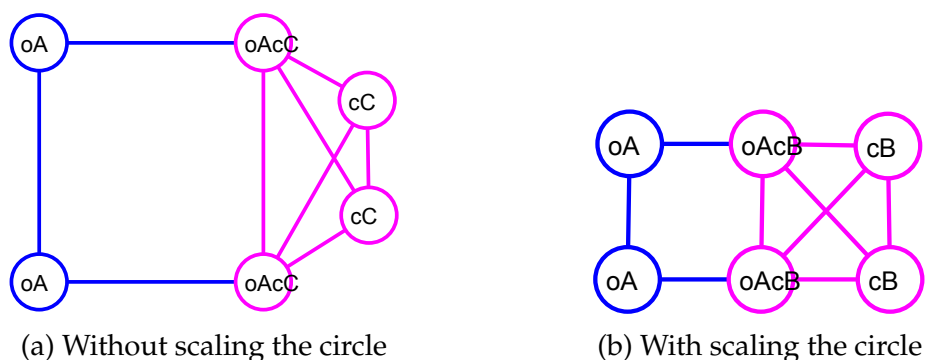


Figure 3.39: Comparison of a drawn pattern with and without scaling

The pattern is then rotated to match the original orientation of the shared edge and moved into the correct location. After rotation, the shared edge and nodes have been returned to the correct position. However, the remaining pattern may be inside the drawn set. This could mean the pattern is positioned in a way that creates unnecessary edge crossings or occlusions. This is shown in Figure 3.40 where reflection has taken place along the edge connecting the two nodes labelled oAcB). To overcome such issues, the pattern may be reflected. Reflection enables incorrectly placed nodes to move to a more aesthetically pleasing position.

When reflection has been completed, the pattern sits in the correct location. Examples of various patterns drawn with this connection type are displayed in

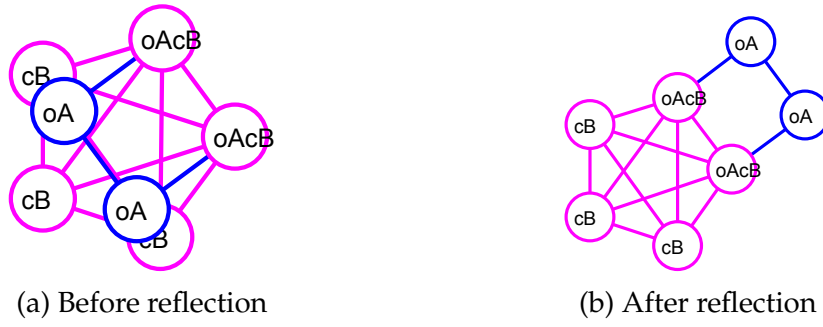


Figure 3.40: Comparison of a drawn pattern before and after reflection

Figure 3.41. The process of drawing a circle, clique or triangle is described in Algorithm 3.7. The time complexity for this algorithm is $\mathcal{O}(\log V_p)$, due to the algorithm iterating through all nodes in the pattern in order to determine their new location (Line 10).

Algorithm 3.7: Share One Edge - Circle, Clique and Triangle

```

01: function DRAWPATTERN( $G_p$ )
02:    $G \leftarrow (V, E)$ 
03:    $drawnPatterns \leftarrow \{g \mid g \in G, \mathbf{drawn}(g)\}$ 
04:    $G_p \leftarrow (V_p, E_p)$  ▷ The pattern about to be drawn
05:    $e_{shared} \leftarrow$  The Shared Edge
06:    $originalEdgeLocation \leftarrow \mathbf{currentLocation}(e_{shared})$ 
07:    $originalEdgeAngle \leftarrow \mathbf{currentAngle}(e_{shared})$ 
08:    $originalEdgeSize \leftarrow \mathbf{currentLength}(e_{shared})$ 
09:    $originalEdgeCrossings \leftarrow \mathbf{edgeCrossings}(drawnPatterns)$ 
10:    $\mathbf{drawInIdealLayout}(G_p)$  ▷ Draw pattern in ideal layout
11:    $\mathbf{scalePattern}(G_p, originalEdgeSize)$ 
12:    $\mathbf{rotatePattern}(G_p, originalEdgeAngle)$ 
13:    $\mathbf{movePattern}(G_p, originalEdgeLocation)$ 
14:    $drawnPatterns \leftarrow drawnPatterns \cup G_p$ 
15:    $finalEdgeCrossings \leftarrow \mathbf{edgeCrossings}(drawnPatterns)$ 
16:   if  $originalEdgeCrossings \neq finalEdgeCrossings$  then ▷  $G_p$  may be drawn inside the drawn
      patterns
17:      $\mathbf{reflectPattern}(G_p, e_{shared})$  ▷ Reflect  $G_p$  along the shared edge
18:     if  $finalEdgeCrossings \leq \mathbf{edgeCrossings}(drawnPatterns)$  then ▷ If there are more edge
      crossings now, then the best location was the previous one
19:        $\mathbf{reflectPattern}(G_p, e_{shared})$  ▷ Reflect back

```

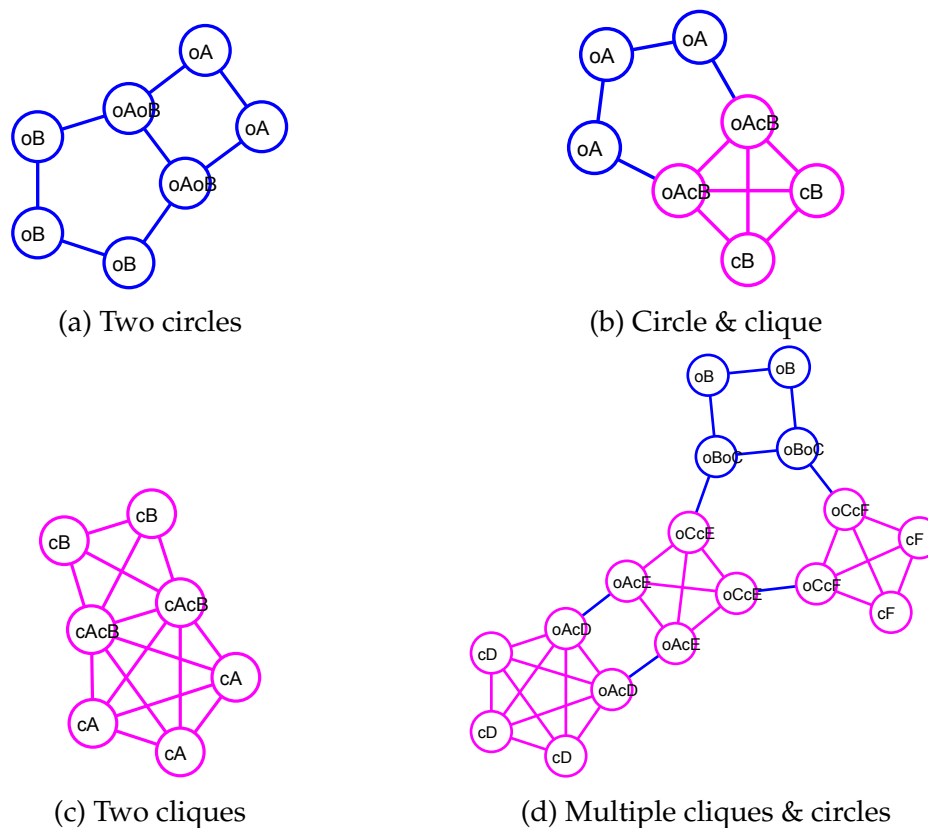


Figure 3.41: Examples of the One Edge Sharing algorithm for Circles & Cliques

3.5.1.3 Star

Although the ideal layout of stars is different, it follows a similar drawing method to that used for cliques and circles.

The ideal layout arranges stars at an appropriate distance from the centre. The spokes of stars are also evenly spaced around the centre. However, the ideal layout cannot always follow this format if other patterns cause occlusions. To overcome and make provision for this, the drawing algorithm identifies invalid areas which are used to restrict the range in which spokes can be drawn.

Once the best valid area has been identified according to the method detailed in Section 3.6.1, the star is drawn in its ideal layout. This layout arranges

the spokes of the star consistently within the valid area, excepting shared edges. Once the spokes have been arranged, the drawing is complete. An example of a star drawn with this method is shown in Figure 3.42. The process of drawing a star is described in Algorithm 3.8. The time complexity for this algorithm is $\mathcal{O}(V \cdot V_p)$, due to the algorithm iterating through all nodes in order to determine the valid drawing area (V , Line 9), and all nodes in the pattern in order to determine their new location (V_p , Line 15 or 17).

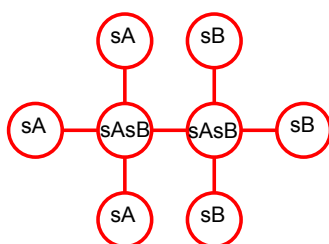


Figure 3.42: Example of the Edge Sharing algorithm for Stars

Algorithm 3.8: Share One Edge - Star

```

01: function DRAWPATTERN( $G_p$ )
02:    $G \leftarrow (V, E)$ 
03:    $nodePadding \leftarrow 10, anglePadding \leftarrow 5^\circ$ 
04:    $drawnPatterns \leftarrow \{g \mid g \in G, \mathbf{drawn}(g)\}$ 
05:    $G_p \leftarrow (V_p, E_p)$  ▷ The pattern about to be drawn
06:    $e_{shared} \leftarrow$  The Shared Edge
07:    $v_{centre} \leftarrow \mathbf{centreNodeOfStar}(G_p)$ 
08:    $drawableAreas \leftarrow \emptyset$  ▷ Empty list of Drawing Areas
09:   for each Node  $\{v \mid v \in V, v \notin V_p, \mathbf{distance}(v, v_{centre}) \leq \mathbf{length}(e_{shared}) + nodePadding\}$  do
10:      $angle \leftarrow \mathbf{angle}(v, v_{centre})$ 
11:      $drawableAreas \leftarrow drawableAreas \cup (angle - anglePadding, angle + anglePadding)$ 
12:    $drawableAreas \leftarrow \mathbf{JOININVALIDAREAS}(drawableAreas)$  ▷ See Algorithm 3.23
13:    $bestArea \leftarrow \mathbf{FINDBESTAREA}(drawableAreas)$  ▷ See Algorithm 3.24
14:   if  $drawableAreas \neq \emptyset$  then
15:      $\mathbf{drawInIdealLayout}(G_p, bestArea)$  ▷ Draw pattern in ideal layout in the  $bestArea$ 
16:   else
17:      $\mathbf{drawInIdealLayout}(G_p)$  ▷ Draw pattern in ideal layout
18:    $drawnPatterns \leftarrow drawnPatterns \cup G_p$ 

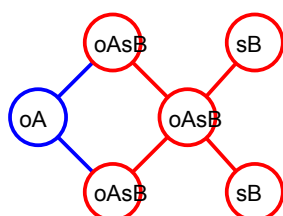
```

3.5.2 Share Two Edges

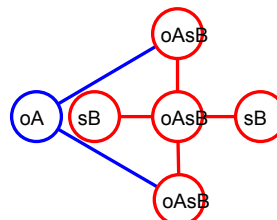
It is only possible for circles, stars and paths to share two edges. There are two separate methods for drawing circles sharing two edges, depending on if these edges are consecutive. Paths may also share two edges with the drawn set.

3.5.2.1 Consecutive Shared Edges

This scenario exists where a circle shares two spokes of a star. A decision had to be made regarding the order in which the star or circle should be drawn. One possible option is to draw the patterns in a similar way to Figure 3.43a. However, this requires changing the main drawing order (as stars are drawn before circles). It was felt that it was unwise to change the order, as this could have a large number of implications. It was also decided that it was not sensible to introduce an exception for this type, as that defeats the object of having a fixed drawing order. Therefore, the drawing order had to be such that circles are drawn after stars, which produces a similar layout to Figure 3.43b.



(a) Star drawn around a Circle



(b) Circle drawn around a Star

Figure 3.43: Possible drawing methods for patterns sharing two edges

The method follows a process similar to the One Edge Sharing drawing method (see Section 3.5.1.2) for Circles. Firstly, the middle node of the star is found and a virtual edge is created between the other two shared nodes (i.e. the outside nodes). This is used to enable compatibility with the One Edge Shared method. The pattern (with the exception of the centre node) is then drawn in its

ideal layout, centred on the origin of the graph (i.e. the point $(0,0)$). The circle is then scaled so that the virtual edge is the same length as it was before the circle was drawn. This ensures that the pattern does not distort the currently drawn set.

The pattern is then rotated to match the original orientation of the virtual edge and then moved into the correct location. Once this is complete, an orientation check is completed. A measure of the edge crossings and occlusion is taken (see Section 3.6.3). If there are no edge crossings or occlusion, then the drawing is complete. If not, the pattern is reflected along the virtual edge. Another calculation of the edge crossings and occlusion is taken. If this is lower than the previous calculation, then this layout is chosen. If not, the pattern is reflected back and the method is complete. This completes the algorithm, examples of which can be seen in Figure 3.44 and is detailed in Algorithm 3.9. The time complexity of this algorithm is $\mathcal{O}(V_p)$ as it must iterate through all nodes in the pattern in order to determine their new location (Line 13).

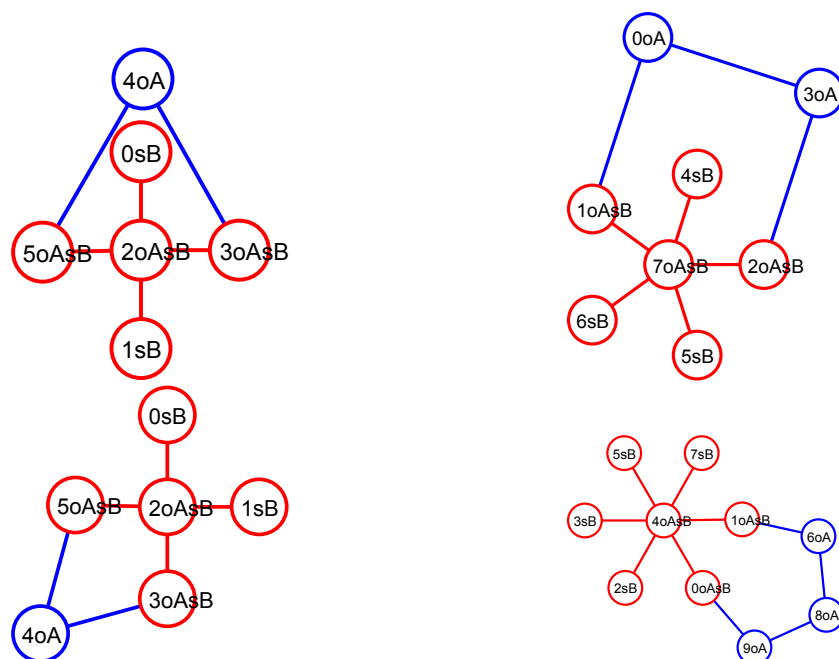


Figure 3.44: Examples of the Multiple Edge Sharing algorithm

Algorithm 3.9: Share Two Edges - Consecutive Shared Edges

```

01: function DRAWPATTERN( $G_p$ )
02:    $G \leftarrow (V, E)$ 
03:    $drawnPatterns \leftarrow \{g \mid g \in G, \mathbf{drawn}(g)\}$ 
04:    $G_p \leftarrow (V_p, E_p)$  ▷ The pattern about to be drawn
05:    $E_s \leftarrow \{e \mid e \in E, e \in E_p\}$  ▷ Shared Edges
06:    $v_m \leftarrow \{v \mid v \in V_p, \mathbf{connectedTo}(E_s[0]), \mathbf{connectedTo}(E_s[1])\}$  ▷ The middle node between the
two shared edges
07:    $middleNodeLocation \leftarrow \mathbf{location}(v_m)$ 
08:    $e_{virtual} \leftarrow (\mathbf{oppositeEnd}(E_s[0], v_m), \mathbf{oppositeEnd}(E_s[1], v_m))$  ▷ Virtual edge between the two
non-connected ends of the shared edges
09:    $originalEdgeLocation \leftarrow \mathbf{currentLocation}(e_{virtual})$ 
10:    $originalEdgeAngle \leftarrow \mathbf{currentAngle}(e_{virtual})$ 
11:    $originalEdgeSize \leftarrow \mathbf{currentLength}(e_{virtual})$ 
12:    $originalScore \leftarrow \mathbf{CALCULATESCORE}(drawnPatterns)$  ▷ See Algorithm 3.26
13:    $\mathbf{drawInIdealLayout}(G_p \setminus v_m)$  ▷ Draw pattern in ideal layout, ignoring  $v_m$ 
14:    $\mathbf{scalePattern}(G_p, originalEdgeSize)$ 
15:    $\mathbf{rotatePattern}(G_p, originalEdgeAngle)$ 
16:    $\mathbf{movePattern}(G_p, originalEdgeLocation)$ 
17:    $\mathbf{location}(v_m) \leftarrow middleNodeLocation$ 

```

```

18: | drawnPatterns ← drawnPatterns ∪ Gp
19: | finalScore ← CALCULATESCORE(drawnPatterns)           ▷ See Algorithm 3.26
20: | if originalScore ≠ finalScore then                   ▷ Gp may be drawn around the drawn patterns
21: | | reflectPattern(Gp, evirtual)                       ▷ Reflect Gp along the shared edge
22: | | if finalScore ≤ CALCULATESCORE(drawnPatterns) then ▷ If there is a worse score now, then
the best location was the previous one, See Algorithm 3.26
23: | | | reflectPattern(Gp, evirtual)                       ▷ Reflect back

```

3.5.2.2 Non-Consecutive Shared Edges

When the shared edges are not consecutive, a different approach is taken. Firstly, all the nodes between the two shared edges are identified (i.e. in Figure 3.45, nodes 8-oA-oC & 11-oA-oC (highlighted in green) and 13-oB-oC & 15-oB-oC (highlighted in yellow)). These nodes are then drawn in a straight line between the shared edges. Once drawn, these are adjusted in a similar way to paths (see Section 3.5.8), with several curves being attempted before the curve with the lowest score is accepted. The algorithm is detailed in Algorithm 3.10, with a time complexity of $\mathcal{O}(V_p)$, due to the algorithm iterating through all nodes in the pattern several times order to determine the correct location (Lines 45 and 46).

Algorithm 3.10: Share Two Edges — Non-Consecutive Shared Edges

```

01: function DRAWPATTERN(Gp)
02: | G ← (V, E)
03: | drawnPatterns ← {g | g ∈ G, drawn(g)}
04: | Gp ← (Vp, Ep)           ▷ The pattern about to be drawn
05: | Es ← {e | e ∈ E, e ∈ Ep}   ▷ Shared Edges
06: | Pair1, Pair2 ← (null, null)  ▷ Node Pairs for use later
07: | Vm1, Vm2 ← []             ▷ Nodes between these pairs
08: | iA ← indexOf(Vp, startNode(Es[0]))
09: | iB ← indexOf(Vp, endNode(Es[0]))
10: | iC ← indexOf(Vp, startNode(Es[1]))
11: | iD ← indexOf(Vp, endNode(Es[1]))

```

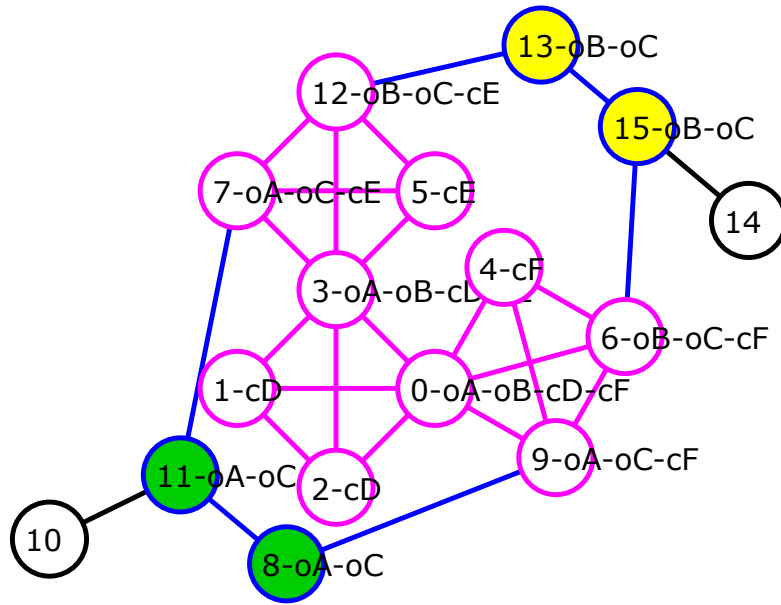


Figure 3.45: Example of Two Shared Non-Consecutive Edges

```

12:   if  $((i_A + 1) \bmod |V_p|) = i_B$  then                                     ▷ Search forwards
13:        $i \leftarrow i_B$ 
14:       while  $(i \bmod |V_p|) \neq i_A$  do
15:            $V_{m_1} \leftarrow V_{m_1} \cup V_p[i \bmod |V_p|]$ 
16:           if  $(i \bmod |V_p|) = i_C$  then
17:                $Pair_1 \leftarrow (V_p[i_B], V_p[i_C])$ 
18:                $Pair_2 \leftarrow (V_p[i_D], V_p[i_A])$ 
19:                $V_{m_2} \leftarrow \{v \mid v \in V_p, v \notin V_{m_1}\}$ 
20:               Break while
21:           else if  $(i \bmod |V_p|) = i_D$  then
22:                $Pair_1 \leftarrow (V_p[i_B], V_p[i_D])$ 
23:                $Pair_2 \leftarrow (V_p[i_C], V_p[i_A])$ 
24:                $V_{m_2} \leftarrow \{v \mid v \in V_p, v \notin V_{m_1}\}$ 
25:               Break while
26:            $i \leftarrow i + 1$ 
27:   else                                                                       ▷ Search backwards
28:        $i \leftarrow i_B + |V_p|$ 
29:       while  $(i \bmod |V_p|) \neq i_A$  do
30:            $V_{m_1} \leftarrow V_{m_1} \cup V_p[i \bmod |V_p|]$ 
31:           if  $(i \bmod |V_p|) = i_C$  then
32:                $Pair_1 \leftarrow (V_p[i_C], V_p[i_B])$ 
33:                $Pair_2 \leftarrow (V_p[i_A], V_p[i_D])$ 
34:                $V_{m_1} \leftarrow \text{reverseOrder}(V_{m_1})$ 
35:                $V_{m_2} \leftarrow \{v \mid v \in V_p, v \notin V_{m_1}\}$ 

```

▷ Searching backwards, but drawing forwards

```

36: | | | Break while
37: | | | else if (i mod |Vp|) = iD then
38: | | |   Pair1 ← (Vp[iD], Vp[iB])
39: | | |   Pair2 ← (Vp[iA], Vp[iC])
40: | | |   Vm1 ← reverseOrder(Vm1)           ▷ Searching backwards, but drawing forwards
41: | | |   Vm2 ← {v | v ∈ Vp, v ∉ Vm1}
42: | | |   Break while
43: | | |   i ← i - 1
44: | drawnPatterns ← drawnPatterns ∪ Gp
45: | DRAWBETWEENNODES(Pair1, Vm1)           ▷ See Below
46: | DRAWBETWEENNODES(Pair2, Vm2)           ▷ See Below
47:
48: function DRAWBETWEENNODES((vstart, vfinish), Vm)
49: | G ← (V, E)
50: | drawnPatterns ← {g | g ∈ G, drawn(g)}
51: | Gp ← (Vp, Ep)           ▷ The pattern about to be drawn
52: | bestScore ← CALCULATESCORE(drawnPatterns)   ▷ See Algorithm 3.26
53: | bestLayout ← 0
54: | Δx ← (vfinishx - vstartx) ÷ (|Vm| - 1)
55: | Δy ← (vfinishy - vstarty) ÷ (|Vm| - 1)
56: | for each v ∈ Vm, {i | i ≥ 0, i < |Vm|} do   ▷ Draw nodes in a straight line
57: | | vx ← vstartx + (i × Δx)
58: | | vy ← vstarty + (i × Δy)
59: | | oldLocation(n) ← currentLocation(n)
60: | | for j ∈ {1, 2, ..., 6} do           ▷ 7 different locations to test (Location 0 is the current one)
61: | | | TRYPATHLOCATIONS(vstart, vfinish, Vm, j)   ▷ See Algorithm 3.19
62: | | | currentScore ← CALCULATESCORE(drawnPatterns)   ▷ See Algorithm 3.26
63: | | | if currentScore < bestScore then
64: | | | | bestLayout ← j
65: | | | | bestScore ← currentScore
66: | | TRYPATHLOCATIONS(vstart, vfinish, Vm, bestLocation)   ▷ Draw in best location. See Algorithm 3.19

```

3.5.3 Share One Node

Circles, cliques, triangles and stars (when sharing an outside node) all share the same drawing algorithm (Section 3.5.3.2) when patterns share one node with the drawn set. Stars which have their centre node shared (Section 3.5.3.3) and paths (Section 3.5.8) have separate methods.

3.5.3.1 Node Swapping

Nodes may be swapped in the pattern to produce a better layout. For example, the order of the spokes of a star or nodes in a clique are irrelevant and by changing this order, a better drawing result may be obtained. Therefore, a list of potential swaps is generated. If the shared node has more than one drawn pattern connecting to it, then the current order is used. This prevents any already drawn nodes from being moved. If not, then all nodes in the already drawn pattern which also have only one drawn pattern (and not the centre of a star) are identified. For each of these potential swapping nodes, the location of the shared node and swapping node are switched, and the pattern is drawn according to the algorithms detailed in Sections 3.5.3.2, 3.5.3.3 and 3.5.8. Once drawn, a score is calculated based on node-node occlusion, edge-node occlusion, edge crossings and symmetry (Section 3.6.3). When all options have been tested, the swap with the best score is used for the final layout.

The process of swapping nodes is described in Algorithm 3.11. The time complexity for this algorithm is dependent on the type of pattern being drawn. If drawing a star sharing its centre node, the complexity is $\mathcal{O}(\log(V) + (V \cdot V_p))$, if drawing a clique the complexity is $\mathcal{O}(\log V + V_p)$, or if drawing all other pattern types the complexity is $\mathcal{O}(V_p)$. This is due to the algorithm iterating through some nodes in the graph to find swaps ($\log(V)$, Line 11 or 13), if needed, and the time complexity of Algorithm 3.12 (V_p , Line 39) or Algorithm 3.13 ($V \cdot V_p$, Line 41).

Algorithm 3.11: Node Swapping (Share One Node)

```

01: function NODESWAPPINGSHAREONENODE
02:    $G \leftarrow (V, E)$ 
03:    $G_p \leftarrow (V_p, E_p)$  ▷ The pattern about to be drawn
04:    $v_s \leftarrow$  Shared Node
05:    $G_{op} \leftarrow \mathbf{drawnPatterns}(v_s)$  ▷ All drawn patterns sharing the shared node

```

```

06:   if  $|G_{op}| = 1$  then
07:        $G_{other} \leftarrow G_{op}[0]$  ▷ The other pattern
08:       if isStar( $G_{other}$ ) OR isClique( $G_{other}$ ) then
09:            $V_{swaps} \leftarrow \emptyset$  ▷ Swapping Nodes
10:           if isStar( $G_{other}$ ) then
11:                $V_{swaps} \leftarrow \{v \mid v \in V_{other}, \mathbf{drawnPatterns}(n) = 1, v \neq \mathbf{centreNode}(G_{other})\}$ 
12:           else if isClique( $G_{other}$ ) then
13:                $V_{swaps} \leftarrow \{v \mid v \in V_{other}, \mathbf{drawnPatterns}(v) = 1\}$ 
14:           if  $V_{swaps} \neq \emptyset$  then
15:                $bestScore \leftarrow \infty$ 
16:                $bestSymmetry \leftarrow \infty$ 
17:                $bestSwap \leftarrow null$ 
18:               for each  $v \in V_{swaps}$  do
19:                   swapNodes( $v, v_s$ ) ▷ Swap the nodes
20:                   DRAWASREQUIRED( $G_p, v$ ) ▷ See Below
21:                    $currentScore \leftarrow \mathbf{CALCULATESCORE}(\mathbf{drawnPatterns})$  ▷ See Algorithm 3.26
22:                    $currentSymmetry \leftarrow \mathbf{SYMMETRYMEASURE}(\mathbf{drawnPatterns})$  ▷ See Algorithm 3.25
23:                   if ( $currentScore < bestScore$ ) OR ( $currentScore = bestScore$  AND  $currentSymmetry <$ 
 $bestSymmetry$ ) then
24:                        $bestScore \leftarrow currentScore$ 
25:                        $bestSymmetry \leftarrow currentSymmetry$ 
26:                        $bestSwap \leftarrow (v, v_s)$ 
27:                       swapNodes( $v, v_s$ ) ▷ Swap back
28:                       swapNodes( $bestSwap$ ) ▷ Swap the best nodes
29:                       DRAWASREQUIRED( $G_p, v$ ) ▷ See Below
30:                   else
31:                       DRAWASREQUIRED( $G_p, v$ ) ▷ See Below
32:           else
33:               DRAWASREQUIRED( $G_p, v$ ) ▷ See Below
34:       else
35:           DRAWASREQUIRED( $G_p, v$ ) ▷ See Below
36:
37: function DRAWASREQUIRED( $G_p, v$ )
38:     if isStar( $G_p$ ) AND centreNode( $G_p$ ) =  $v_s$  then
39:         DRAWPATTERN( $G_p$ ) ▷ Draw (Star (Centre Node)). See Algorithm 3.13
40:     else
41:         DRAWPATTERN( $G_p$ ) ▷ Draw (All except Star (Centre Node)). See Algorithm 3.12

```

3.5.3.2 Circles, Cliques, Triangles & Stars (Outside Node)

Circles, cliques, triangles and stars that share an outside node all have the same algorithm for drawing. In this algorithm, the pattern is first drawn in its ideal

layout, centred on the origin of the graph, before being moved so that the shared node is back in its original location.

A search based approach is then taken in order to find the best layout. The pattern is rotated 5° and a fitness score based on the number of occlusions and edge crossings is calculated and compared to the current best. If the new score is lower, then rotation is chosen. If this rotation is equal to the previous best, then a symmetry score is calculated and the more symmetrical layout is chosen. The rotation step of 5° is chosen to reduce the number of iterations of this method and to avoid the need for a complicated algorithm which has no knowledge of any other patterns that may be cause occlusion or edge crossings. The pattern is then reflected along the line between the shared node and the centre of the pattern and the same calculations and comparisons are performed. The process then continues until the pattern has completed a full rotation and the pattern is then redrawn with the best rotation and reflection found. This completes the algorithm, examples of which can be seen in Figure 3.46 and which is detailed in Algorithm 3.12. This algorithm has a time complexity of $\mathcal{O}(V_p)$, due to the algorithm iterating through all nodes in the pattern several times order to determine the correct location (Lines 7 and 32).

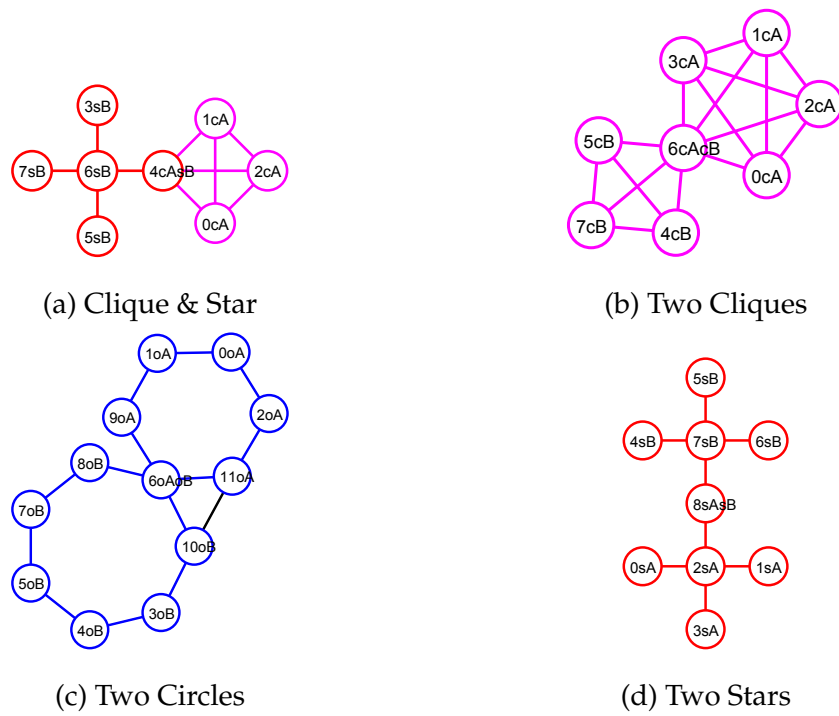


Figure 3.46: Examples of the Shared Node algorithm for Cliques, Circles, and Stars (outside node)

**Algorithm 3.12: Share One Node - Circle,
Clique, Triangle, Star (Outside)**

```

01: function DRAWPATTERN( $G_p$ )
02:    $G \leftarrow (V, E)$ 
03:    $drawnPatterns \leftarrow \{g \mid g \in G, \mathbf{drawn}(g)\}$ 
04:    $G_p \leftarrow (V_p, E_p)$ 
05:    $V_s \leftarrow \{v \mid v \in V, v \in V_p\}$ 
06:    $originalLocation \leftarrow \mathbf{currentLocation}(v_s)$ 
07:    $\mathbf{drawInIdealLayout}(G_p)$ 
08:    $drawnPatterns \leftarrow drawnPatterns \cup G_p$ 
09:    $\mathbf{movePattern}(G_p, originalLocation)$ 
10:    $bestScore \leftarrow \mathbf{CALCULATESCORE}(drawnPatterns)$ 
11:    $bestAngle \leftarrow 0$ 
12:    $bestReflection \leftarrow \mathbf{false}$ 
13:    $bestSymmetry \leftarrow \mathbf{SYMMETRYMEASURE}(drawnPatterns)$ 
14:   for  $i \leftarrow \{5, 10, 15, \dots, 355\}$  do
15:      $\mathbf{rotatePattern}(G_p, i)$ 

```

▷ The pattern about to be drawn
 ▷ The shared node
 ▷ Draw pattern in ideal layout
 ▷ See Algorithm 3.26
 ▷ See Algorithm 3.25
 ▷ Rotate Pattern

```

16: |   newScore ← CALCULATESCORE(drawnPatterns)           ▷ See Algorithm 3.26
17: |   newSymmetry ← SYMMETRYMEASURE(drawnPatterns)       ▷ See Algorithm 3.25
18: |
19: |   if (newScore < bestScore) OR (newScore = bestScore AND newSymmetry < bestSymmetry)
    then
20: |       bestScore ← newScore
21: |       bestAngle ← i
22: |       bestReflection ← false
23: |       bestSymmetry ← newSymmetry
24: |       reflectPattern(Gp, (ns, centreOf(Gp)))           ▷ Reflect pattern
25: |       if (newScore < bestScore) OR (newScore = bestScore AND newSymmetry < bestSymmetry)
    then
26: |           bestScore ← newScore
27: |           bestAngle ← i
28: |           bestReflection ← true
29: |           bestSymmetry ← newSymmetry
30: |           reflectPattern(Gp, (ns, centreOf(Gp)))           ▷ Reflect back
31: |           rotatePattern(Gp, -i)                         ▷ Rotate back to reduce compound rounding errors
32: |           drawInIdealLayout(Gp)                       ▷ Draw pattern again, to mitigate any rounding errors
33: |           movePattern(Gp, originalLocation)           ▷ Move pattern back
34: |           rotatePattern(Gp, bestAngle)                 ▷ Rotate Pattern
35: |           if bestReflection then
36: |               reflectPattern(Gp, (ns, centreOf(Gp)))           ▷ Reflect pattern

```

3.5.3.3 Stars (Centre Node)

Stars that share their central node with another pattern are drawn differently to stars sharing an outside node.

The method for drawing stars is similar to that for stars sharing an edge (Section 3.5.1.3). The largest valid area for drawing is calculated, using the method described in Section 3.6.1 and once found, the nodes of the star are drawn evenly spaced in this area. This algorithm is detailed in Algorithm 3.13 and an example given in Figure 3.47. This algorithm has a time complexity of $\mathcal{O}(V + V_p)$, due to the algorithm iterating through all nodes in the graph to find the best drawable area (V , Line 10), and the nodes in the pattern to determine their new location (V_p , Line 16 or 18).

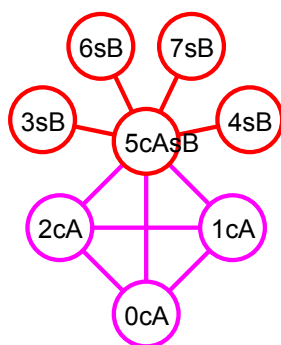


Figure 3.47: Example of Shared Node algorithm for Stars (centre node)

Algorithm 3.13: Share One Node - Star (Centre Node)

```

01: function DRAWPATTERN( $G_p$ )
02:    $G \leftarrow (V, E)$ 
03:    $drawnPatterns \leftarrow \{g \mid g \in G, \mathbf{drawn}(g)\}$ 
04:    $G_p \leftarrow (V_p, E_p)$  ▷ The pattern about to be drawn
05:    $V_s \leftarrow \{v \mid v \in V, v \in V_p\}$  ▷ The shared node
06:    $v_c \leftarrow \mathbf{centreNodeOfStar}(G_p)$  ▷ The centre node
07:    $drawnPatterns \leftarrow drawnPatterns \cup G_p$ 
08:    $v_{centre} \leftarrow \mathbf{centreNodeOfStar}(G_p)$ 
09:    $drawableAreas \leftarrow \emptyset$  ▷ Empty list of Drawing Areas
10:   for each Node  $\{v \mid v \in V, v \notin V_p, \mathbf{distance}(v, v_{centre}) \leq (|V_p| \times 20) + \mathbf{nodePadding}\}$  do
11:      $angle \leftarrow \mathbf{angle}(v, v_{centre})$ 
12:      $drawableAreas \leftarrow drawableAreas \cup (angle - \mathbf{anglePadding}, angle + \mathbf{anglePadding})$ 
13:    $drawableAreas \leftarrow \mathbf{JOININVALIDAREAS}(drawableAreas)$  ▷ See Algorithm 3.23
14:    $bestArea \leftarrow \mathbf{FINDBESTAREA}(drawableAreas)$  ▷ See Algorithm 3.24
15:   if  $drawableAreas \neq \emptyset$  then
16:      $\mathbf{drawInIdealLayout}(G_p, bestArea)$  ▷ Draw pattern in ideal layout in the  $bestArea$ 
17:   else
18:      $\mathbf{drawInIdealLayout}(G_p)$  ▷ Draw pattern in ideal layout
19:     ▷ Check if pattern is inside the drawn set
20:    $bestScore \leftarrow \mathbf{CALCULATESCORE}(drawnPatterns)$  ▷ See Algorithm 3.26
21:    $bestSymmetry \leftarrow \mathbf{SYMMETRYMEASURE}(drawnPatterns)$  ▷ See Algorithm 3.25
22:    $\mathbf{rotatePattern}(G_p, 180^\circ)$  ▷ Rotate Pattern
23:    $newScore \leftarrow \mathbf{CALCULATESCORE}(drawnPatterns)$  ▷ See Algorithm 3.26
24:    $newSymmetry \leftarrow \mathbf{SYMMETRYMEASURE}(drawnPatterns)$  ▷ See Algorithm 3.25
25:   if  $newScore > bestScore$  OR  $(newScore = bestScore \text{ AND } newSymmetry \geq bestSymmetry)$  then
26:      $\mathbf{rotatePattern}(G_p, 180^\circ)$  ▷ Rotate back

```

3.5.4 Share Two Nodes

It is only possible for circles, triangles and paths to share two nodes, but no edges.

3.5.4.1 Circles

The method for drawing circles with this type of connection is very similar to patterns which share one edge. A imaginary edge is created between the two shared nodes, and the process runs as in Section 3.5.1.2. In the example of Figure 3.48 below, the imaginary shared edge is represented with a dashed line, while the other edges ensure no stars are created. The algorithm is defined below in Algorithm 3.14 and has a time complexity of $\mathcal{O}(V_p)$, due to the algorithm iterating through all nodes in the pattern several times order to determine the correct location (Line 11).

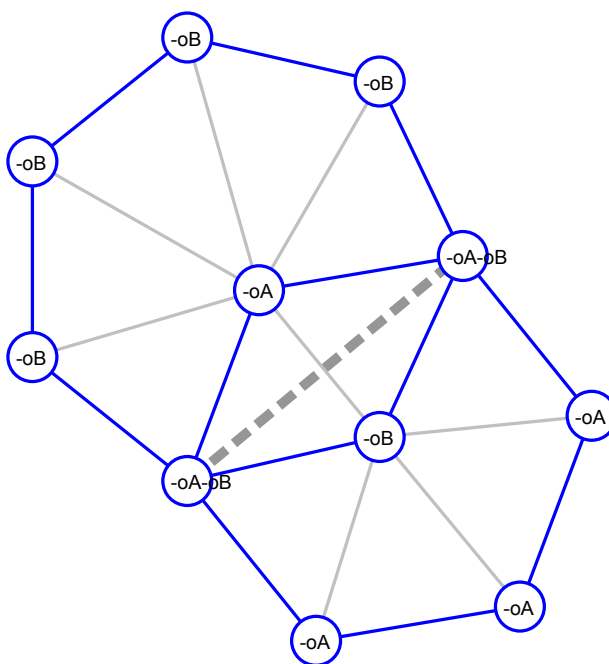


Figure 3.48: Example of two circles sharing two nodes

Algorithm 3.14: Share Two Nodes - Circle

```

01: function DRAWPATTERN( $G_p$ )
02:    $G \leftarrow (V, E)$ 
03:    $drawnPatterns \leftarrow \{g \mid g \in G, \mathbf{drawn}(g)\}$ 
04:    $G_p \leftarrow (V_p, E_p)$  ▷ The pattern about to be drawn
05:    $V_s \leftarrow \{v \mid v \in V, v \in V_p\}$  ▷ Shared Nodes
06:    $V_{other} \leftarrow \{v \mid v \in V_p, v \notin V_s\}$ 
07:    $e_v \leftarrow (V_s[0], V_s[1])$  ▷ Create a virtual edge
08:    $oldLocation \leftarrow \mathbf{currentLocation}(e_v)$ 
09:    $oldAngle \leftarrow \mathbf{currentAngle}(e_v)$ 
10:    $oldLength \leftarrow \mathbf{length}(e_v)$ 
11:    $\mathbf{drawInIdealLayout}(G_p)$  ▷ Draw pattern in ideal layout
12:    $drawnPatterns \leftarrow drawnPatterns \cup G_p$ 
13:    $\mathbf{scalePattern}(G_p, oldLength)$ 
14:    $\mathbf{rotatePatern}(G_p, oldAngle)$ 
15:    $\mathbf{movePattern}(G_p, oldLocation)$ 
16:    $finalScore \leftarrow \mathbf{CALCULATESCORE}(drawnPatterns)$  ▷ See Algorithm 3.26
17:   if  $originalScore \neq finalScore$  then ▷  $G_p$  may be drawn around the drawn patterns
18:      $\mathbf{reflectPattern}(G_p, e_v)$  ▷ Reflect  $G_p$  along the virtual edge
19:     if  $finalScore \leq \mathbf{CALCULATESCORE}(drawnPatterns)$  then ▷ If there is a worse score now, then
the best location was the previous one, See Algorithm 3.26
20:      $\mathbf{reflectPattern}(G_p, e_v)$  ▷ Reflect back

```

3.5.4.2 Triangles

If a triangle shares two nodes, there only remains one node left to draw. This node is drawn in such a way to make an isosceles triangle. This location is then tested by altering the distance of this node and reflecting it along the line between the two other nodes. The best location (determined by the edge crossing and occlusion score) is found and this is where the remaining node is drawn. The algorithm is defined below in Algorithm 3.15 and has a time complexity of $\mathcal{O}(V_p)$, due to the algorithm iterating through all nodes in the pattern several times order to determine the correct location (Line 13).

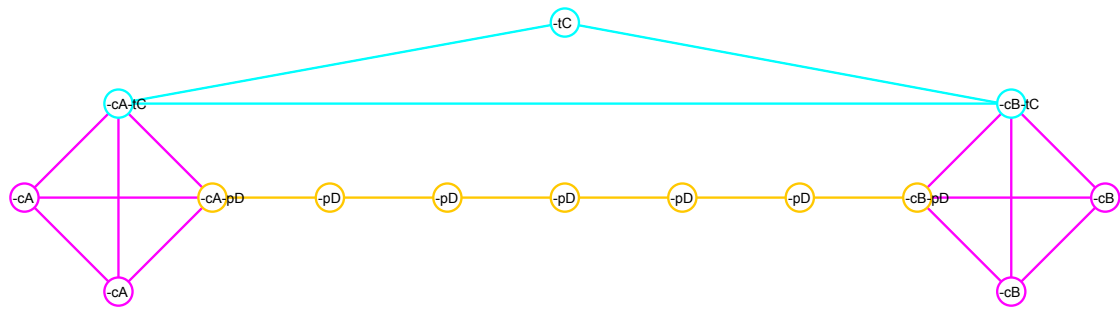


Figure 3.49: Example of a triangle sharing two nodes

Algorithm 3.15: Share Two Nodes - Triangles

```

01: function DRAWPATTERN( $G_p$ )
02:    $G \leftarrow (V, E)$ 
03:    $drawnPatterns \leftarrow \{g \mid g \in G, \mathbf{drawn}(g)\}$ 
04:    $G_p \leftarrow (V_p, E_p)$  ▷ The pattern about to be drawn
05:    $V_s \leftarrow \{v \mid v \in V, v \in V_p\}$  ▷ Shared Nodes
06:    $v_{other} \leftarrow \{v \mid v \in V_p, v \notin V_s\}$ 
07:    $v_{other_x} \leftarrow (V_s[0]_x + V_s[1]_x) \div 2$ 
08:    $v_{other_y} \leftarrow (V_s[0]_y + V_s[1]_y) \div 2$ 
09:    $drawnPatterns \leftarrow drawnPatterns \cup \{G_p\}$ 
10:    $bestLayout \leftarrow 0$ 
11:    $bestScore \leftarrow \infty$  ▷ The current layout has node-edge occlusion
12:   for  $j \in \{1, 2, \dots, 6\}$  do ▷ 7 different locations to test (Location 0 is the current one)
13:     TRYPATHLOCATIONS( $v_{start}, v_{finish}, v_{other}, j$ ) ▷ See Algorithm 3.19
14:      $currentScore \leftarrow \text{CALCULATESCORE}(drawnPatterns)$  ▷ See Algorithm 3.26
15:     if  $currentScore < bestScore$  then
16:        $bestLayout \leftarrow j$ 
17:        $bestScore \leftarrow currentScore$ 
18:   TRYPATHLOCATIONS( $n_{start}, n_{finish}, n_{other}, bestLocation$ ) ▷ Draw in best location. See

```

Algorithm 3.19

3.5.5 Connected by an Edge

The algorithm for drawing patterns connected to the drawn set of patterns by one edge is the same for every pattern (Section 3.5.5.2).

3.5.5.1 Node Swapping

Nodes may be swapped in the pattern to produce a better layout. For example, the order of nodes in a clique or spokes in a star is irrelevant and by changing this order, a better drawing result may be obtained. Therefore, a list of potential swaps is generated. If the total number of patterns that share the start and end nodes of the connecting edge is exactly 2 (i.e. the already drawn pattern and the one about to be drawn), then swapping is possible. If not, then the current order is used, as this prevents any already drawn nodes from being moved.

Both cliques and stars share the same method for determining the potential nodes to be swapped. A list of nodes in the drawn pattern which only have one pattern (i.e. this drawn pattern) are identified as potential swaps (the centre of stars is also excluded). There is no need to swap nodes within the pattern about to be drawn, as this will be rotated and reflected as part of the drawing procedure.

For each of these potential swapping nodes, the location is swapped with that of the connecting node (i.e. the node at which the connecting edge connects to the pattern about to be drawn). The pattern is then drawn according to the methods detailed in Sections 3.5.5.2 and 3.5.8. Once drawn, a score is calculated based on node-node occlusion, edge-node occlusion, edge crossings and symmetry (Section 3.6.3). When all options have been tested, the swap with the best score is used for the final layout.

The process of swapping nodes is described in Algorithm 3.16. The time complexity for this algorithm is dependent on the type of pattern being drawn.

If drawing a star or clique, the complexity is $\mathcal{O}(\log(V) + V_p + \log E)$, or if drawing all other pattern types the complexity is $\mathcal{O}(V_p + \log E)$. This is due to the algorithm iterating through some nodes in the graph to find swaps ($\log(V)$, Line 20), if needed, and the time complexity of Algorithm 3.17 ($V_p + \log E$, Lines 22 and 31–37).

Algorithm 3.16: Node Swapping (Connected By an Edge)

```

01: function NODESWAPPINGEDGECONNECTING
02:    $G \leftarrow (V, E)$ 
03:    $G_p \leftarrow (V_p, E_p)$  ▷ The pattern about to be drawn
04:    $e_c \leftarrow$  Connecting Edge
05:    $v_c \leftarrow \{v \mid v \in V_p, e_c \in \text{connectingEdges}(v)\}$  ▷ Attaching node
06:    $G_{op} \leftarrow \text{patterns}(\text{startNode}(e_c)) \cup \text{patterns}(\text{endNode}(e_c))$  ▷ All patterns sharing the start and
    end nodes of the connecting edge
07:    $drawnPatterns \leftarrow drawnPatterns \cup G_p$ 
08:   if  $|G_{op}| = 2$  then ▷ Swapping can only occur if there's one other pattern sharing the nodes at
    either end of this edge
09:      $G_{other} \leftarrow \{g \mid g \in G_{op}, g \neq G_p\}$  ▷ The other pattern
10:     if isStar( $G_{other}$ ) OR isClique( $G_{other}$ ) then
11:        $N_{swaps} \leftarrow \emptyset$  ▷ Swapping Nodes
12:       if isStar( $G_{other}$ ) then
13:          $V_{swaps} \leftarrow \{v \mid v \in V_{other}, \text{drawnPatterns}(v) = 1, v \neq \text{centreNode}(G_{other})\}$ 
14:       else if isClique( $G_{other}$ ) then
15:          $V_{swaps} \leftarrow \{v \mid v \in V_{other}, \text{drawnPatterns}(v) = 1\}$ 
16:       if  $V_{swaps} \neq \emptyset$  then
17:          $bestScore \leftarrow \infty$ 
18:          $bestSymmetry \leftarrow \infty$ 
19:          $bestSwap \leftarrow \text{null}$ 
20:         for each  $v \in V_{swaps}$  do
21:           swapNodes( $v, v_c$ ) ▷ Swap the nodes
22:           DRAWPATTERN( $G_p$ ) ▷ Draw pattern (See Algorithm 3.17)
23:            $currentScore \leftarrow \text{CALCULATESCORE}(drawnPatterns)$  ▷ See Algorithm 3.26
24:            $currentSymmetry \leftarrow \text{SYMMETRYMEASURE}(drawnPatterns)$  ▷ See Algorithm 3.25
25:           if ( $currentScore < bestScore$ ) OR ( $currentScore = bestScore$  AND  $currentSymmetry <$ 
     $bestSymmetry$ ) then
26:              $bestScore \leftarrow currentScore$ 
27:              $bestSymmetry \leftarrow currentSymmetry$ 
28:              $bestSwap \leftarrow (v, v_c)$ 
29:           swapNodes( $v, v_c$ ) ▷ Swap back
30:           swapNodes( $bestSwap$ ) ▷ Swap the best nodes
31:           DRAWPATTERN( $G_p$ ) ▷ Draw pattern (See Algorithm 3.17)

```



```

32: | | | else
33: | | | DRAWPATTERN( $G_p$ )           ▷ Draw pattern (See Algorithm 3.17)
34: | | else
35: | | DRAWPATTERN( $G_p$ )           ▷ Draw pattern (See Algorithm 3.17)
36: | else
37: | DRAWPATTERN( $G_p$ )           ▷ Draw pattern (See Algorithm 3.17)

```

3.5.5.2 Cliques, Circles, Triangles & Stars

The algorithm for drawing a pattern connected by an edge to the drawn set utilises a search based approach. A grid of possible options is created and for each of this options, various combinations of rotations and reflections are tested. Numerous metrics are calculated to represent this and each is then tested in the order below:

1. Distance between the pattern and the currently drawn set
2. The sum of the number of edge crossings in the graph and any occluded nodes (See Section 3.6.3)
3. Length of the connecting edge
4. A measure of symmetry (See Section 3.6.2)

For a combination to be considered, the separation distance must be larger than a constant value. If it is valid for consideration, the total of the number of edge crossings and occluded nodes are examined and if lower than the previous best, the current combination is considered to be the current best.

However, the values may be equal. When this happens, the length of the connecting edge is compared. If this length is less than the previous best length, this combination is ranked as the best. If this metric is also equal, then a calculation of symmetry is made (see Section 3.6.2). If this measure is lower (i.e. the

pattern is more symmetrical to the currently drawn patterns), then this combination is considered to be the current best.

This process is repeated a further two times (reflection is not performed after the first iteration) with the search space being a factor of ten smaller each time (See Figure 3.50). This means that the spacing between each combination and the overall search space is ten times smaller. Once this process is complete, the pattern is drawn in the best location and combination of rotation and reflection. The algorithm is defined below in Algorithm 3.17 and has a time complexity of $\mathcal{O}(V_p + \log E)$, due to the algorithm iterating through all nodes in the pattern several times order to determine their new location (V_p , Line 26), and to identify the length of connecting edges ($\log E$, Line 94).

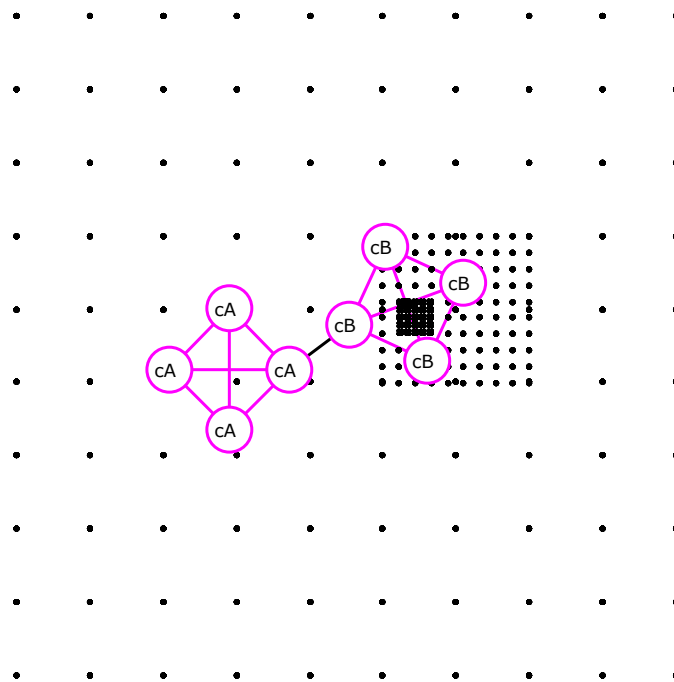


Figure 3.50: Example of the grid based search algorithm

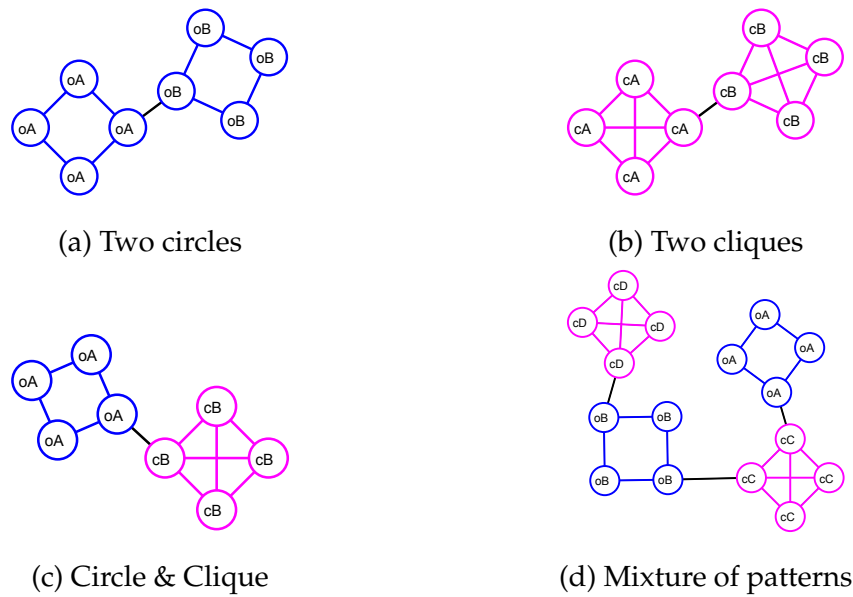


Figure 3.51: Example of the Edge Connected algorithm

**Algorithm 3.17: Connected By ≥ 1 Edges -
Clique, Circle, Triangle, Star**

```

01: function DRAWPATTERN( $G_p$ )
02:    $G \leftarrow (V, E)$ 
03:    $G_p \leftarrow (V_p, E_p)$ 
04:    $E_c \leftarrow$  Connecting Edges
05:    $V_c \leftarrow \{v \mid v \in V_p, e_c \in \text{connectingEdges}(v)\}$ 
06:    $gridPoints \leftarrow 10$ 
07:    $rotationPoints \leftarrow 8$ 
08:    $min_{separation} \leftarrow 50$ 
09:    $padding_x \leftarrow \text{width}(G_p) + 100$ 
10:    $padding_y \leftarrow \text{height}(G_p) + 100$ 
11:    $min_x \leftarrow \text{minX}(G_p) - padding_x$ 
12:    $max_x \leftarrow \text{maxX}(G_p) + padding_x$ 
13:    $min_y \leftarrow \text{minY}(G_p) - padding_y$ 
14:    $max_y \leftarrow \text{maxY}(G_p) + padding_y$ 
15:    $min_{angle} \leftarrow 0^\circ$ 
16:    $max_{angle} \leftarrow 360^\circ$ 
17:    $width \leftarrow max_x - min_x$ 
18:    $height \leftarrow max_y - min_y$ 
19:    $angleSweep \leftarrow max_{angle} - min_{angle}$ 

```

▷ The pattern about to be drawn

▷ Attaching nodes

▷ Define search parameters

```

20:   $gridSpacing_x \leftarrow width \div (gridPoints - 1)$ 
21:   $gridSpacing_y \leftarrow height \div (gridPoints - 1)$ 
22:   $angleIncrease \leftarrow angleSweep \div rotationPoints$ 
23:   $best_x \leftarrow -1$ 
24:   $best_y \leftarrow -1$ 
25:   $best_{rotation} \leftarrow 0$ 
26:  drawPatternInIdealLayout( $G_p$ )
27:   $drawnPatterns \leftarrow drawnPatterns \cup G_p$ 
28:  PERFORMSEARCH(true) ▷ Perform a search. See Below
29:  while  $gridSpacing_x \geq 10$  OR  $gridSpacing_y \geq 10$  do
30:       $width \leftarrow width \div (gridPoints - 1)$  ▷ Resize search area
31:       $height \leftarrow height \div (gridPoints - 1)$ 
32:       $angleIncrease \leftarrow angleSweep \div rotationPoints$ 
33:       $min_x \leftarrow best_x - (width \div 2)$ 
34:       $max_x \leftarrow best_x + (width \div 2)$ 
35:       $min_y \leftarrow best_y - (height \div 2)$ 
36:       $max_y \leftarrow best_y + (height \div 2)$ 
37:      if  $|N_c| = 1$  then
38:           $angleSweep \leftarrow angleSweep \div 3$ 
39:           $min_{angle} \leftarrow best_{rotation} - (angleSweep \div 2)$ 
40:           $max_{angle} \leftarrow best_{rotation} + (angleSweep \div 2)$ 
41:      PERFORMSEARCH(false) ▷ Perform a search. See Below
42:
43:  function PERFORMSEARCH(firstIteration)
44:       $best_x \leftarrow -1$ 
45:       $best_y \leftarrow -1$ 
46:       $best_{rotation} \leftarrow \infty$ 
47:       $best_{edgeLength} \leftarrow \infty$ 
48:       $best_{symmetry} \leftarrow \infty$ 
49:       $best_{score} \leftarrow \infty$ 
50:       $best_{reflection} \leftarrow false$ 
51:      for each  $x \in \{min_x, min_x + gridSpacing_x, \dots, max_x\}$  do
52:          for each  $y \in \{min_y, min_y + gridSpacing_y, \dots, max_y\}$  do
53:              for each  $rotation \in \{min_{angle}, min_{angle} + angleIncrease, \dots, max_{angle}\}$  do
54:                  for each  $reflection \in \{true, false\}$  do
55:                      if  $|N_c| = 1$  then
56:                          if firstIteration AND reflection then
57:                              reflectPattern( $G_p$ )
58:                          else
59:                              Break reflection loop
60:                      else ▷ Reflect every time if there's more than 1 connecting edge
61:                           $currentReflection \leftarrow reflection$ 
62:                          if reflection then
63:                              reflectPattern( $G_p$ )
64:                          centrePattern( $G_p, x, y$ )

```

```

65: | | | | | rotatePattern( $G_p, rotation$ )
66: | | | | |  $current_{edgeLength} \leftarrow \text{CALCULATEEDGELENGTH}(E_c)$  ▷ See Below
67: | | | | |  $current_{score} \leftarrow \text{CALCULATESCORE}(drawnPatterns)$  ▷ See Algorithm 3.26
68: | | | | |  $current_{symmetry} \leftarrow \text{SYMMETRYMEASURE}(drawnPatterns)$  ▷ See Algorithm 3.25
69: | | | | |  $current_{distance} \leftarrow \text{closestDistance}(G_p, drawnPatterns)$ 
70: | | | | | if  $current_{distance} \geq min_{separation}$  then ▷ Do not draw too close to existing patterns
71: | | | | | | if  $current_{score} < best_{score}$  then
72: | | | | | | | SAVEBESTLOCATION ▷ See Below
73: | | | | | | else if  $current_{score} = best_{score}$  then
74: | | | | | | | if  $current_{edgeLength} < best_{edgeLength}$  then
75: | | | | | | | | SAVEBESTLOCATION ▷ See Below
76: | | | | | | | else if  $current_{edgeLength} = best_{edgeLength}$  then
77: | | | | | | | | if  $current_{symmetry} < best_{symmetry}$  then
78: | | | | | | | | | SAVEBESTLOCATION ▷ See Below
79: | drawPatternInIdealLayout( $G_p$ ) ▷ Draw pattern in best location
80: | centrePattern( $G_p, best_x, best_y$ )
81: | if  $best_{reflection}$  then
82: | | reflectPattern( $G_p$ )
83: | rotatePattern( $G_p, best_{rotation}$ )
84:
85: function SAVEBESTLOCATION
86: |  $best_x \leftarrow x$ 
87: |  $best_y \leftarrow y$ 
88: |  $best_{rotation} \leftarrow rotation$ 
89: |  $best_{reflection} \leftarrow reflection$ 
90: |  $best_{score} \leftarrow current_{score}$ 
91: |  $best_{symmetry} \leftarrow current_{symmetry}$ 
92: |  $best_{edgeLength} \leftarrow current_{edgeLength}$ 
93:
94: function CALCULATEEDGELENGTH( $E_c$ )
95: |  $x \leftarrow 0$ 
96: | for each Edge  $e \in E_c$  do
97: | |  $x \leftarrow x + (\text{length}(e)^2)$ 
98: | return  $x$ 

```

3.5.6 Connected by Multiple Edges

Some patterns may be connected to the currently drawn set by more than one edge. The drawing algorithm for this is identical to that for one edge (See Section 3.5.5), with one exception. The pattern is reflected each time, rather than

on the first iteration. This enables the possibility of uncrossing the connecting edges, (a condition which does not exist when only one edge is connected). Although it has no effect on the layout of a pattern connected by one edge, the metric for calculating edge length totals the squared length of each edge: this enables more balanced edge lengths to be given a lower score than highly mismatched edges. This method is described in Algorithm 3.17 and an example given in Figure 3.52.

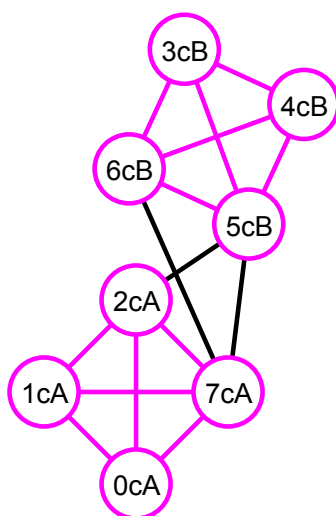


Figure 3.52: Example of patterns connected by multiple edges

3.5.7 No connections

Patterns may not be connected to the currently drawn set by any of the above methods. These are said to have “no connection”, although they are not disconnected graphs - these patterns have no direct connection to the drawn set. To find a sensible location for these patterns, any routes between the pattern and the drawn set are found using a breadth first search. Virtual edges are created between the start and finish of these routes, and the process of drawing is similar to the multiple edges approach. The ideal distance of these edges is

dependent on the number of nodes contained within the route. Routes are then drawn according to the method detailed in Section 3.5.8 and are highlighted in solid grey, as can be seen in Figure 3.53. The algorithm is defined below in Algorithm 3.18 and has a time complexity of $\mathcal{O}(V_p + E_p)$, due to the algorithm performing a breadth first search.

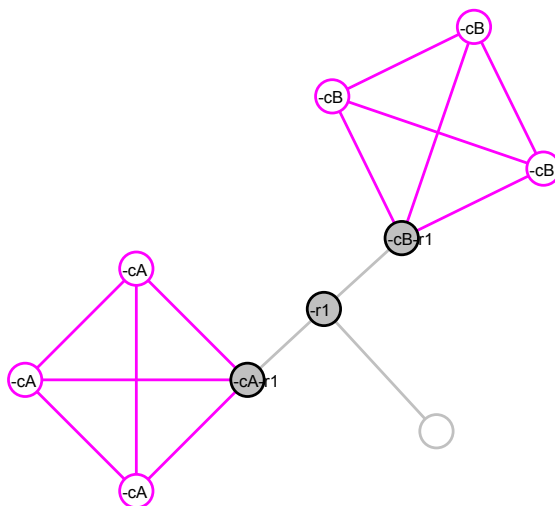


Figure 3.53: Example of two cliques having no connection

Algorithm 3.18: No connections

```

01: function DRAWPATTERN( $G_p$ )
02:    $G \leftarrow (V, E)$ 
03:    $G_p \leftarrow (V_p, E_p)$  ▷ The pattern being drawn
04:    $E_{pairs} \leftarrow \emptyset$ 
05:    $P_r \leftarrow \emptyset$  ▷ List of routes
06:   for each Node  $v \in V_p$  do
07:      $(V_r, E_r) \leftarrow$  new Route
08:      $V_r \leftarrow v$ 
09:      $Q_r \leftarrow []$  ▷ List of routes
10:      $Q_r \leftarrow Q_r \cup (V_r, E_r)$ 
11:     while  $Q_r \neq \emptyset$  do
12:        $(V_r, E_r) \leftarrow Q_r[0]$ 
13:        $Q_r \leftarrow Q_r \cup (V_r, E_r)$ 
14:        $n_{last} \leftarrow$  lastNode( $V_r$ )
15:       if inAnotherPattern( $v_{last}, G_p$ ) then

```

```

16: | | | |  $P_r \leftarrow P_r \cup (V_r, E_r)$ 
17: | | | | Skip this route
18: | | | | for each Node  $\{v_c \mid v_c \in \text{connectingNodes}(v_{last}), v_c \notin V_p, v_c \notin V_r\}$  do
19: | | | | |  $R_r \leftarrow (V_r \cup v_c, \emptyset)$ 
20: | | | | |  $Q_r \leftarrow Q_r \cup R_r$ 
21: | | | | for Route  $(V_r, E_r) \in P_r$  do
22: | | | | | for  $i \in \{0, 1, 2, \dots, |V_r| - 1\}$  do
23: | | | | | |  $e \leftarrow \text{edgesBetween}(V_r[i], V_r[i + 1])$ 
24: | | | | | |  $E_r \leftarrow E_r \cup e$ 
25: | | | | | |  $e_{pair} \leftarrow (V_r[i], V_r[i + 1])$ 
26: | | | | | |  $E_{pairs} \leftarrow E_{pairs} \cup e_{pairs}$ 
27: | | | | |  $sepDistance \leftarrow 0$ 
28: | | | | | for Route  $(V_r, E_r) \in P_r$  do
29: | | | | | |  $sepDistance \leftarrow sepDistance + (|V_r| \times 50)$ 
30: | | | | |  $\text{CONNECTEDBYEDGES}(G_p, sepDistance) \triangleright$  Draw using Connected By Edges (See Algorithm 3.17)
31: | | | |  $\text{drawn}(P_r) \leftarrow \text{true}$ 
32: | | | |  $\text{destroyEdges}(E_{pairs})$ 

```

3.5.8 Paths & Routes

Paths and routes (the series of nodes that joins two patterns which are classed as having no connection, as described in Section 3.5.7) have no ideal layout, due to their flexibility and varied nature. Instead, they must be drawn using a different method to other patterns. Paths and routes are initially drawn as a straight line, either emanating from a node, or connecting two nodes. Once all other patterns are drawn, this straight line is then investigated to find a more suitable layout. Several options of curve are tested, and an occlusion & edge crossing score is calculated, as can be seen in Figure 3.54. The layout with the best score is chosen as the final drawing location. The algorithm is defined below in Algorithm 3.19 and has a time complexity of $\mathcal{O}(V_p)$, due to the algorithm iterating through every node in the pattern in order to determine its new location (Line 33).

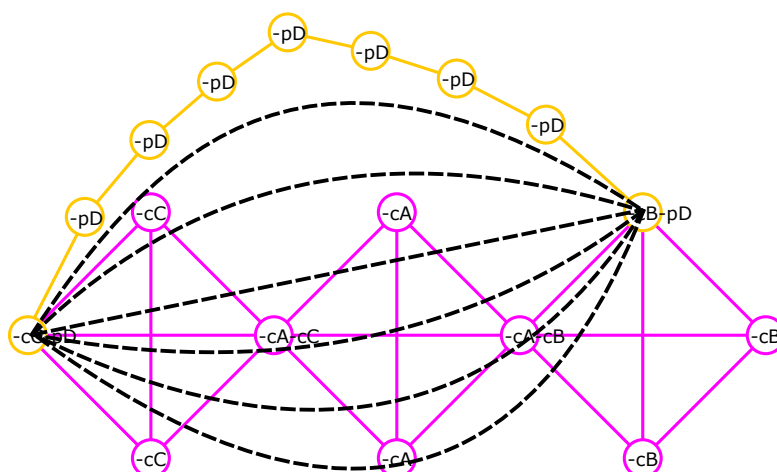


Figure 3.54: Example of various possible path locations

Algorithm 3.19: Path and Route Adjustment

```

01: function ADJUSTPATHS( $G_p$ )
02:    $G \leftarrow (V, E)$ 
03:    $G_p \leftarrow (V_p, E_p)$  ▷ The pattern being modified
04:    $bestScore \leftarrow \text{CALCULATESCORE}(drawnPatterns)$  ▷ See Algorithm 3.26
05:    $bestLayout \leftarrow 0$ 
06:   for  $j \in \{1, 2, \dots, 6\}$  do ▷ 7 different locations to test (Location 0 is the current one)
07:     TRYPATHLOCATIONS( $(V_p, E_p), j$ ) ▷ See Below
08:      $currentScore \leftarrow \text{CALCULATESCORE}(drawnPatterns)$  ▷ See Algorithm 3.26
09:     if  $currentScore < bestScore$  then
10:        $bestLayout \leftarrow j$ 
11:        $bestScore \leftarrow currentScore$ 
12:     TRYPATHLOCATIONS( $(V_p, E_p), bestLocation$ ) ▷ Draw in best location. See Below
13:
14: function TRYPATHLOCATIONS( $(V_p, E_p), j$ )
15:    $v_{first} \leftarrow V_p[0]$ 
16:    $v_{last} \leftarrow V_p[|V_p| - 1]$ 
17:    $V_m \leftarrow V_p$ 
18:   if  $j = 0$  then
19:     MOVEPATH( $(V_p, E_p), 0, V_m, v_{first}, v_{last}, 1$ ) ▷ See Below
20:   else if  $j = 1$  then
21:     MOVEPATH( $(V_p, E_p), 50, V_m, v_{first}, v_{last}, 1$ ) ▷ See Below
22:   else if  $j = 2$  then
23:     MOVEPATH( $(V_p, E_p), 50, V_m, v_{first}, v_{last}, -1$ ) ▷ See Below
24:   else if  $j = 3$  then
25:     MOVEPATH( $(V_p, E_p), 75, V_m, v_{first}, v_{last}, 1$ ) ▷ See Below
26:   else if  $j = 4$  then

```

```

27: | | MOVEPATH((Vp, Ep), 75, Vm, vfirst, vlast, -1) ▷ See Below
28: | else if j = 5 then
29: | | MOVEPATH((Vp, Ep), 100, Vm, vfirst, vlast, 1) ▷ See Below
30: | else if j = 6 then
31: | | MOVEPATH((Vp, Ep), 100, Vm, vfirst, vlast, -1) ▷ See Below
32:
33: function MOVEPATH((Vp, Ep), amount, Vm, vfirst, vlast, flip)
34: | gap ← 2
35: | patternAngle ← angle(vfirst, vlast)
36: | rotationAngle ← (patternAngle + 90°) mod 360°
37: | if isPath(Vp, Ep) then
38: | | if patterns(Vm[1]) > 1 then
39: | | | Vm ← Vm \ vfirst ▷ Ignore the first node if this is a path sharing an edge
40: | | if patterns(Vm[|Vm| - 2]) > 1 then
41: | | | Vm ← Vm \ vlast ▷ Ignore the last node if this is a path sharing an edge
42: | | for each v ∈ Vm, {i | i ≥ 1, i < |Vm| - 1} do
43: | | | fromFirst ← i
44: | | | fromLast ← |Vm| - 1 - i
45: | | | d ← min(fromFirst, fromLast)
46: | | | distance ← gap · amount · log(d + 1) · flip
47: | | | height ← -distance · sin(rotationAngle)
48: | | | width ← distance · cos(rotationAngle)
49: | | | vx ← vx + width
50: | | | vy ← vy + height

```

3.5.9 Loose nodes

It is also necessary to draw any nodes that do not belong to patterns. These nodes are assigned to two groups: *singles* and *multiples*. *Singles* are those nodes which connect to either none or one node that belongs in a pattern and may connect to any number of other loose nodes. *Multiples* are nodes which connect to 2 or more patterns and these may connect to any number of other loose nodes. Although these restrictions may suggest a large number of nodes will be classed as loose, in practice most nodes are actually contained within patterns, and thus not loose.

3.5.9.1 Identifying and Drawing Loose Nodes

Once all patterns have been drawn, loose nodes may be identified and drawn. If no patterns have been drawn (because none were identified), then the most connected node is considered to have been drawn as a “starter” node. To identify any potential loose nodes, every drawn node is considered. For each of these, every connected node that has not been drawn is considered to be a loose node; if the loose node is connected to one drawn node then it is a single (and the node connecting to this is stored), if not, it is a multiple.

Once a list of loose nodes is identified, all multiples are drawn first using a simple barycentre approach [120], where the centre of all the connecting nodes is found and the loose node is drawn there. However, this location may be occluded, so a small grid is checked to find a better location if needed. Once complete, all single nodes are drawn. The list of nodes containing singles is iterated through, with the best drawable area found (See Section 3.6.1) and the single loose nodes are arranged within that area.

Once this list is complete, the identification and drawing process starts again until all nodes within the graph are drawn. This iterative process has the effect of building tree-like structures if there are a number of single loose nodes connected together. The identification process is shown in Algorithm 3.20 and has a time complexity of $\mathcal{O}(V \cdot (\log V)^2)$. This is due to the identification process ($(\log V)^2$, Line 16), and the drawing of multiples (Algorithm 3.21, $\log V$) and singles (Algorithm 3.22, $V \cdot \log V$), both of which are run $\log V$ times.

In a previous version of the method, a list of single loose nodes was identified, and sorted after each node was drawn. This list of singles was then sorted in order of greater number of patterns connecting and greater number of connections. Once sorted, the largest drawable area around the centre was found

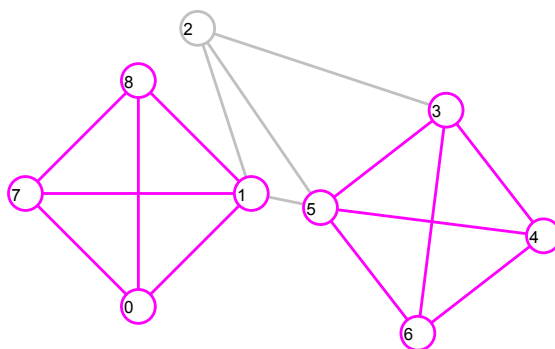


Figure 3.55: Example of Loose Nodes - Multiples

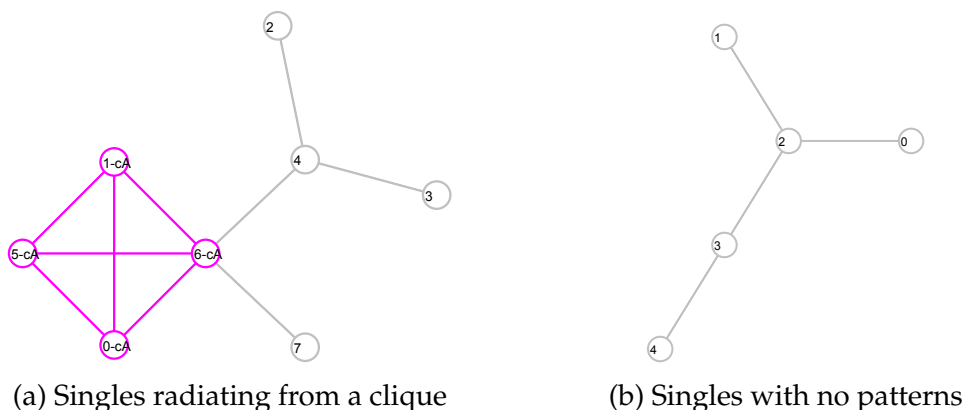


Figure 3.56: Examples of Loose Nodes - Singles

and the single loose nodes are arranged within that. This method was unfortunately both poor at identifying all loose nodes and had an unnecessarily complicated implementation. An improved system of identifying single nodes was therefore completed instead.

Algorithm 3.20: Loose Nodes Identification

```

01: function TIDYLOOSENODES( $G_p$ )
02:    $G \leftarrow (V, E)$ 
03:   visited( $G$ )  $\leftarrow$  false
04:    $V_s \leftarrow \emptyset$ 
05:    $V_m \leftarrow \emptyset$ 
06:   FINDLOOSENODES( $(V, E), V_s, V_m$ )
07:   while  $V_s \neq \emptyset$  OR  $V_m \neq \emptyset$  do

```

▷ Nodes with singles

▷ Multiple Loose Nodes

▷ See Below

```

08:   |   | for each Node  $v \in V_m$  do
09:   |   |   | DRAWMULTIPLE( $v$ )                                ▷ See Algorithm 3.21
10:   |   | for each Node  $v \in V_s$  do
11:   |   |   | DRAWSINGLES( $v$ )                                ▷ See Algorithm 3.22
12:   |   |  $V_s \leftarrow \emptyset$ 
13:   |   |  $V_m \leftarrow \emptyset$ 
14:   |   | FINDLOOSENODES( $(V, E), V_s, V_m$ )                    ▷ See Below
15:
16: function FINDLOOSENODES( $(V, E), V_s, V_m$ )
17:   | for each Node  $\{v \mid v \in V, \text{drawn}(v)\}$  do
18:   |   | for each Node  $\{v_c \mid v \in V, \neg \text{drawn}(v_c), v_c \in \text{connectingNodes}(v)\}$  do
19:   |   |   | if connectingDrawnNodes( $v_c$ ) = 1 AND  $v \notin V_s$  then                                ▷ Single found
20:   |   |   |   |  $V_s \leftarrow V_s \cup v$ 
21:   |   |   |   | singles( $v$ )  $\leftarrow$  singles( $v$ )  $\cup v_c$ 
22:   |   |   | else if connectingDrawnNodes( $v_c$ ) > 1 AND  $v \notin V_m$  then                                ▷ Multiple found
23:   |   |   |   |  $V_m \leftarrow V_m \cup v$ 

```

Algorithm 3.21: Drawing Loose Nodes (Multiples)

```

01: function DRAWMULTIPLE( $v$ )
02:   |  $G \leftarrow (V, E)$ 
03:   | visited( $v$ )  $\leftarrow$  true
04:   | drawn( $v$ )  $\leftarrow$  true
05:   | drawn(connectingEdges( $v$ ))  $\leftarrow$  true
06:   |  $total_x \leftarrow 0$ 
07:   |  $total_y \leftarrow 0$ 
08:   |  $total_{nodes} \leftarrow 0$ 
09:   | for each Node  $\{v_c \mid v_c \in V, v_c \in \text{connectingNodes}(v), \text{patterns}(v_c) \neq \emptyset\}$  do
10:   |   |  $total_x \leftarrow total_x + v_{c_x}$ 
11:   |   |  $total_y \leftarrow total_y + v_{c_y}$ 
12:   |   |  $total_{nodes} \leftarrow total_{nodes} + 1$ 
13:   |   |  $new_x \leftarrow total_x \div total_{nodes}$ 
14:   |   |  $new_y \leftarrow total_y \div total_{nodes}$ 
15:   |   | location( $v$ )  $\leftarrow$  ( $new_x, new_y$ )
16:   |   |  $currentLocation \leftarrow (new_x, new_y)$ 
17:   |   |  $minValue \leftarrow -100$                                 ▷ Define the search area
18:   |   |  $maxValue \leftarrow 100$ 
19:   |   |  $step \leftarrow 10$ 
20:   |   |  $bestScore \leftarrow$  CALCULATESCORE( $drawnPatterns$ )                                ▷ See Algorithm 3.26
21:   |   |  $best\Delta x \leftarrow 0$ 
22:   |   |  $best\Delta y \leftarrow 0$ 
23:   |   | for  $\Delta x \in \{minValue, minValue + step, \dots, maxValue\}$  do

```

```

24:   for  $\Delta y \in \{minValue, minValue + step, \dots, maxValue\}$  do
25:        $v_x \leftarrow currentLocation_x + \Delta x$ 
26:        $v_y \leftarrow currentLocation_y + \Delta y$ 
27:        $currentScore \leftarrow CALCULATESCORE(drawnPatterns)$  ▷ See Algorithm 3.26
28:       if  $currentScore < bestScore$  then
29:            $bestScore \leftarrow currentScore$ 
30:            $best\Delta x \leftarrow \Delta x$ 
31:            $best\Delta y \leftarrow \Delta y$ 
32:    $location(v) \leftarrow (currentLocation + best\Delta x, currentLocation + best\Delta y)$ 

```

Algorithm 3.22: Drawing Loose Nodes (Singles)

```

01: function DRAWSINGLES( $v_c$ )
02:    $G \leftarrow (V, E)$ 
03:    $V_s \leftarrow singles(v_c)$  ▷ The single loose nodes attached to this node
04:    $idealDistance \leftarrow 120$ 
05:    $drawableAreas \leftarrow \emptyset$  ▷ Empty list of Drawing Areas
06:   for each Node  $\{v \mid v \in V, v \notin V_s, v \neq v_c, drawn(v), distance(v, v_c) \leq (idealDistance \times 1.2)\}$  do
07:        $padding \leftarrow 5^\circ$ 
08:        $angle \leftarrow angle(v, v_c)$ 
09:        $drawableAreas \leftarrow drawableAreas \cup (angle - padding, angle + padding)$ 
10:    $drawableAreas \leftarrow JOININVALIDAREAS(drawableAreas)$  ▷ See Algorithm 3.23
11:    $bestArea \leftarrow FINDBESTAREA(drawableAreas)$  ▷ See Algorithm 3.24
12:    $startAngle \leftarrow startAngle(bestArea)$ 
13:    $endAngle \leftarrow endAngle(bestArea)$ 
14:    $angleStep \leftarrow (endAngle - startAngle) \div (|V_s| + 1)$ 
15:   for each Node  $v \in V_s$  do
16:        $angle \leftarrow (angleStep \cdot (indexOf(V_s, v) + 1)) + startAngle$ 
17:        $\Delta_x \leftarrow (idealDistance \cdot \cos(-thisAngle))$ 
18:        $\Delta_y \leftarrow (idealDistance \cdot \sin(-thisAngle))$ 
19:        $v_x \leftarrow v_{c_x} + \Delta_x$ 
20:        $v_y \leftarrow v_{c_y} + \Delta_y$ 
21:    $visited(V_s) \leftarrow true$ 
22:    $drawn(V_s) \leftarrow true$ 
23:    $drawn(connectingEdges(V_s)) \leftarrow true$ 

```

3.6 Methods Used in Numerous Drawing Techniques

Throughout the implementation, it has been necessary to create a number of methods with specific jobs that may be used by a number of the drawing techniques.

3.6.1 Drawable Areas

Certain patterns require that they be drawn in areas that are currently uninhabited by other nodes. For example, when drawing a star that shares one edge, the outer nodes may only be drawn in areas where they would not cause occlusion or unnecessary edge crossings. To avoid this, the concept of drawable areas is introduced.

Any nodes within the entire graph that exist within range of the centre node (and that are not part of this pattern) have an invalid area created around them (with padding of 5° either side to avoid occlusion). Once all nodes have been considered, invalid areas are then processed. Areas may need processing because they meet any of the following conditions:

1. The area overlaps the origin (See Figure 3.57a)
2. The two areas are in the same location and have the same size (See Figure 3.57b)
3. The two areas are adjacent (where the join is not the origin) (See Figure 3.57c)
4. One area is contained within another (See Figure 3.57d)
5. One area partially overlaps another (See Figure 3.57e)

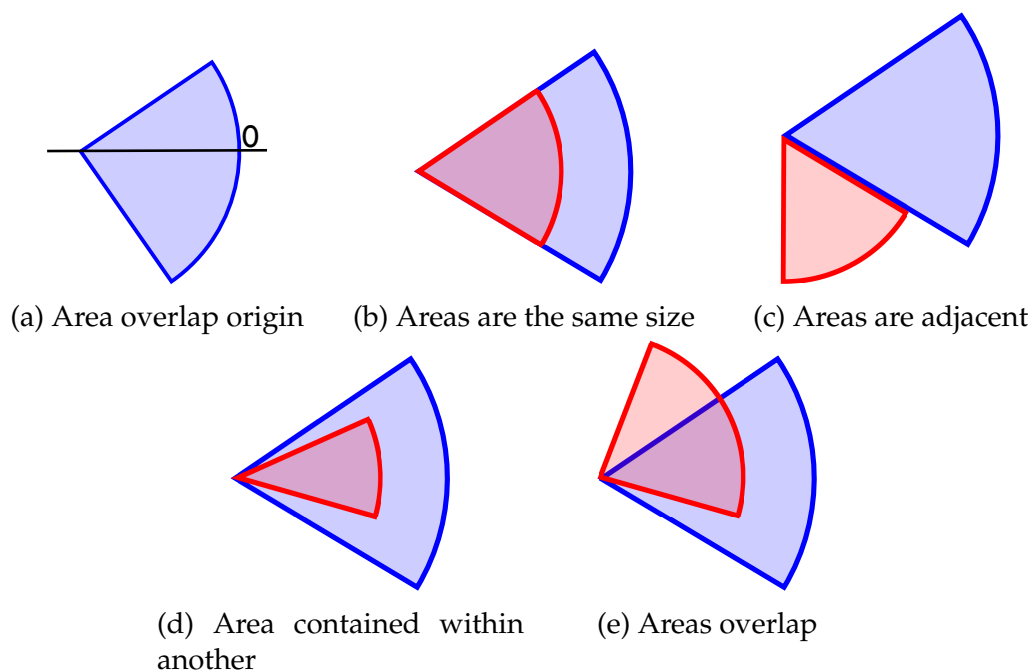


Figure 3.57: Examples of Area Intersections

In a previous version of the method, areas which had nodes connecting to nodes in other areas were joined, so that edge crossings were considered when picking the best area. However, the best area was also only ever then chosen based on size. Since the best area is now calculated using the number of edge crossings and occlusions, it is not necessary to join areas connected by edges.

This process ensures that a list of non-contiguous areas is identified. Once a list of invalid areas has been created, the system then finds the best valid area, according to the formula below:

$$(10 * occlusionAndEdgeScore) - sizeOfRegion$$

To identify the best area, the occlusion and edge crossing score is calculated (See Section 3.6.3). This is then multiplied by 10 and subtracted from the size of the region. This allows the system to pick an area which is not necessarily

the biggest, but will allow a better result to be drawn. The area is then returned to the drawing technique for use. The concept is shown in Figure 3.58, where invalid areas are shown in yellow and the best valid area is shown in green.

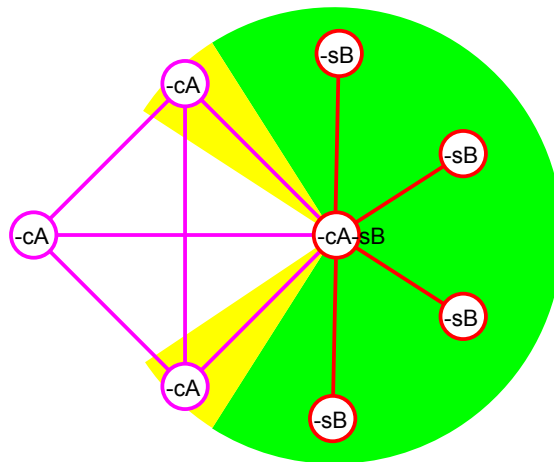


Figure 3.58: The invalid areas and the best valid area when drawing pattern sB

The process of finding the best valid area is shown in Algorithms 3.23 and 3.24. Algorithm 3.23 has a time complexity of $\mathcal{O}(A^2)$, where A is the set of areas, while Algorithm 3.24 has a time complexity of $\mathcal{O}(A \cdot V)$

Algorithm 3.23: Join Drawing Areas

```

01: function JOINDRAWINGAREAS( $A$ )
02:    $G \leftarrow (V, E)$ 
03:    $changed \leftarrow false$ 
04:   while  $changed$  do
05:      $changed \leftarrow false$ 
06:      $A_a \leftarrow \emptyset$  ▷ Areas to add
07:      $A_r \leftarrow \emptyset$  ▷ Areas to remove
08:      $changed \leftarrow CHECKDRAWINGAREAS(A, A_a, A_r)$  ▷ See Line 12
09:      $A \leftarrow (A \setminus A_r) \cup A_a$ 
10:   return  $A$ 
11:
12: function CHECKDRAWINGAREAS( $A, A_a, A_r, minValue = 0, maxValue = 360$ )
13:   for each Area ( $a_{start}, a_{finish}$ )  $\in A$  do
14:      $a \leftarrow (a_{start}, a_{finish})$ 
15:     if  $a_{finish} > maxValue$  then ▷ See Condition 1 (Figure 3.57a)
16:        $A_a \leftarrow A_a \cup \{(a_{start}, maxValue), (minValue, a_{finish} \bmod maxValue)\}$ 
17:        $A_r \leftarrow A_r \cup a$ 
18:       return true
19:     else if  $a_{start} < minValue$  then ▷ See Condition 1 (Figure 3.57a)
20:        $A_a \leftarrow A_a \cup \{(a_{start} + maxValue, maxValue), (minValue, a_{finish} \bmod maxValue)\}$ 
21:        $A_r \leftarrow A_r \cup a$ 
22:       return true
23:     else if  $a_{start} > a_{finish}$  then ▷ See Condition 1 (Figure 3.57a)
24:        $A_a \leftarrow A_a \cup \{(a_{start}, maxValue), (minValue, a_{finish})\}$ 
25:        $A_r \leftarrow A_r \cup a$ 
26:       return true
27:   for each Area ( $a_{start}, a_{finish}$ )  $\in A$  do
28:     for each Area ( $b_{start}, b_{finish}$ )  $\in A$  do
29:        $a \leftarrow (a_{start}, a_{finish})$ 
30:        $b \leftarrow (b_{start}, b_{finish})$ 
31:       if  $a = b$  then
32:         Skip this area  $b$ 
33:       if  $a_{start} = b_{start}$  AND  $a_{finish} = b_{finish}$  then ▷ See Condition 2 (Figure 3.57b)
34:          $A_a \leftarrow A_a \cup (a_{start}, a_{finish})$ 
35:          $A_r \leftarrow A_r \cup \{a, b\}$ 
36:         return true
37:       if  $a_{finish} = b_{start}$  AND ( $a_{finish} \neq minValue$  OR  $a_{finish} \neq maxValue$ ) then ▷ See
38:         Condition 3 (Figure 3.57c)
39:          $A_a \leftarrow A_a \cup (a_{start}, b_{finish})$ 
40:          $A_r \leftarrow A_r \cup \{a, b\}$ 
41:         return true
42:       if  $b_{finish} = a_{start}$  AND ( $b_{finish} \neq minValue$  OR  $b_{finish} \neq maxValue$ ) then ▷ See

```

```

Condition 3 (Figure 3.57c)
42:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
43:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
44:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
45:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
46:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
47:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
48:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
49:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
50:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
51:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
52:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
53:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
(Figure 3.57e)
54:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
55:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
56:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
57:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
(Figure 3.57e)
58:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
59:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
60:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
61:   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
return false

```

Algorithm 3.24: Find Best Drawing Areas

```

01: function FINDBESTAREA( $A_i$ ,  $minValue = 0$ ,  $maxValue = 360$ )
02:    $G \leftarrow (V, E)$ 
03:   if  $A_i = \emptyset$  then
04:     return ( $minValue$ ,  $maxValue$ )
05:    $A_v \leftarrow$  FINDVALIDAREAS( $A_i$ ,  $minValue$ ,  $maxValue$ )           ▷ Valid Drawing Areas. See Below
06:    $a_{best} \leftarrow null$                                            ▷ Best Drawing Area
07:    $bestScore \leftarrow -\infty$ 
08:   for each Area ( $a_{start}, a_{finish}$ )  $\in A_v$  do
09:      $a \leftarrow (a_{start}, a_{finish})$ 
10:     if  $a_{best} = null$  then
11:        $a_{best} \leftarrow a$ 
12:        $bestScore \leftarrow -(10 \times \text{CALCULATESCORE}(\text{drawnPatterns})) + (a_{finish} - a_{start})$    ▷ See
Algorithm 3.26
13:       Skip this area
14:        $currentScore \leftarrow -(10 \times \text{CALCULATESCORE}(\text{drawnPatterns})) + (a_{finish} - a_{start})$    ▷ See
Algorithm 3.26

```

```

15: | | if currentScore > bestScore then
16: | | | abest ← a
17: | | | bestScore ← currentScore
18: | return abest
19:
20: function FINDVALIDAREAS(Ai, minValue, maxValue)
21: | Av ← ∅ ▷ Valid Areas
22: | firstStart ← maxValue
23: | for each Area  $\{(a_{start}, a_{finish}) \mid (a_{start}, a_{finish}) \in A_i, a_{finish} \neq maxValue\}$  do
24: | | a ← (astart, afinish)
25: | | closestStart ← maxValue
26: | | if astart < firstStart then
27: | | | firstStart ← astart
28: | | for each Area  $\{(b_{start}, b_{finish}) \mid (b_{start}, b_{finish}) \in A_i, a \neq (b_{start}, b_{finish})\}$  do
29: | | | b ← (bstart, bfinish)
30: | | | if bstart < closestStart AND bstart > afinish then
31: | | | | closestStart ← bstart
32: | | if closestStart = maxValue then
33: | | | Av ← Av ∪ (afinish, maxValue)
34: | | else
35: | | | Av ← Av ∪ (afinish, closestStart)
36: | if firstStart ≠ minValue then
37: | | (astart, afinish) ←  $\{(a_{start}, a_{finish}) \mid (a_{start}, a_{finish}) \in A_v, a_{finish} = maxValue\}$ 
38: | | if (astart, afinish) = ∅ then
39: | | | Av ← Av ∪ (minvalue − (maxValue − astart), firstStart)
40: | | | Av ← Av \ (astart, afinish)
41: | | else
42: | | | Av ← Av ∪ (minvalue, firstStart)
43: | return Av

```

3.6.2 Symmetry Score

A number of drawing algorithms require the calculation of a score to represent the symmetry of the newly drawn pattern. Calculating such a score is not immediately obvious and several versions were created.

The first method was to compare the distance between the centre of the currently drawn set and the newly drawn pattern. This method did not return an

adequate representation, and resulted in solutions similar to Figure 3.59.

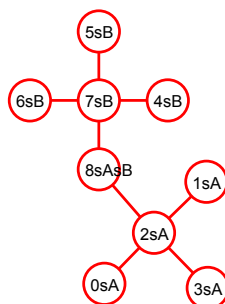


Figure 3.59: Poor symmetry (1)

An improvement was made to this where the distance between every node connected to the shared node or edge in both the currently drawn set and the newly drawn pattern was calculated. In order to avoid occlusion, if any node was less than a fixed distance from another, then the maximum score would be returned. This, however, also provided an inadequate representation, with results similar to Figure 3.60.

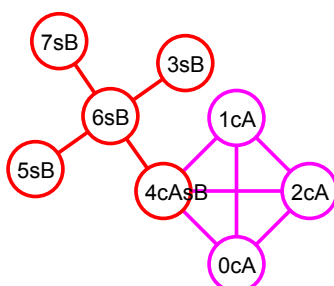


Figure 3.60: Poor symmetry (2)

A third method was then implemented, which focussed on angles rather than distance and this is the method currently used (see Algorithm 3.25). The angle between the horizontal and the imaginary line connecting the centre of the newly drawn pattern and the shared edge or node and the angle between the centre of each shared pattern and the shared edge or node was calculated

(i.e. in Figure 3.61, the angle between the centre of pattern A and node 6cAcB is compared to the angle between the centre of pattern B and node 6cAcB).

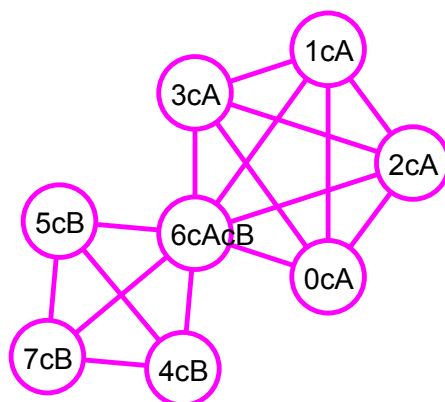


Figure 3.61: Good symmetry

The most symmetrical score would be that where the difference between the angle of the new pattern that of the connecting patterns is zero. This produces symmetrical layouts and is demonstrated in Figure 3.62. Pattern tA is drawn first, in a horizontal position. Pattern tB is drawn second, and the angle between the imaginary line connecting the centre of this pattern and the shared node (shown as a dashed blue line) and the horizontal is identical (with some rounding errors) to the angle between the imaginary line connecting the centre of pattern tA and the shared node (in this case, the same as the horizontal). Therefore, this is the best symmetrical location for pattern tB . The two varieties of symmetry score are described in Algorithm 3.25 and share the time complexity of $\mathcal{O}(P)$, where P is the set of currently drawn patterns.

Algorithm 3.25: Symmetry Score

```

01: function SYMMETRYSCORE( $G_p, v$ ) ▷ Shared Node
02:    $result \leftarrow 0$ 
03:    $patternAngle \leftarrow \mathbf{angle}(v, \mathbf{centre}(G_p))$ 
04:   for each Pattern  $G_{pa} \in \mathbf{patterns}(v)$  do

```

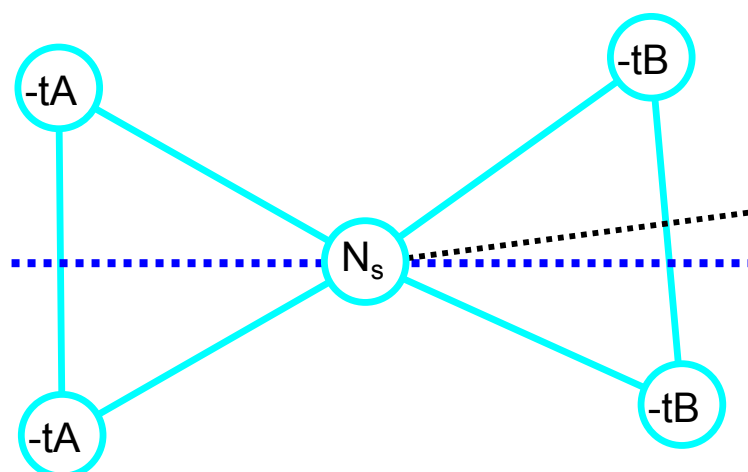


Figure 3.62: Symmetry Example

```

05: |   |   angle ← angle(centre( $G_{pa}$ ),  $v$ )           ▷ Note reversed order
06: |   |   result ← result + |patternAngle - angle|
07: |   |   return result
08:
09: function SYMMETRYSCORE( $G_p, e$ )           ▷ Connecting Edge
10: |   |    $v_p$  ← { $v_p$  |  $v_p \in G_p, v_p \in e$ }   ▷ Pattern Node
11: |   |    $v_o$  ← { $v_o$  |  $v_o \in e, v_o \neq v_p$ }   ▷ Other Node
12: |   |   centre ← centre( $e$ )
13: |   |   result ← 0
14: |   |   patternAngle ← angle(centre, centre( $G_p$ ))
15: |   |   for each Pattern  $G_{pa} \in$  patterns( $v$ ) do
16: |   |   |   angle ← angle(centre( $G_{pa}$ ), centre)   ▷ Note reversed order
17: |   |   |   result ← result + |patternAngle - angle|
18: |   |   return result

```

3.6.3 Occlusion & Edge Crossing Score

There are a number of circumstances where certain metrics need to be calculated. The metrics used in this system are edge crossings (E_c), node-node occlusion (O_{nn}) and node-edge occlusion (O_{ne}). All three are only calculated with respect to nodes or edges that have been drawn. To create a score for the current

layout, the following formula is used:

$$2(O_{nn} + O_{ne}) + E_c$$

The node-node and node-edge occlusion values are both multiplied by two as it is more important to reduce occlusion than edge crossings. The occlusion and edge crossing score calculation are described in Algorithm 3.26 and has the time complexity of $\mathcal{O}(V + E \log E)$, where both occlusion scores have the time complexity of $\mathcal{O}(V)$ (Lines 2 and 3) and the edge crossing calculation has the complexity of $\mathcal{O}(E \log E)$ (Line 4).

Algorithm 3.26: Occlusion and Edge Crossing Score

```

01: function OCCLUSIONSCORE(drawnPatterns)
02:   nnOccScore ← nodeNodeOcclusion(drawnPatterns)
03:   neOccScore ← nodeEdgeOcclusion(drawnPatterns)
04:   edgeCrossings ← edgeCrossings(drawnPatterns)
05:   score ← (2 × (nnOccScore + neOccScore)) + edgeCrossings
06:   return score

```

Chapter 4

Analysis of Generated Layouts

4.1 Introduction

This chapter contains various examples of real world data. Firstly, the collection of the data is discussed (see Section 4.2), before a side-by-side comparison of examples drawn with this pattern based layout and a force-directed layout (see Section 4.3). Following this, a number of metrics are compared for the earlier examples and discussed (see Section 4.4).

4.1.1 Force Directed Method

Force directed layout methods are amongst the most popular drawing techniques for graphs. It is a very common method as it produces good results, is relatively easy to implement and implementations exist in variety of platforms and languages. There are many variations of force directed methods and a number are described in Section 2.1.2. However, the version used for the examples in this chapter and the empirical study (see Chapter 5) is a modified version of Fruchterman and Reingold's [48] version of a spring embedder. The main

variation is the difference in force calculations, with the attractive force being defined as kd and the repulsive forces being defined as $\frac{r}{d^2}$, with d representing the distance between two nodes, $k = 0.05$ and $r = 10,000$. The layout method is complete once every force calculation is below 0.1. Fruchterman and Reingold's implementation is a highly common version of the force directed method and is also simple to implement. The chosen constant values were also used initially within the underlying graph creation software and produced good layouts. It was therefore considered not necessary to change these.

4.2 Data Collection

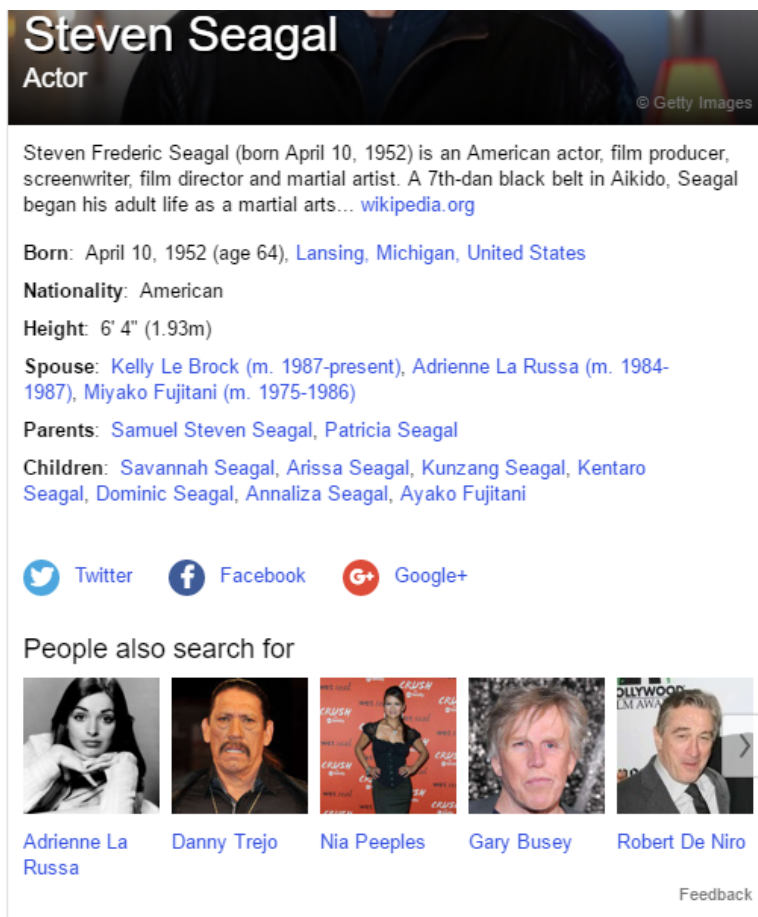
Described here are the real world data sources that are needed to provide real world examples for evaluating the system. All of these data sources were used to generate the examples used in the formal empirical study, which is described in Chapter 5.

4.2.1 Academy Award Nominees

One source of data was a social network based on Yahoo's "People also search for" section [4] for results of celebrities. This is an infobox on the side of the search results that suggests a number of other items based on the results of other users' searches¹. For each year from 1996 (the 69th) to 2014 (the 87th) the Best Actor and Best Actress nominees in the Academy Awards were compiled. Each of these were searched on Yahoo and an connection was created between them and any person Yahoo suggested. For example, in Figure 4.1, Steven Seagal

¹At the time, both Google and Bing altered the number of results by screen size, so scraping their sites only result in a maximum of 3 suggestions returned, as opposed to Yahoo's then maximum of 5. A maximum of 3 is likely to generate sparse graphs with very few patterns, so Yahoo was chosen as opposed to the more popular search engines

would be connected to Adrienne La Russa, Danny Trejo and so on. This process was repeated for the Best Actor, Best Actress, Best Supporting Actor and Best Supporting Actress to generate 38 examples.



Steven Seagal
Actor

Steven Frederic Seagal (born April 10, 1952) is an American actor, film producer, screenwriter, film director and martial artist. A 7th-dan black belt in Aikido, Seagal began his adult life as a martial arts... wikipedia.org

Born: April 10, 1952 (age 64), [Lansing, Michigan, United States](#)

Nationality: American

Height: 6' 4" (1.93m)

Spouse: [Kelly Le Brock](#) (m. 1987-present), [Adrienne La Russa](#) (m. 1984-1987), [Miyako Fujitani](#) (m. 1975-1986)

Parents: [Samuel Steven Seagal](#), [Patricia Seagal](#)

Children: [Savannah Seagal](#), [Arissa Seagal](#), [Kunzang Seagal](#), [Kentaro Seagal](#), [Dominic Seagal](#), [Annaliza Seagal](#), [Ayako Fujitani](#)

[Twitter](#) [Facebook](#) [Google+](#)

People also search for

[Adrienne La Russa](#) [Danny Trejo](#) [Nia Peeples](#) [Gary Busey](#) [Robert De Niro](#)

Feedback

Figure 4.1: Yahoo’s suggested “People also search for” for Steven Seagal [4]

4.2.2 Formula One Teammates

Another source of data was connecting Formula One drivers to their teammates. For each 3-year group between 1995 to 2015 (generating 19 examples), drivers were connected if they competed as team-mates in a Formula One race.

This does not result in a series of paired connections, as drivers often move between teams at the end of (or sometimes during) a season, and each example is taken over a three year period.

More formally, drivers were connected if they entered a FIA Formula One World Championship event for the same entrant. Drivers are also connected if they did not participate in the race, but took part in any qualifying (but not practice) sessions. The reason for not connecting drivers who were simply at the same team in a season is the example shown in Table 4.1. Here, the rounds that each Lotus-Renault driver participated in from the 2012 and 2013 seasons are shown [89, 90]. Connections would exist between Kimi Räikkönen and Romain Grosjean (based on 19 rounds in 2012 and 17 in 2013), but not between Kimi Räikkönen and Heikki Kovalainen as they never competed as team-mates (and indeed were in rival teams in 2012).

Table 4.1: Rounds competed by Lotus-Renault drivers in the 2012 and 2013 Formula One seasons [89, 90]

	2012 Season	2013 Season
Kimi Räikkönen	1–20	1–17
Heikki Kovalainen	-	18–19
Romain Grosjean	1–12, 14–20	1–19
Jérôme d’Ambrosio	13	-

4.2.3 Character Relations in Novels

One further source of data is identifying character relationships in novels. Classic novels were obtained from the Project Gutenberg website [2], had their copyright information (and any contents pages) removed and grouped into 5 line sections (in the form of 1–5, 2–6, 3–7, etc). Separately, a list of characters is

compiled manually, including all possible names of each character. Each 5 line section is then investigated and if two or more characters appear within that section, the characters are said to be connected. There are a number of challenges to the extraction of this data. Firstly, it is very difficult to draw context from the use of names within a passage of text. For example, in *A Christmas Carol*, Bob Cratchit is referred to using “Bob”, “Mr. Cratchit” and just “Cratchit”. Unfortunately, simply searching for “Crachit” would also create connections between his wife and children (when they are referred to by their full name). *Treasure Island* takes this even further with four characters called “John” and three called “Tom” all mentioned at various points in the novel. In these cases, any ambiguous names are not used and ignored. Secondly, partly because of the first point, obtaining the character lists can take some time, hence why only 15 examples were obtained.

4.3 Examples of Drawing

Below are a number of examples of both the pattern based system and a force directed layout. Data from the sources described in Section 4.2 and others are used for a visual comparison. For both drawing methods, the largest connected subgraph is chosen. For the spring embedder examples, the layout starts with a random starting position. Labels have been removed to improve the clarity of the drawings (unless needed for reference), as no work has gone into sensible label placing. Colours have also been removed, so as not to draw unfair attention to the patterns, while some images have been rotated to allow better positioning on the page.

4.3.1 Academy Award Nominees

The examples from Academy Award nominees (see Section 4.2.1) are often drawn in structures similar to trees. As neither drawing method is optimised for tree layouts, nor considers the overall area of the drawing, it is interesting to note the similarities between results from the two layouts.

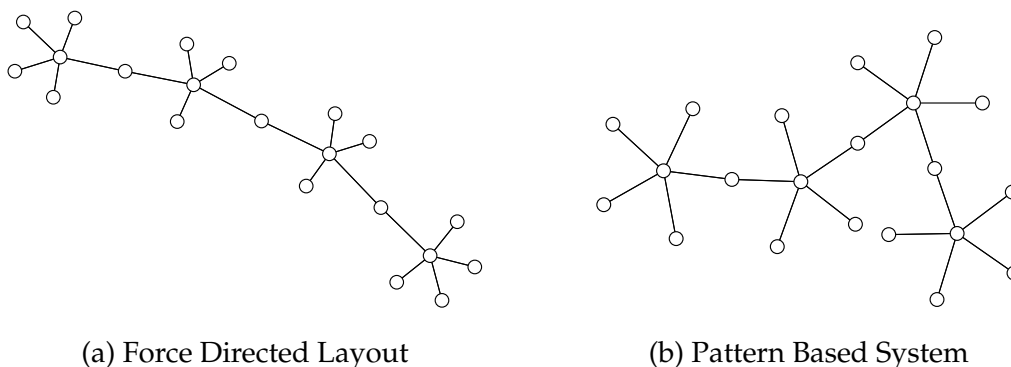


Figure 4.2: 1998 Best Leading and Supporting Actor and Actress Academy Award Nominees

In the example of the 1998 Best Leading and Supporting Actor and Actress nominees (Figure 4.2), the two layout methods have similar results. The force directed layout produces a longer, thinner layout while the pattern based method bends back upon itself, reducing the overall size of the layout and creating a more balanced aspect ratio.

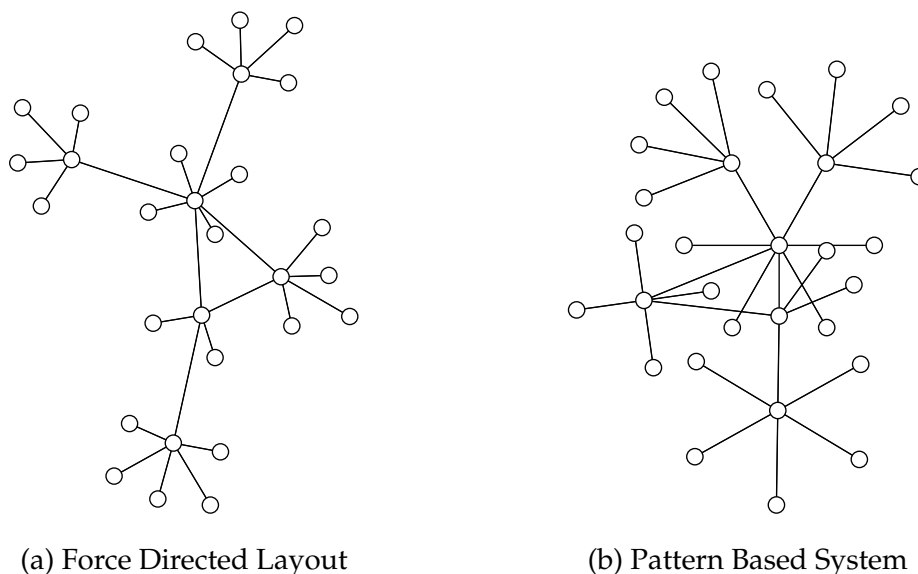


Figure 4.3: 1997 Best Leading and Supporting Actor and Actress Academy Award Nominees

The layout of the 1998 Best Leading and Supporting Actor and Actress nominees (Figure 4.3) contains a triangle. Both layouts draw a node inside this triangle, however the force directed layout draws the triangle large enough that nearby spokes do not create edge crossings or occlusion. The pattern based method again draws a more compact layout, which does cause the unnecessary edge crossings.

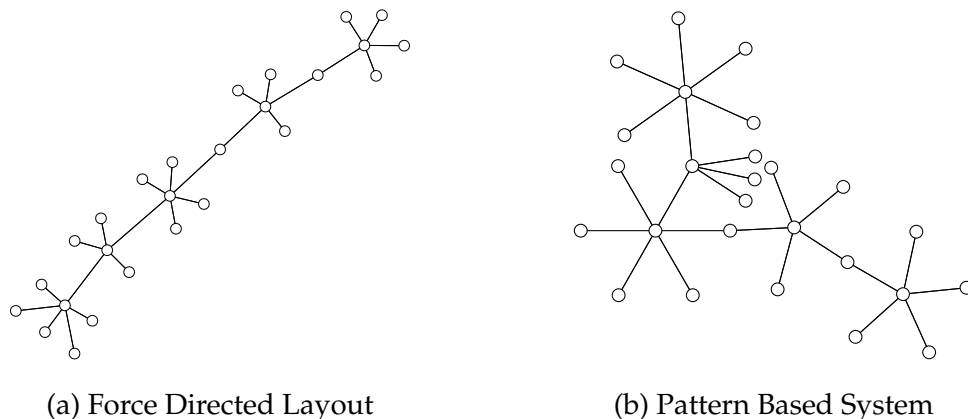


Figure 4.4: 2007 Best Leading Actor and Actress Academy Award Nominees

The 2007 Best Leading Actor and Actress nominees (Figure 4.4) example results in a similar layout from both methods. The pattern based system has created a tighter layout, and this has required three spokes of a star to be drawn close together. However, the force directed layout has resulted in a very long and thin example, which is a poor aspect ratio.

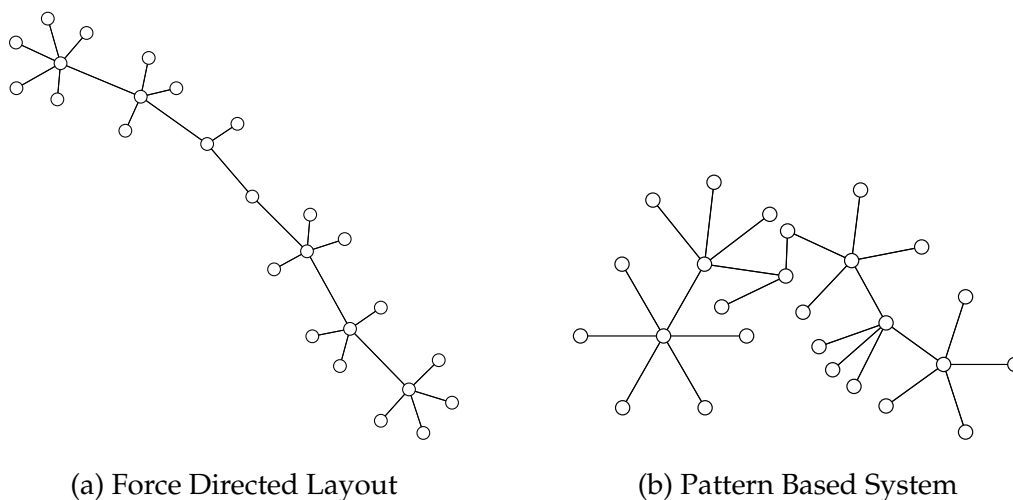


Figure 4.5: 2006 Best Leading and Supporting Actor and Actress Academy Award Nominees

In the example of the 2006 Best Leading and Supporting Actor and Actress nominees (Figure 4.5), again the two layout methods have drawn the graphs in a similar manner to the previous example. The pattern layout again has created a tighter layout, which de-emphasises the chain of nodes that can be clearly seen on the force directed layout. Again, the force directed method has produced a very long and thin layout.

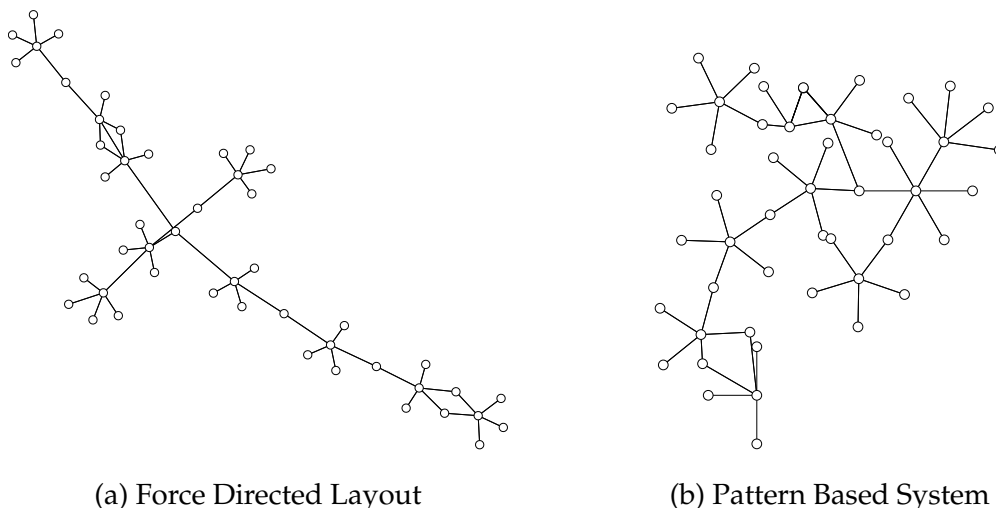


Figure 4.6: 2004 Best Leading and Supporting Actor and Actress Academy Award Nominees

The layout of the 2004 Best Leading and Supporting Actor and Actress nominees (Figure 4.6) contains two triangles and a circle, as well as many stars. The force directed layout has continued to produce a long, thin example while the aspect ratio is much more even for the pattern based layout. However, the pattern based layout has created a very small angle between edges near the bottom, and two overlapping triangles near the top. The pattern based layout has drawn many of the stars in a very regular fashion, compared to the force directed layout.

4.3.2 Formula One Teammates

The results from the various Formula One team-mates (see Section 4.2.2) datasets contain a wide variety of patterns.

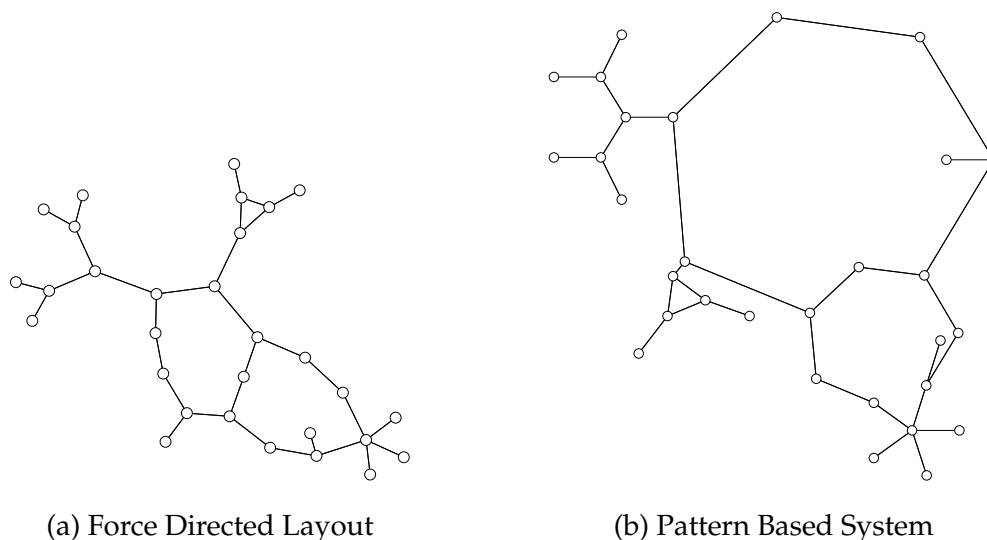


Figure 4.7: Formula One team-mates, 1996–1998

Team-mates from 1996 to 1998 (Figure 4.7) contains two circles, both of which share two nodes. As can be seen in the pattern layout, these are drawn in regular shapes, with the largest being particularly distinctive. The star is also well laid out, although the loose node connecting to it is drawn with a poor angle. The force directed layout also performs well on this example, although the irregularity of the shapes is noticeable.

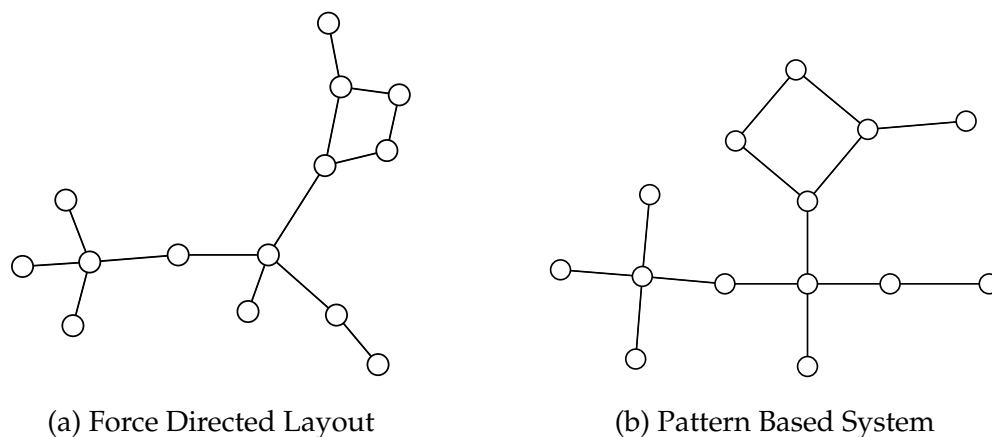


Figure 4.8: Formula One team-mates, 2006–2008

In example of team-mates from 2006 to 2008 (Figure 4.8), both drawing methods produce similar layouts. However, as in the previous example, the pattern based layout produces another very good layout, with the regular shapes of pattern being particularly noticeable.

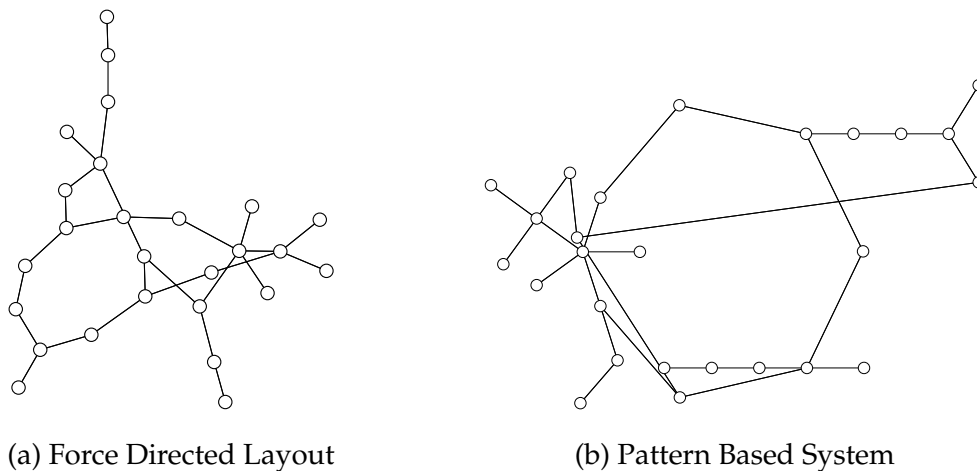


Figure 4.9: Formula One team-mates, 1995–1997

The layout of 1995 to 1997 team-mates (Figure 4.9), is radically different between the two layouts. The pattern based system has drawn a large circle, while

clustering nodes to the left, whereas the force directed layout has resulted in a more even density. The pattern based layout introduces a long edge that crosses the diagram. The circle is much more noticeable in the pattern based layout.

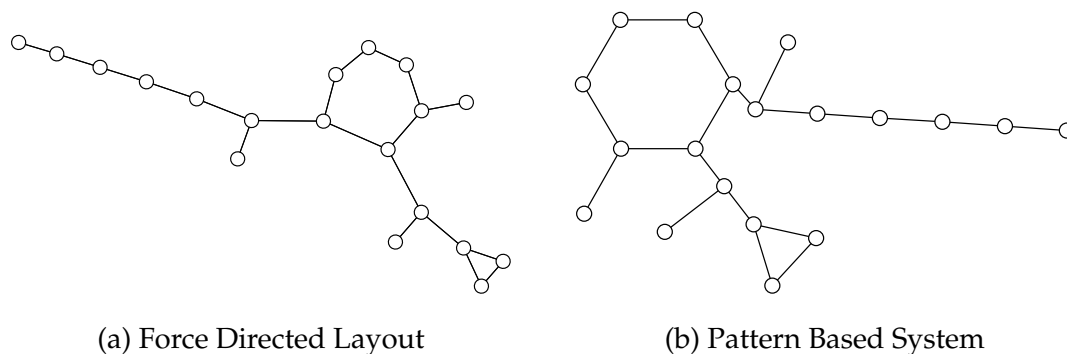


Figure 4.10: Formula One team-mates, 2013–2015

The team-mates from 2013 to 2015 (Figure 4.10) example contains a circle, triangle and path. The two methods also draw this in a similar manner, although the regular polygon layout of the circle in the pattern based system is noticeable. The regular spacing between nodes in the path is also aesthetically pleasing.

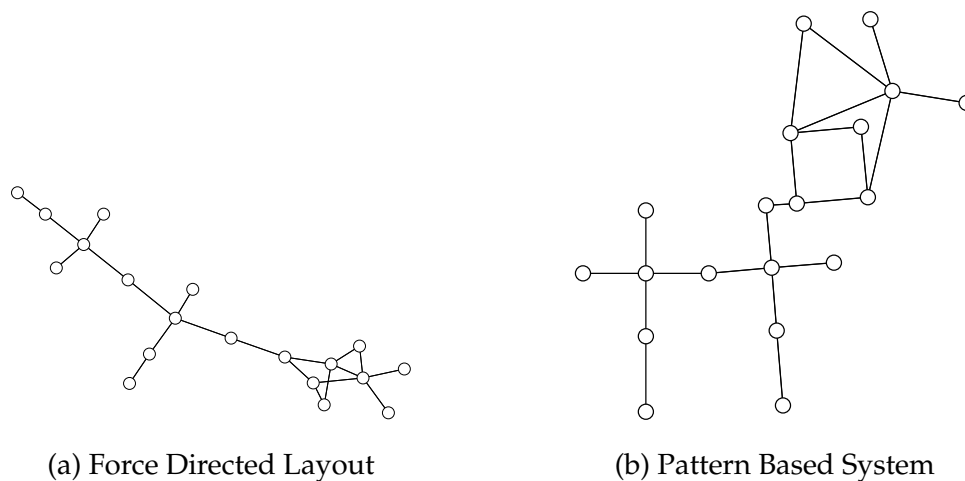


Figure 4.11: Formula One team-mates, 2002–2004

In the example of team-mates from 2002 to 2004 (See Figure 4.11), there are two stars, a circle and triangle. The pattern based method draws this example particularly well and emphasises the regular shapes of the stars, and draws these in a consistent manner. The force directed method introduces an edge crossing while it also has an irregular aspect ratio. The force based method also creates an unnecessarily dense area, resulting in an additional edge crossing.

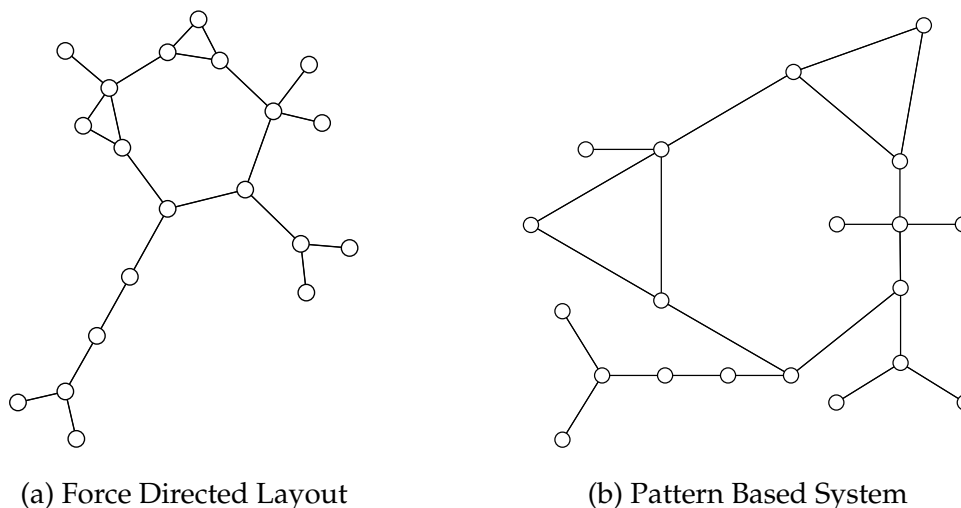


Figure 4.12: Formula One team-mates, 2009–2011

The layout of 2009 to 2011 team-mates (Figure 4.12), has a significant variation between the two methods. The large circle is strongly emphasised with two triangles being equally prominent. The triangles are not as noticeable in the force directed layout with the path drawing the eye. The pattern based layout also maintains a consistent density of nodes, while also ensuring a consistent distance between nodes. However, the path and some loose nodes are drawn at unsymmetrical angles.

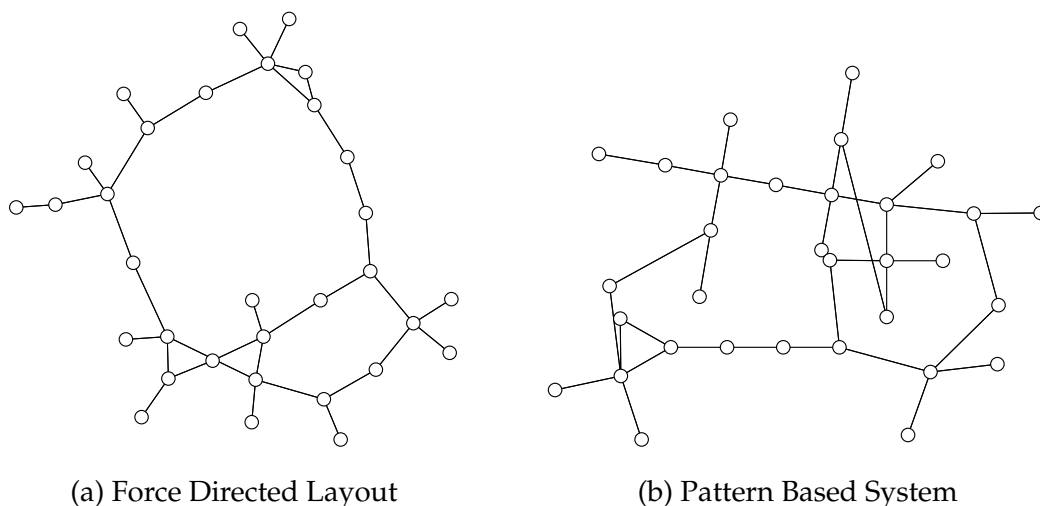


Figure 4.13: Formula One team-mates, 2005–2007

The layout of 2005 to 2007 team-mates (Figure 4.13) is vastly different between the two layout systems. The circle that is prominent in the force directed layout is not visible in the pattern based system. In this case, the circle is too large to be detected. The pattern layout does emphasise two lines of nodes, but introduces unnecessary edge crossings and occlusion.

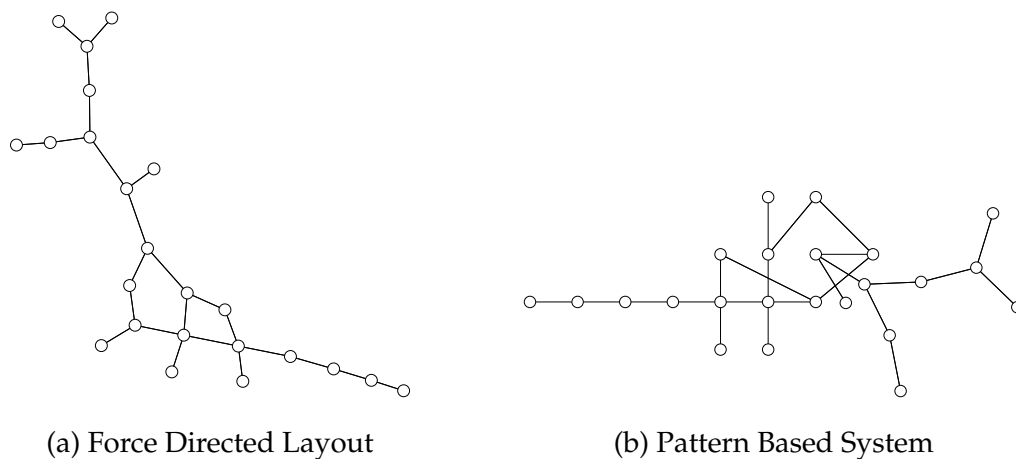


Figure 4.14: Formula One team-mates, 1998–2000

The team-mates from 1998 to 2000 (Figure 4.14) example contains a path,

star and circle. The force directed method produces a long, thin layout without any edge crossings. The pattern based layout, however does introduce these, by choosing a poor angle for a node, although draws the path in a perfectly straight line.

4.3.3 Character Relations in Novels

The results from the various character relation analysis of novels (see Section 4.2.3) datasets contain a wide variety of patterns, although mostly cliques.

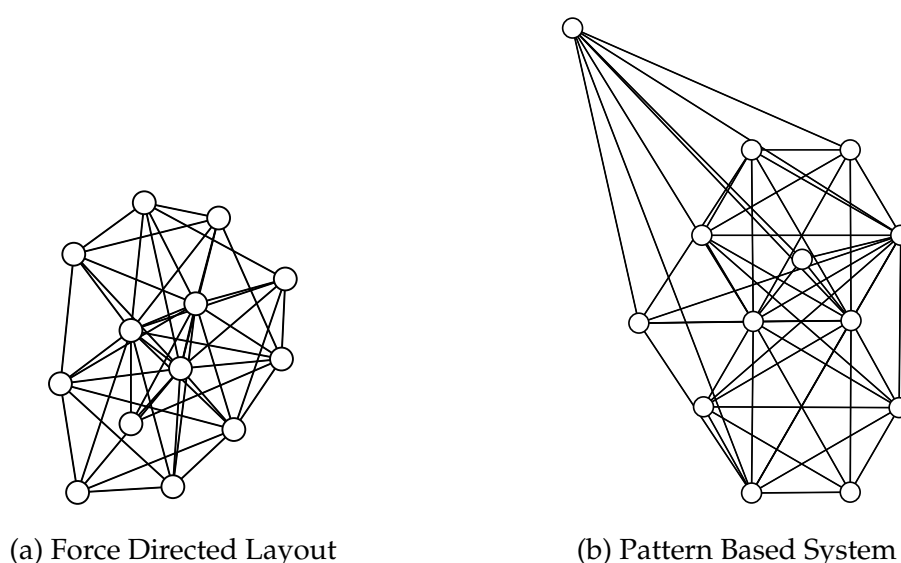


Figure 4.15: *The Jungle Book*, by Rudyard Kipling (1894)

The characters in *The Jungle Book* (Figure 4.15) are represented as a very dense graph with two main cliques. Neither system draws this layout with any excellence, although the regular layout of the cliques can be seen in the pattern based system. It seems to have abandoned one node to the top left of the diagram. The particularly noticeable feature, and one that demonstrates an advantage of the pattern based system, is that although the layout is dense, it

is immediately obvious that there are two cliques of size 6, as these are represented with two regular hexagons. This is not observable in the force directed layout, and it does take great difficulty to even determine if there are any cliques present in this layout.

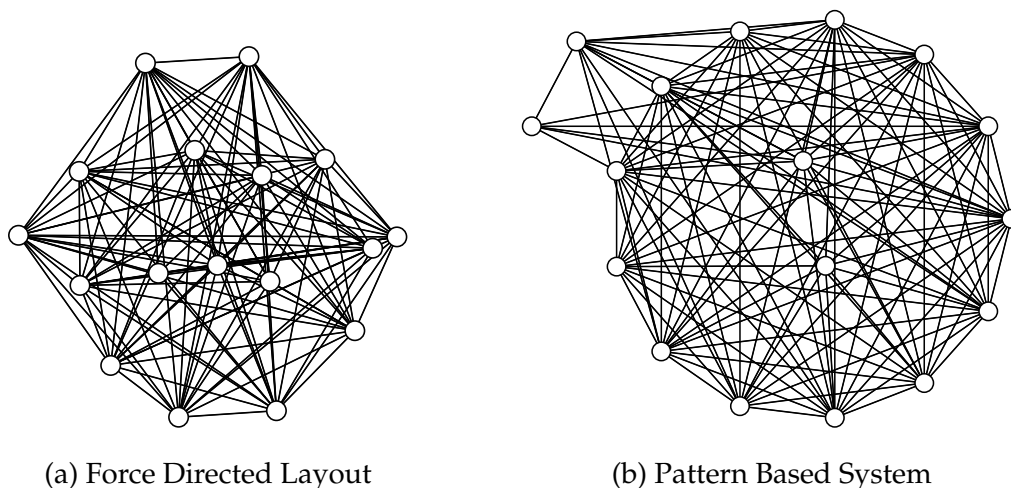


Figure 4.16: *Pride and Prejudice*, by Jane Austen (1813)

The characters in *Pride and Prejudice* (Figure 4.16) are represented as a large clique, a small clique and some scattered nodes. Again, neither system handles such a dense graph well and the nodes contained within the clique are almost impossible to identify in both layouts. However, again the regular structure of a large clique is immediately obvious in the pattern based layout, whereas it is not clear in the force directed layout.

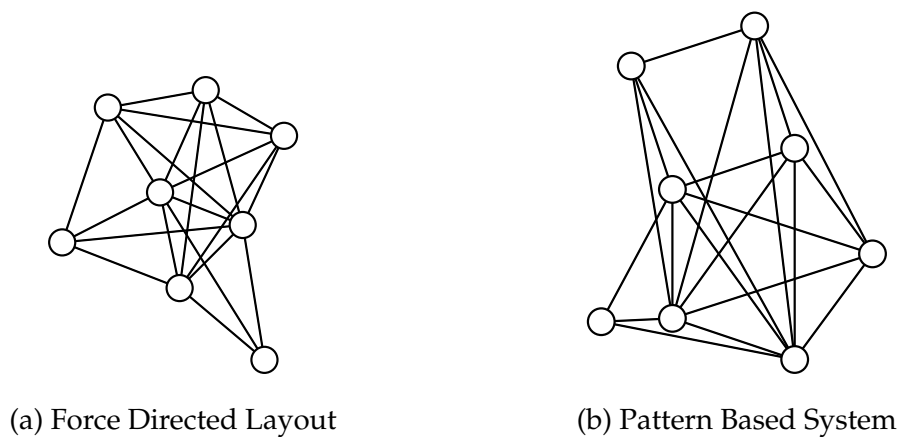
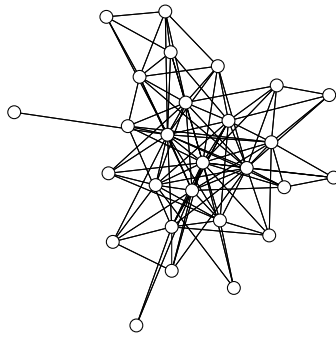
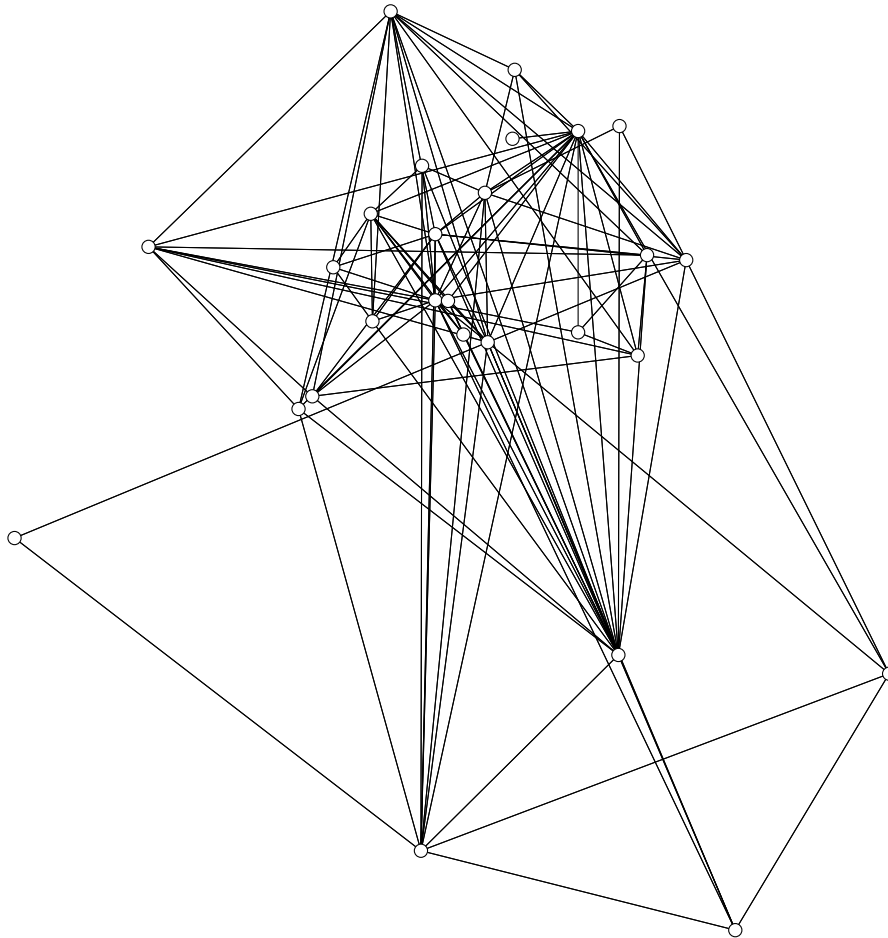


Figure 4.17: *Strange Case of Dr Jekyll and Mr Hyde*, by Robert Louis Stevenson (1886)

The characters in *Strange Case of Dr Jekyll and Mr Hyde* (Figure 4.17) are represented by two cliques, a circle and one remaining node. This much smaller example draws well in both methods, and again the regular layout of the pattern based method makes the clique much easier to identify.



(a) Force Directed Layout



(b) Pattern Based System

Figure 4.18: *Great Expectations*, by Charles Dickens (1860-1861)

The characters in *Great Expectations* (Figure 4.18) are drawn very poorly with both methods. The force directed method creates a layout where any type of connection is difficult to spot. The pattern based layout identifies a circle within two nodes some distance apart and therefore creates an enormous circle. To display this large circle results in the other nodes being impossible to identify.

4.3.4 Other Data Sources

While the previous datasets have all been used in the empirical study discussed in Chapter 5, a number of other datasets have been obtained to provide some interesting test and display examples.

4.3.4.1 *Rome Graphs*

The *Rome Graphs* are a set of over 11,000 undirected graphs ranging in size from 10 to 100 nodes, available online [1]. 100 of these were selected at random for analysis. The layouts from these graphs contain a variety of patterns, often circles and stars. Figures 4.27 and 4.28 show that although these randomly selected graphs have, on average, more nodes and edges than the other datasets, they have comparable edge densities.

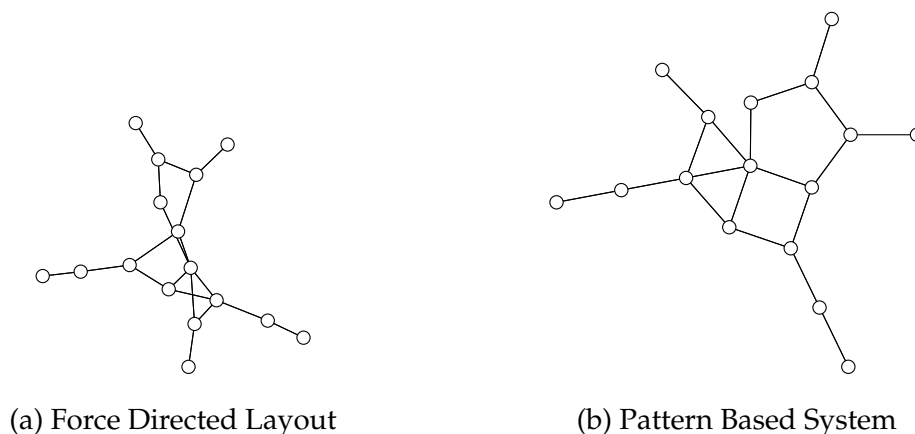


Figure 4.19: Graph 693 . 16

The layouts for Graph 693 . 16 (Figure 4.19) are similar. However, the force based layout has introduced edge occlusion and edge crossings. The regular structure of the pattern layout is particularly noticeable and results in a very clear and aesthetically pleasing layout.

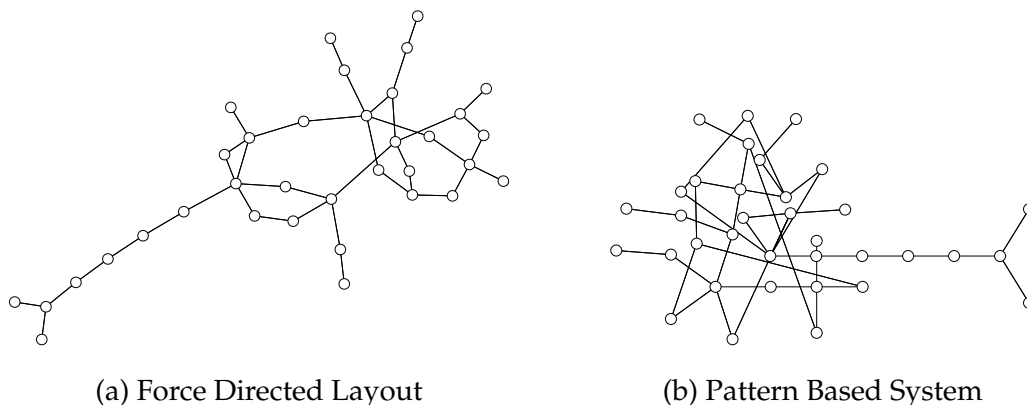


Figure 4.20: Graph 4037 . 35

Graph 4037 . 35 (Figure 4.20) produces two different layouts. The force directed layout is much clearer than the pattern based layout, which introduces a number of node-edge occlusions and edge crossings. However, the regular shape of stars, and the straight path are also noticeable.

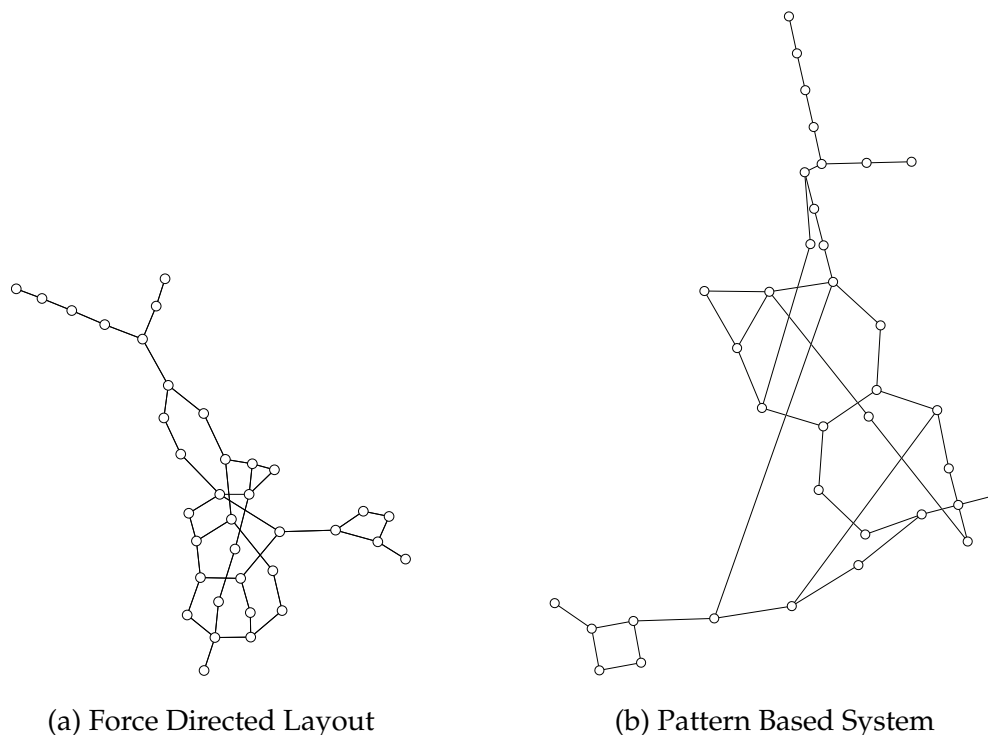
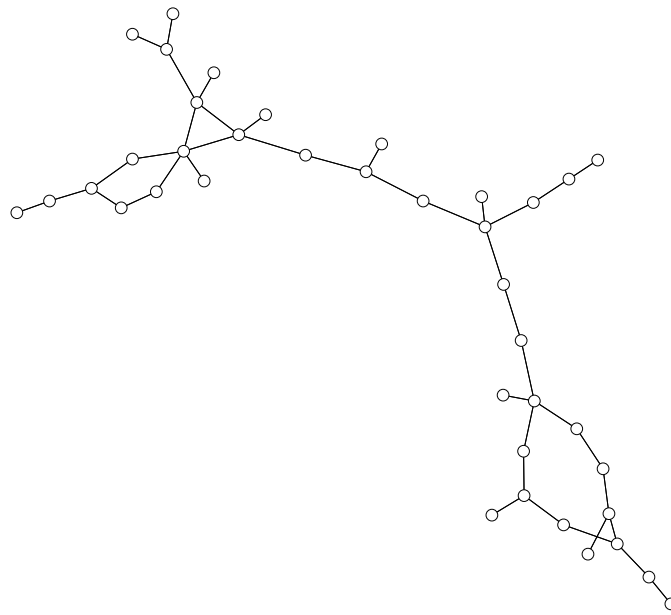
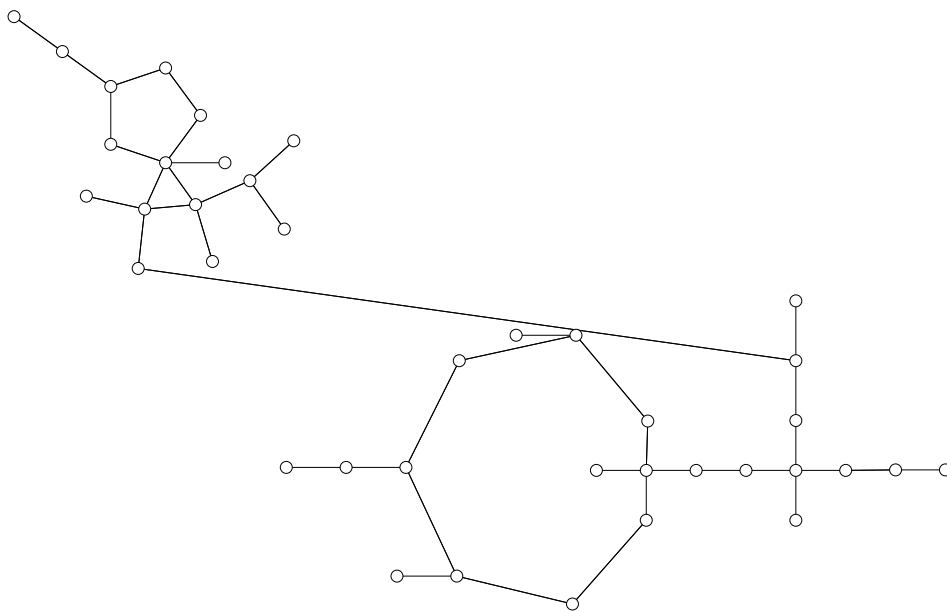


Figure 4.21: Graph 6028.36

The force directed method for Graph 6028.36 (Figure 4.21) produces a much more compact layout than the pattern based method. Both systems have edges that cross the layout, and the pattern based system also chooses angles which are too small. Such a layout decision would not be made by the force directed algorithm as those nodes would be considered “too close” and would have strong repulsive forces. The circles and path are much clearer on the pattern based layout than on the force directed layout.



(a) Force Directed Layout



(b) Pattern Based System

Figure 4.22: Graph 6952.39

The layouts of Graph 6952.39 (Figure 4.22) result in some interesting differences. The force directed layout produces another long, thin layout, whereas the pattern based method produces a layout with a better aspect ratio. Oddly, the pattern based method has drawn a number of patterns over to the left, resulting in a very long edge that almost occludes a node. The large circle dominates in a way that it does not in the force directed layout, and the collection of paths and stars towards the right is drawn with each node evenly spaced between its neighbours. There is a tight angle on a loose node on the top of the circle - this is drawn in this manner to avoid an edge crossing with the long edge. A better placement of the right-hand cluster of patterns would have allowed a larger and more symmetrical angle for edge between the loose node and the circle.

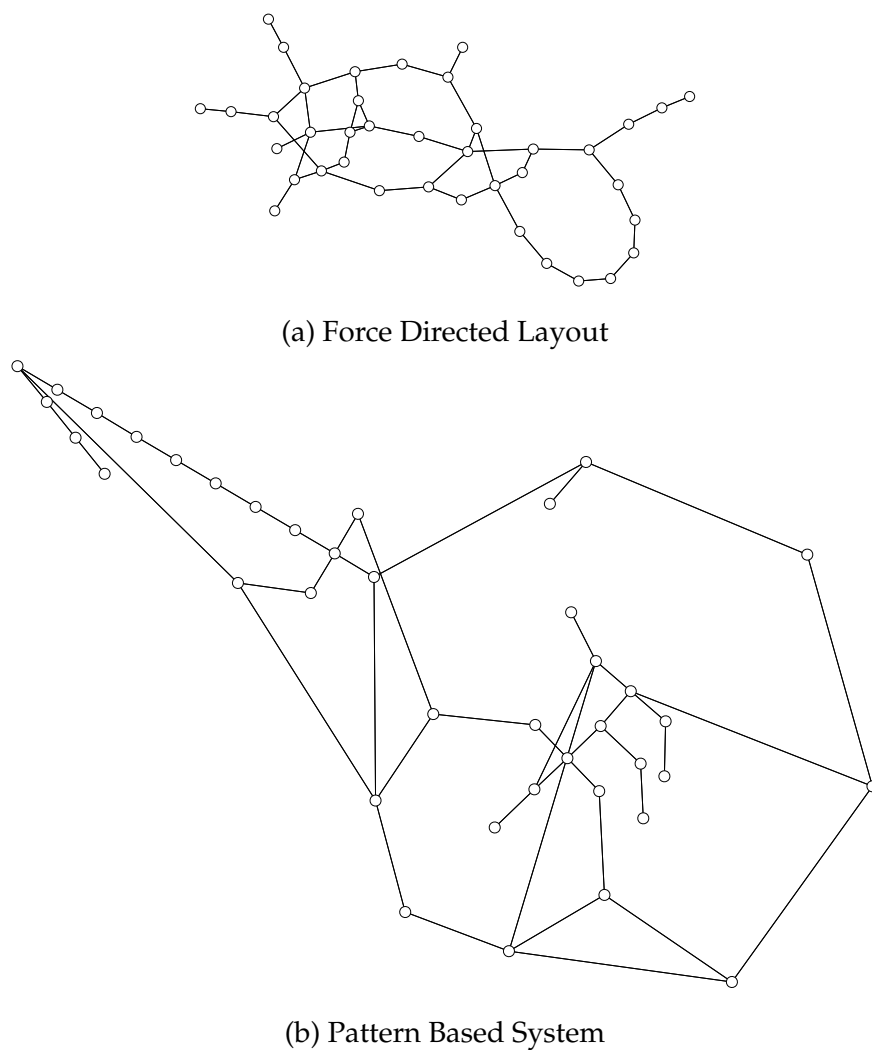


Figure 4.23: Graph 11669.39

Graph 11669.39 (Figure 4.23) also produces two layouts which are substantially different. The force directed method produces a smaller and denser drawing, with one node-edge occlusion and a few edge crossings. The pattern based method interestingly draws a circle and a number of stars inside a large circle. It also draws a large path, which results in a long edge - this is due to a circle being too large to detect, and is represented in a more aesthetically pleasing way in the force directed layout.

4.3.4.2 1960s and 1970s NASA Space launches

An example can be found creating a connection between NASA astronauts that have launched on space missions together in the 1960s and 1970s (Figure 4.24). With the pattern method, the 3-man Apollo missions can easily be seen as triangles, all of similar size, with other connections being the 2-man Gemini launches. The triple person missions are not so clear in the force directed layout.

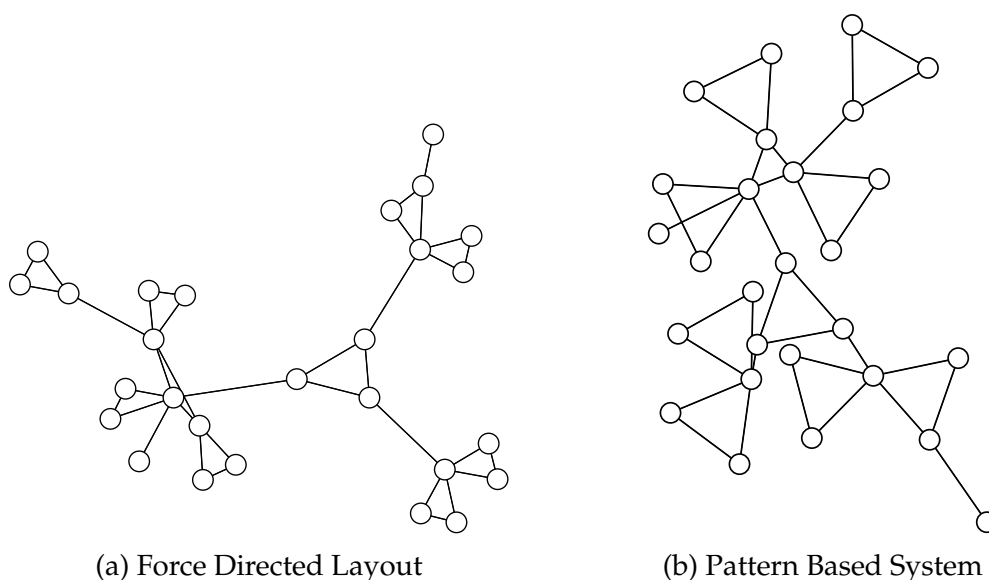


Figure 4.24: Crew of 1960s & 1970s NASA Space Launches

4.3.4.3 Chemicals

Chemicals are often drawn in a skeletal layout. This has a specific notation in the field of chemistry, and although neither method are optimised for the specific rules for this field, it is interesting to compare some results from the two drawing methods (See Section 6.2.3).

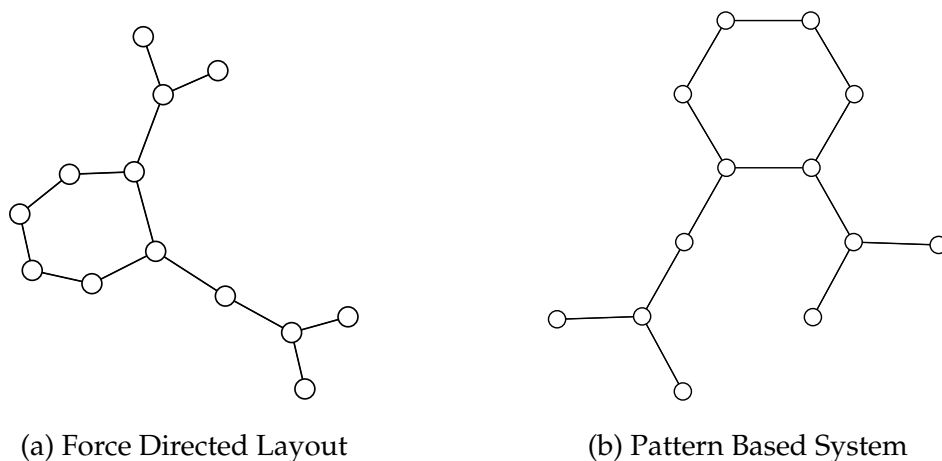


Figure 4.25: Skeletal Layout of Aspirin

Aspirin is a common painkiller and the force directed and pattern based layouts both return similar results. The benefit of drawing patterns in regular shapes is clear with the circle being strongly emphasised.

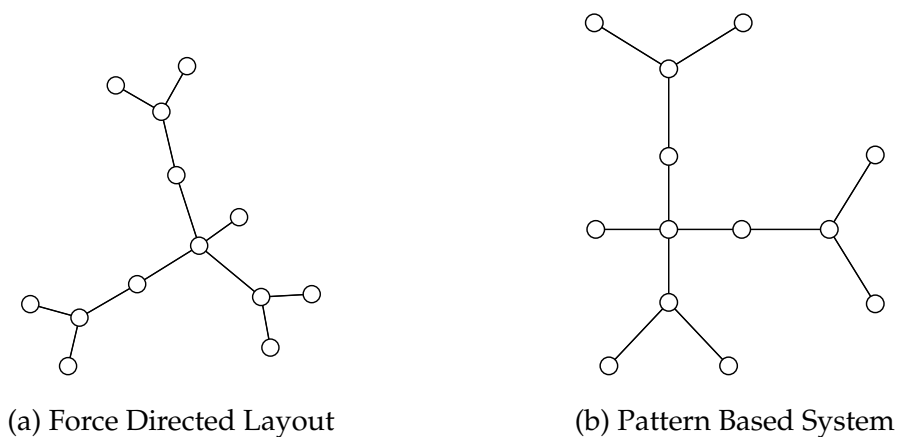


Figure 4.26: Skeletal Layout of Citric Acid

Citric acid is another common substance found in citrus fruit such as lemons and oranges. The pattern based system results in two visually strong lines, one horizontal and the other vertical. Otherwise, the force directed method draws in a similar manner.

4.4 Quantitative Analysis of Examples

Displayed below are aggregate charts that various metrics for each of the datasets described in Section 4.2 and the *Rome graphs* described in Section 4.3.4.1, when drawn by the two methods. Full charts of every example in the datasets are available in Appendix A and all averages are have been calculated using the arithmetic mean.

Figure 4.27 shows the average number of nodes and edges in each of the datasets. As can be seen, many have a similar number of nodes and edges, although Books have considerable more edges.

From the previous graph, it is no surprise that Figure 4.28 shows that the Books dataset has a higher density than the others. For this example, density is measured using a common definition in the field [29]:

$$\frac{2|E|}{|V|(|V| - 1)}$$

Figure 4.29 shows a summary of the time taken for each dataset when drawn with the pattern based system. Most examples are completed within 5 seconds, however there are some outliers which can take longer to complete. There is no comparison with the force directed layout as that method typically runs for a certain number of iterations. The timing data was calculated on a machine running Windows 10, with an AMD X4 965 3.4GHz Quad-core processor with 8GB of RAM.

Shown in Figure 4.30 is the time taken and number of nodes for every example in all the datasets. As can be seen, the time taken remains roughly constant until around 60 nodes are reached, when the time taken begins to increase rapidly.

Figure 4.31 is the time taken and number of edges for every example in all

the datasets. In a similar manner to Figure 4.30, the time taken remains roughly constant until around 60 edges are reached, at which point the time taken increases.

The average number of patterns for each dataset is shown in Figure 4.32, grouped by type. The Books dataset is the only to contain cliques, while the Academy Awards datasets contain numerous stars. The *Rome graphs* contain a wide variety of different patterns.

Figure 4.33 shows the average percentage of found patterns that are drawn or discarded. Discarded patterns are those which do not match any of the connection types. As can be seen, the Books and *Rome* datasets both discard over half the identified patterns.

The average percentage of nodes in drawn patterns, nodes only in discarded patterns, and nodes in no patterns are shown in Figure 4.34. Around 1 in 3 nodes in the Formula One dataset are not included in patterns. Interestingly, although a large number of patterns are discarded (see Figure 4.33) for the Books and *Rome* datasets, most nodes are in patterns which are drawn. This suggests these datasets contain a large number of overlapping patterns, something which is to be expected considering the edge densities of these datasets (see Figure 4.28).

Figure 4.35 shows the average number of instances of node-node occlusion in each dataset for the pattern based and force directed methods. Both systems are comparable in the Books dataset, where the high connectivity results in difficult layouts. In the other datasets, the pattern based system results in node-node occlusions whereas the force directed method does not. However, these values are relatively low, with the highest being just over 0.7 instances of occlusion for the *Rome* dataset.

The average number of instances of node-edge occlusion in each dataset for

the force directed and pattern based methods are shown in Figure 4.36. The force directed method performs worse than the pattern based method for the Books dataset, while the reverse is true for the *Rome* dataset. It is interesting to note this, as unlike the pattern based system, the force directed method makes no attempt to optimise for this metric.

Figure 4.37 shows the average number of edge crossings for each dataset when drawn with the two layout methods. Both methods produce comparable results, with the Books dataset having a large number of edge crossings. This is due to its density and smaller number of nodes which limits the potential number of layouts.

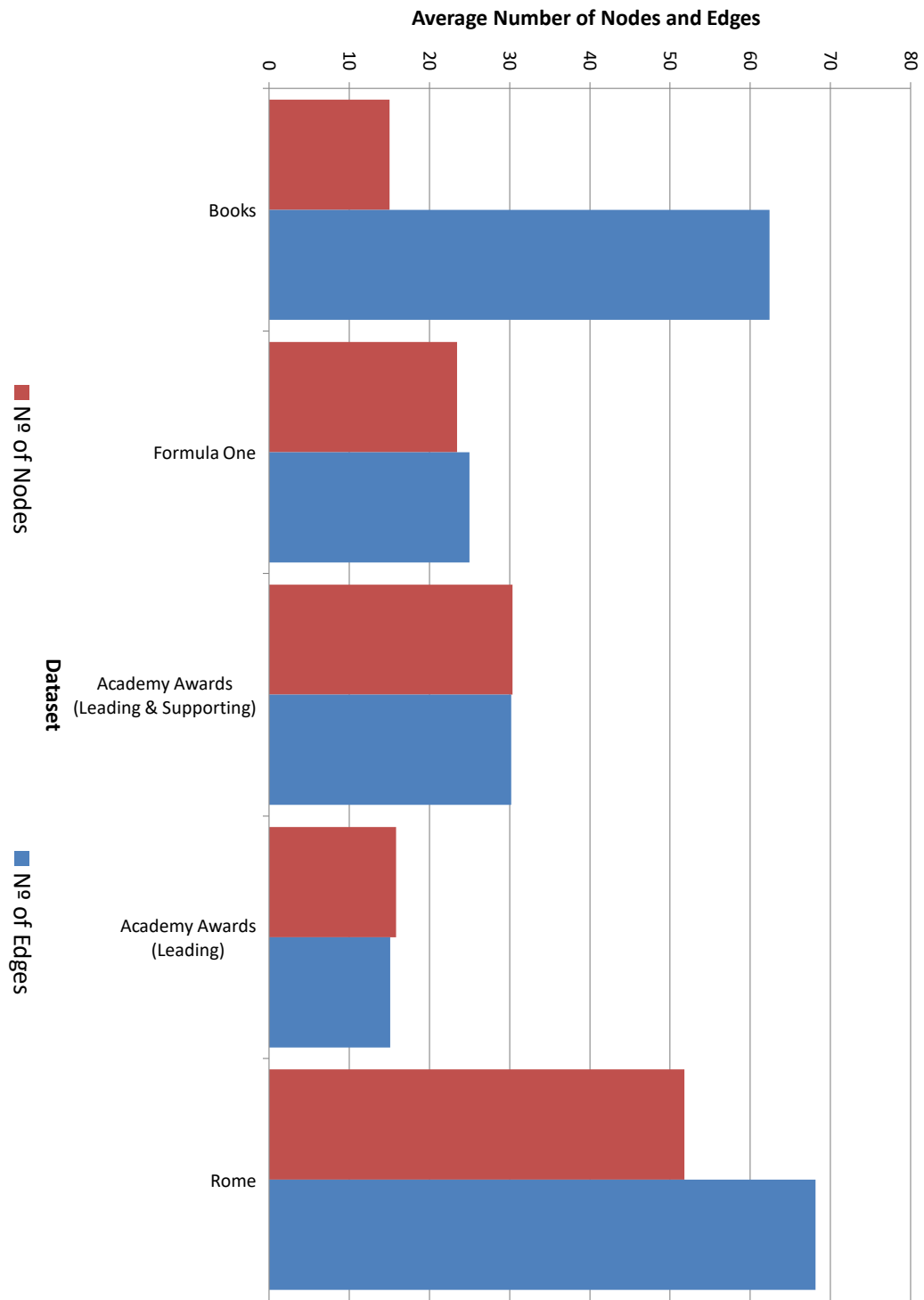


Figure 4.27: Average Number of Nodes and Edges

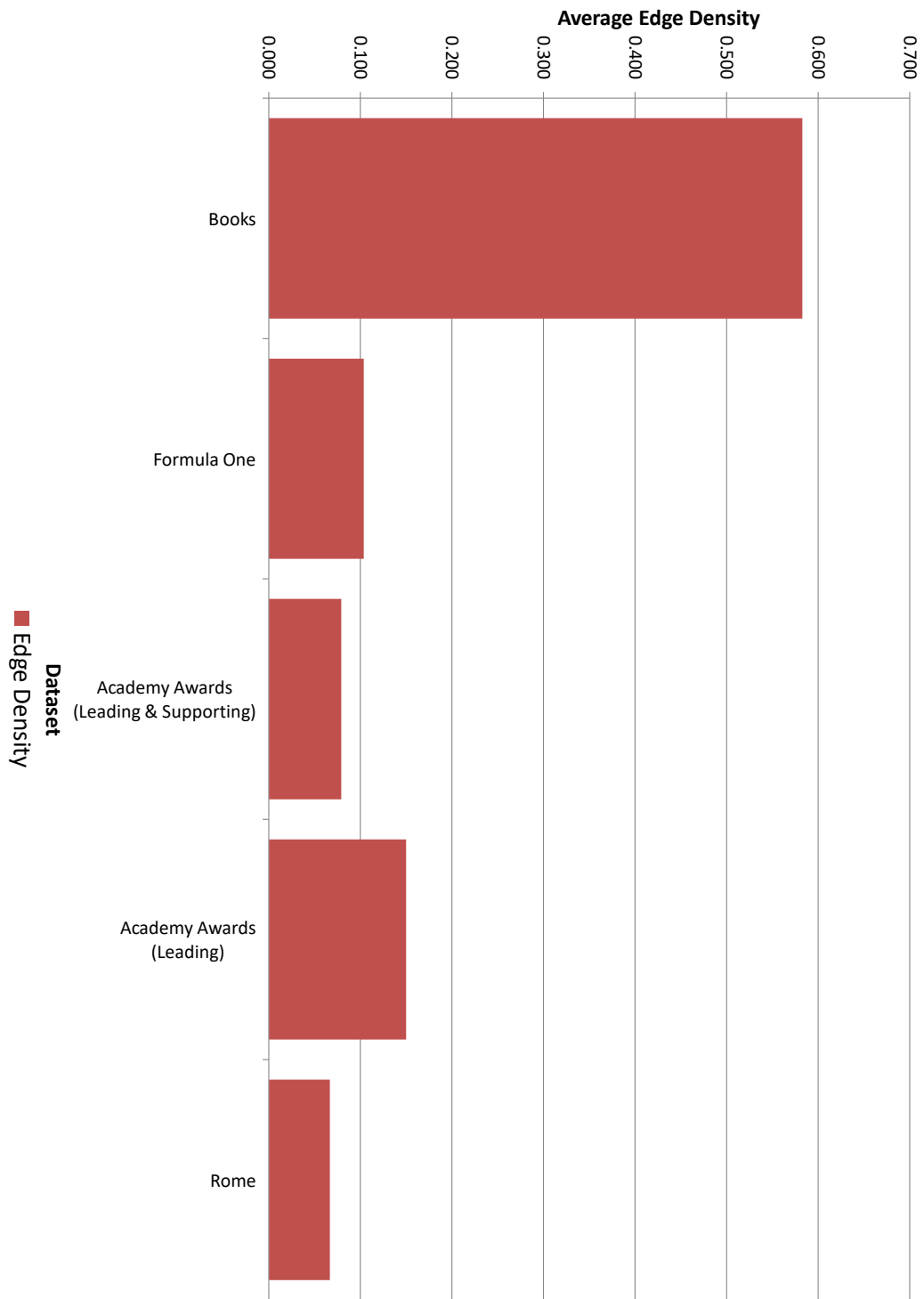


Figure 4.28: Average Edge Density

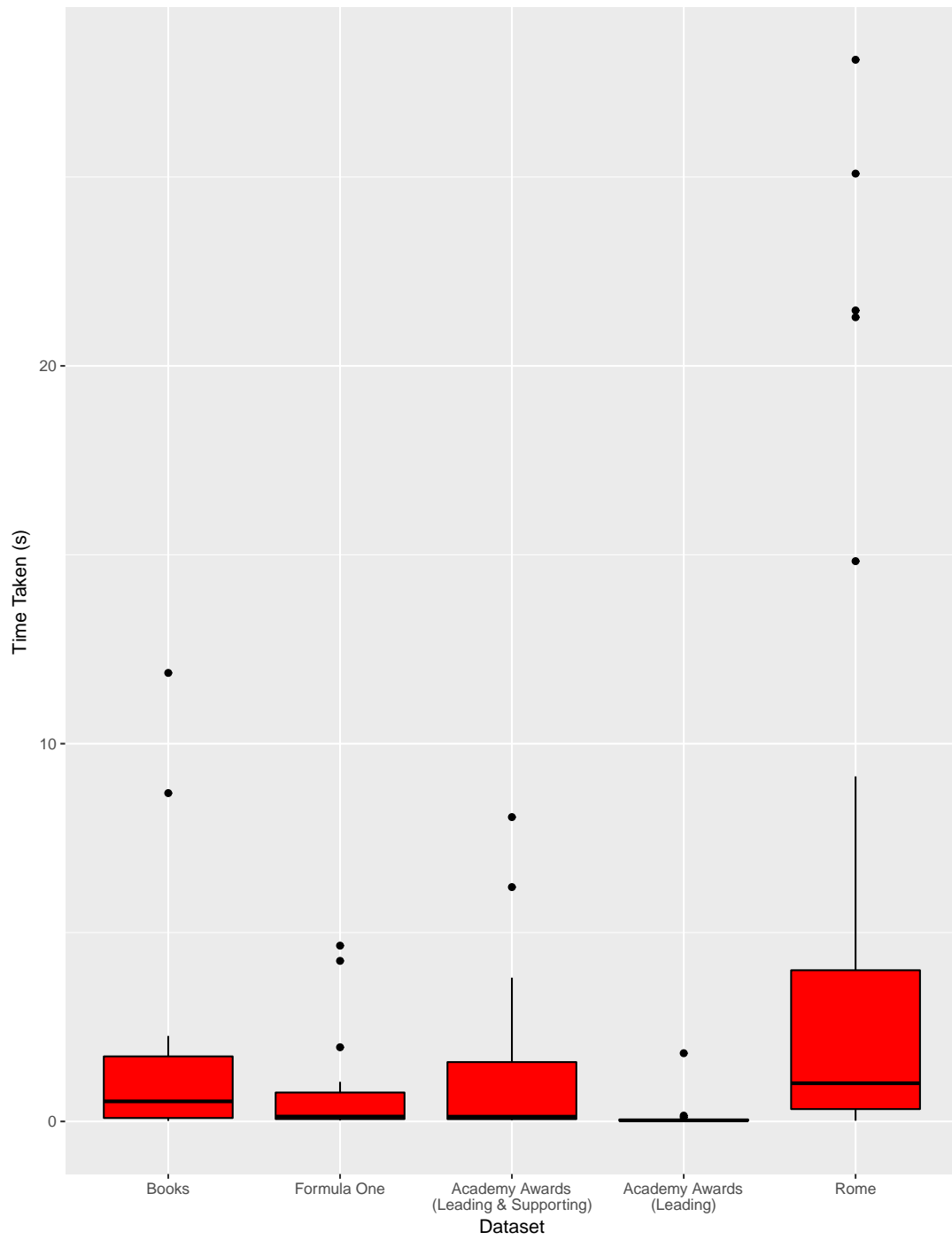


Figure 4.29: Summary of Time Taken (s)

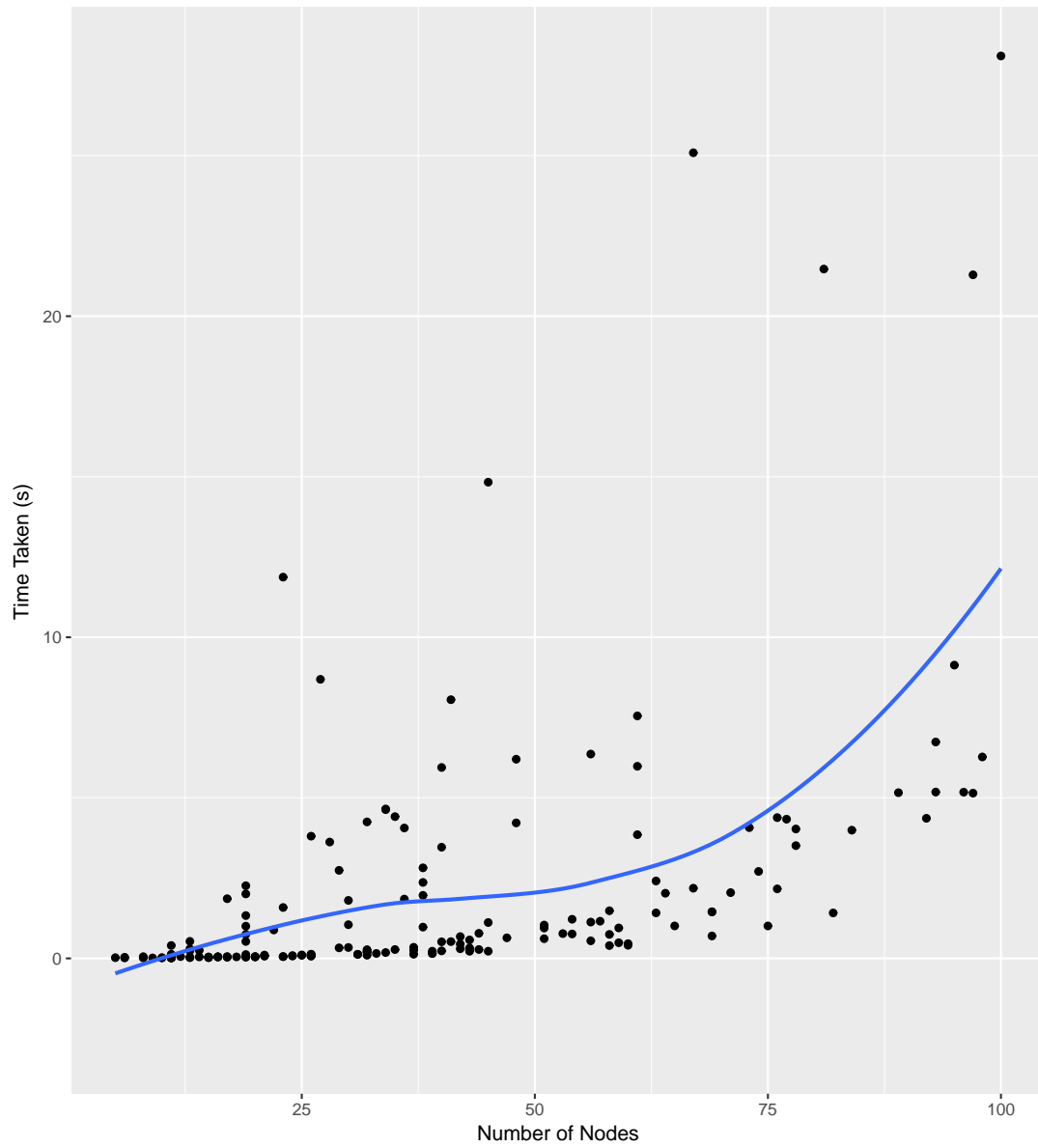


Figure 4.30: Time Taken (s) vs Number of Nodes

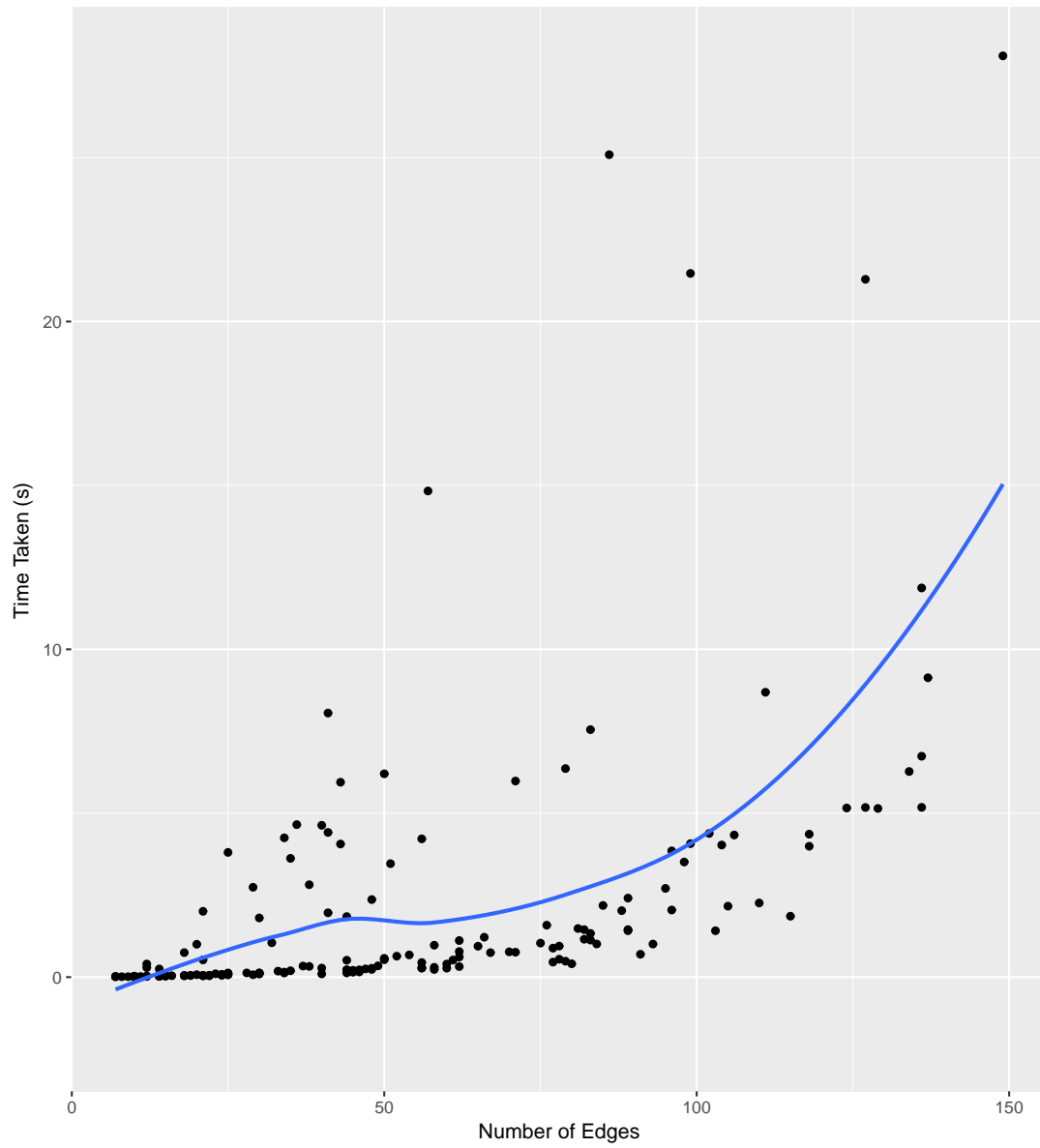


Figure 4.31: Time Taken (s) vs Number of Edges

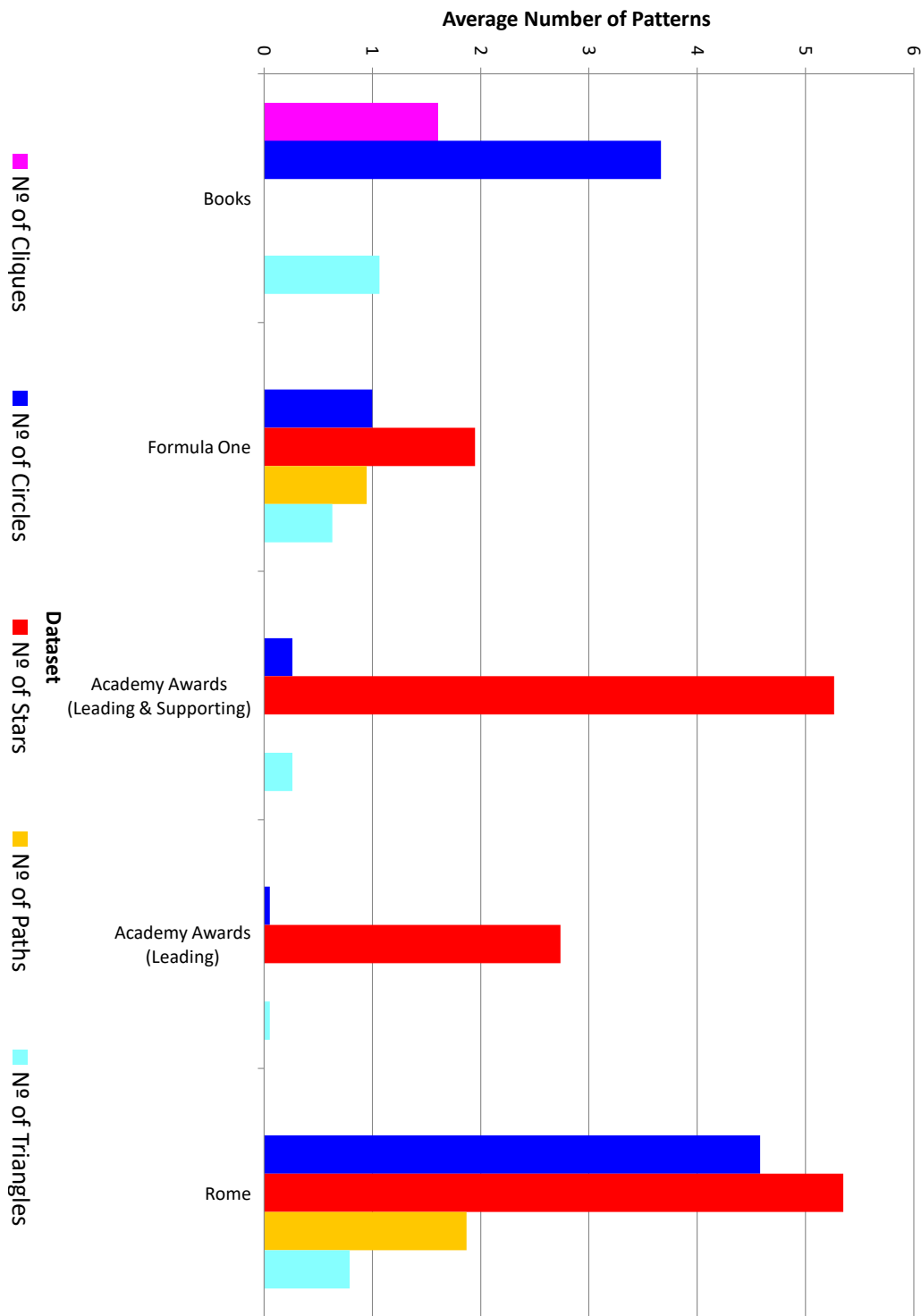


Figure 4.32: Average Number of Each Type of Pattern

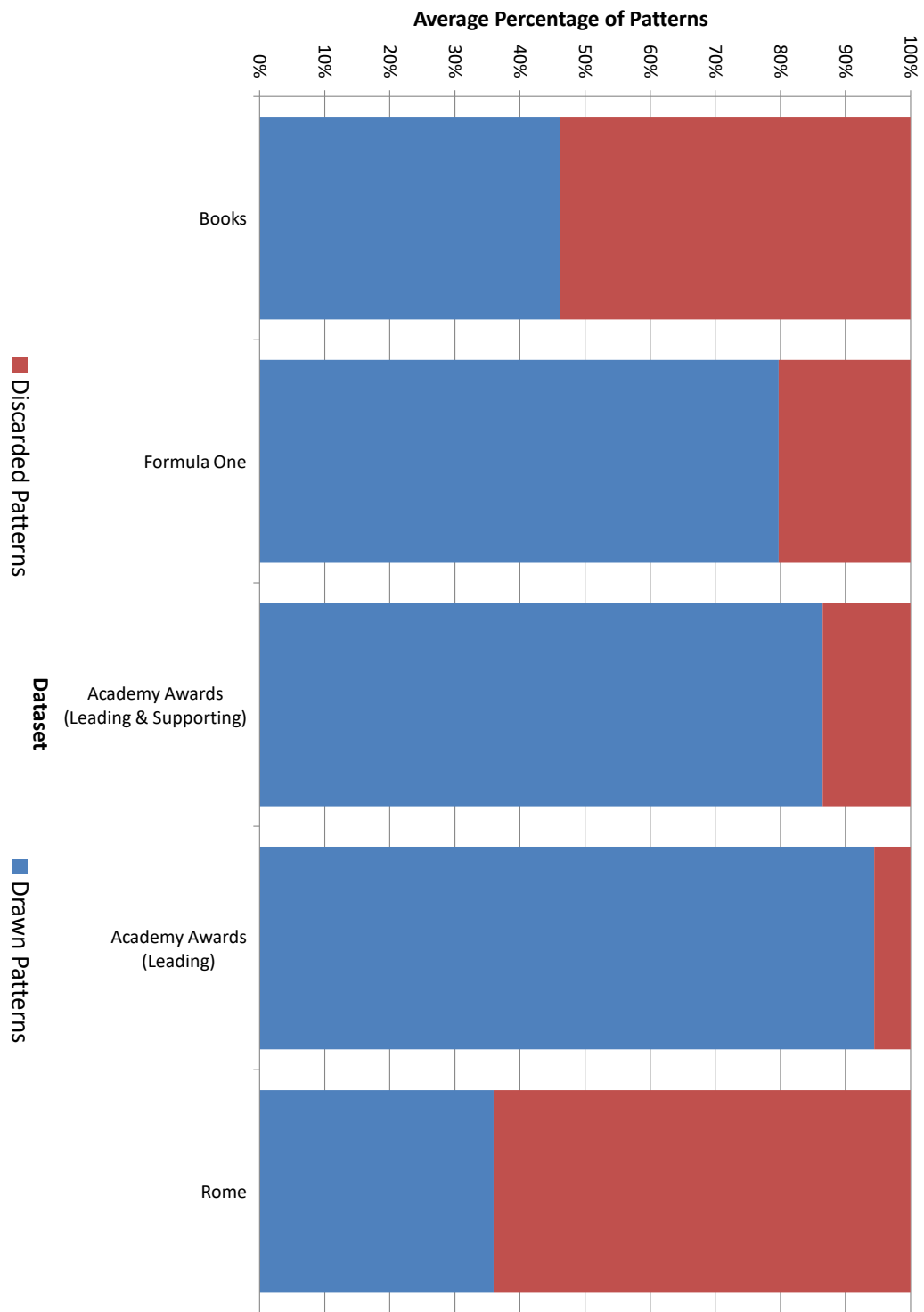


Figure 4.33: Average Percentage of Patterns Drawn or Discarded

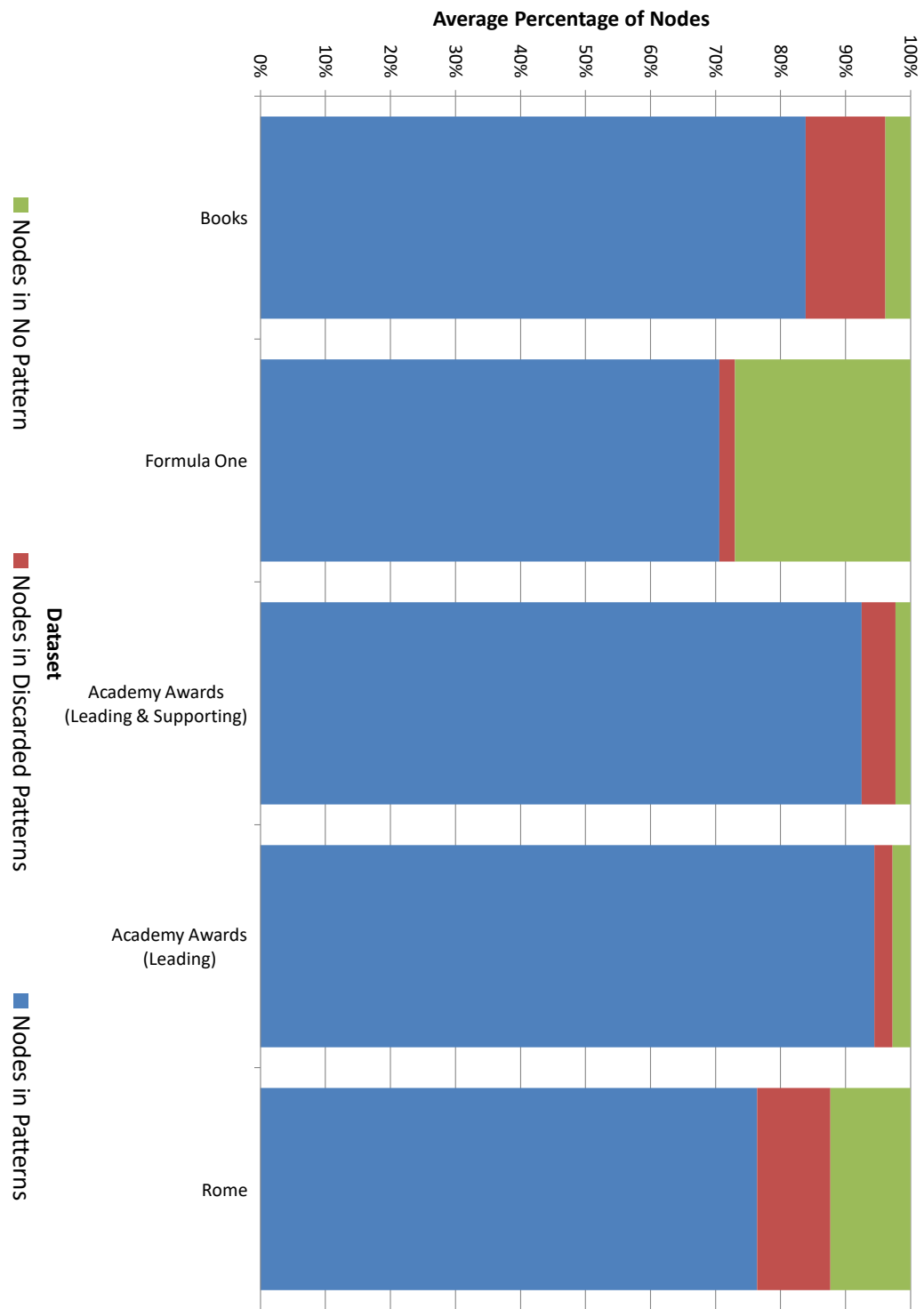


Figure 4.34: Average Percentage of Nodes in Drawn or Discarded Patterns or in No Pattern

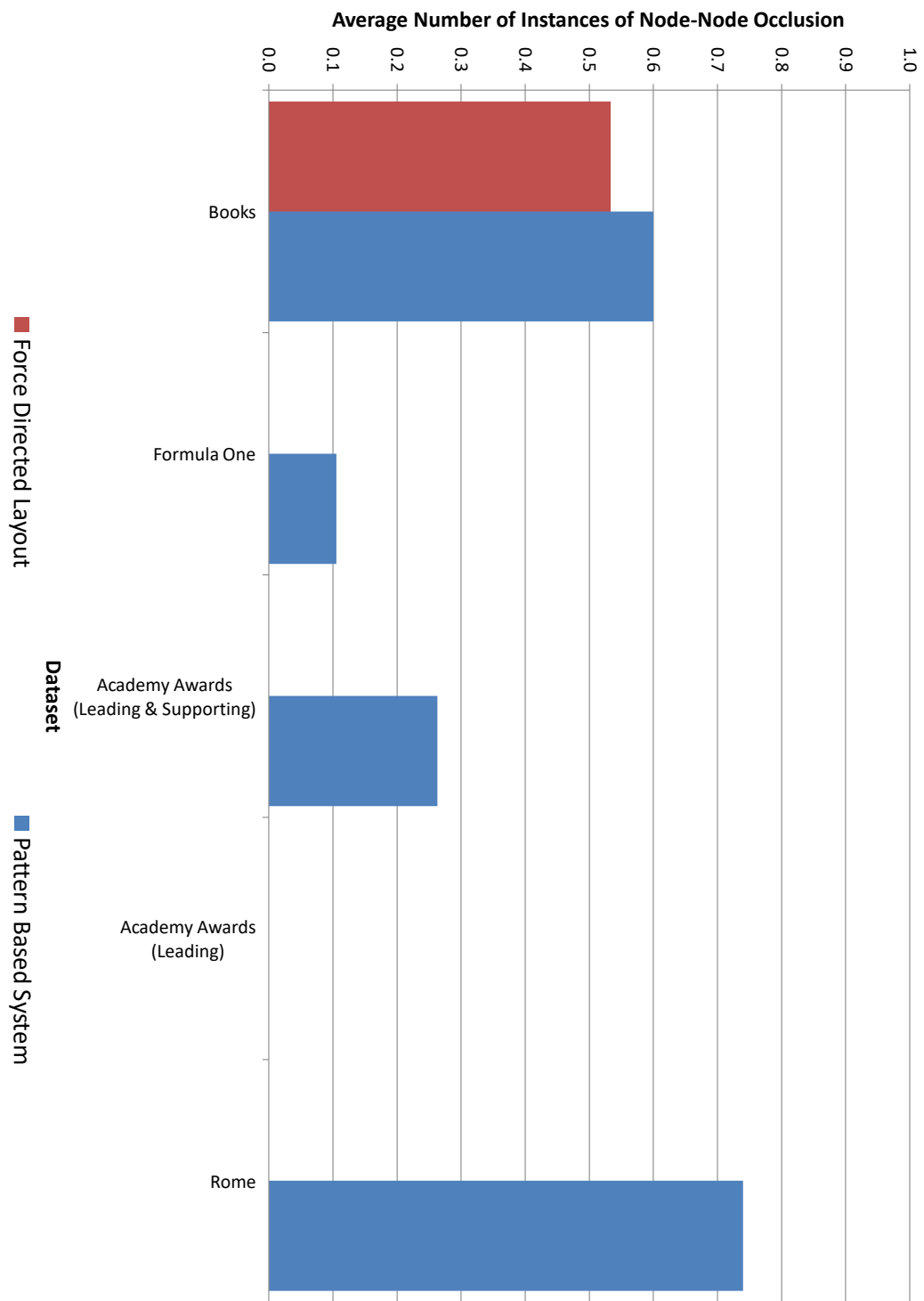


Figure 4.35: Average Number of Instances of Node-Node Occlusion

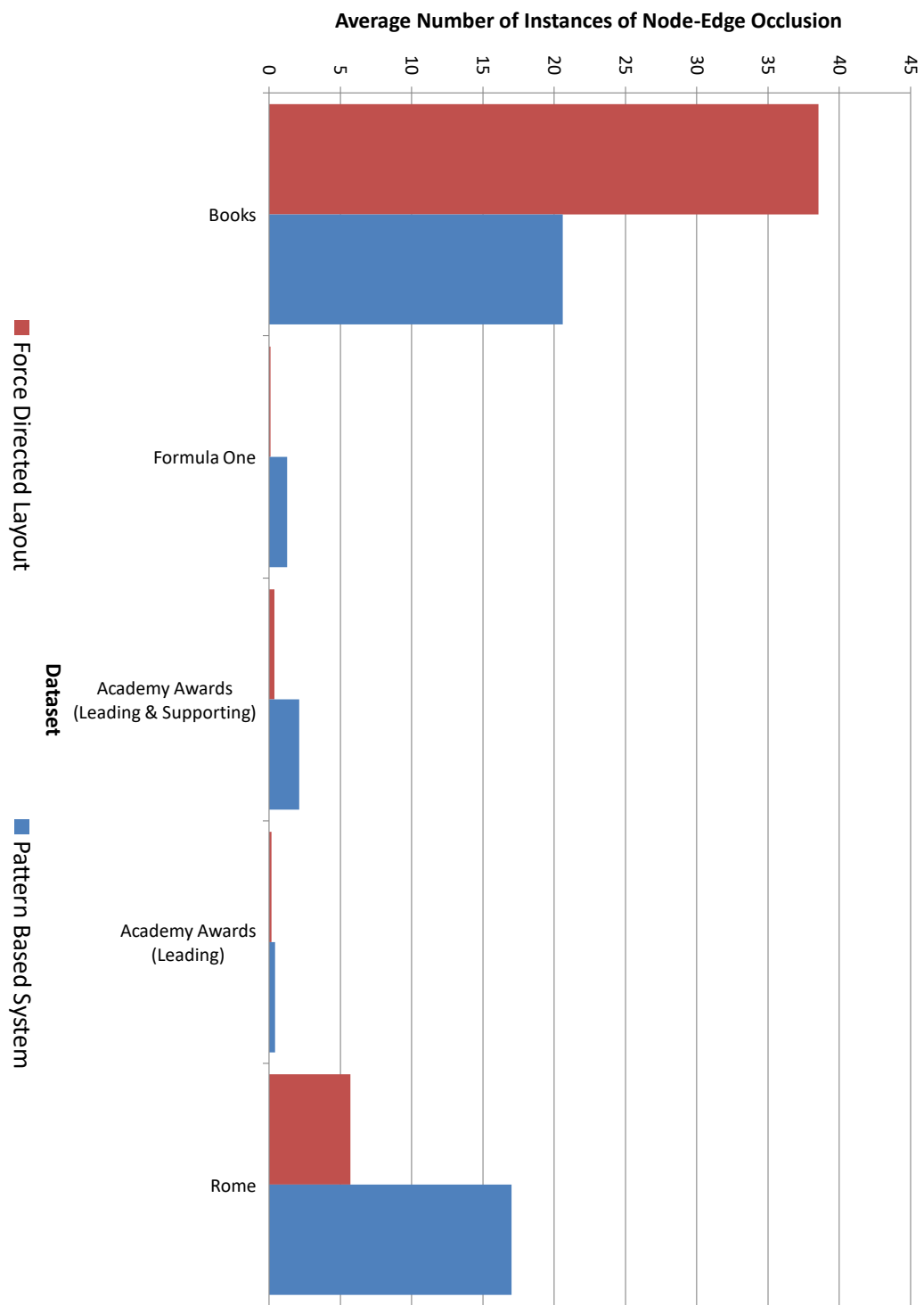


Figure 4.36: Average Number of Instances of Node-Edge Occlusion

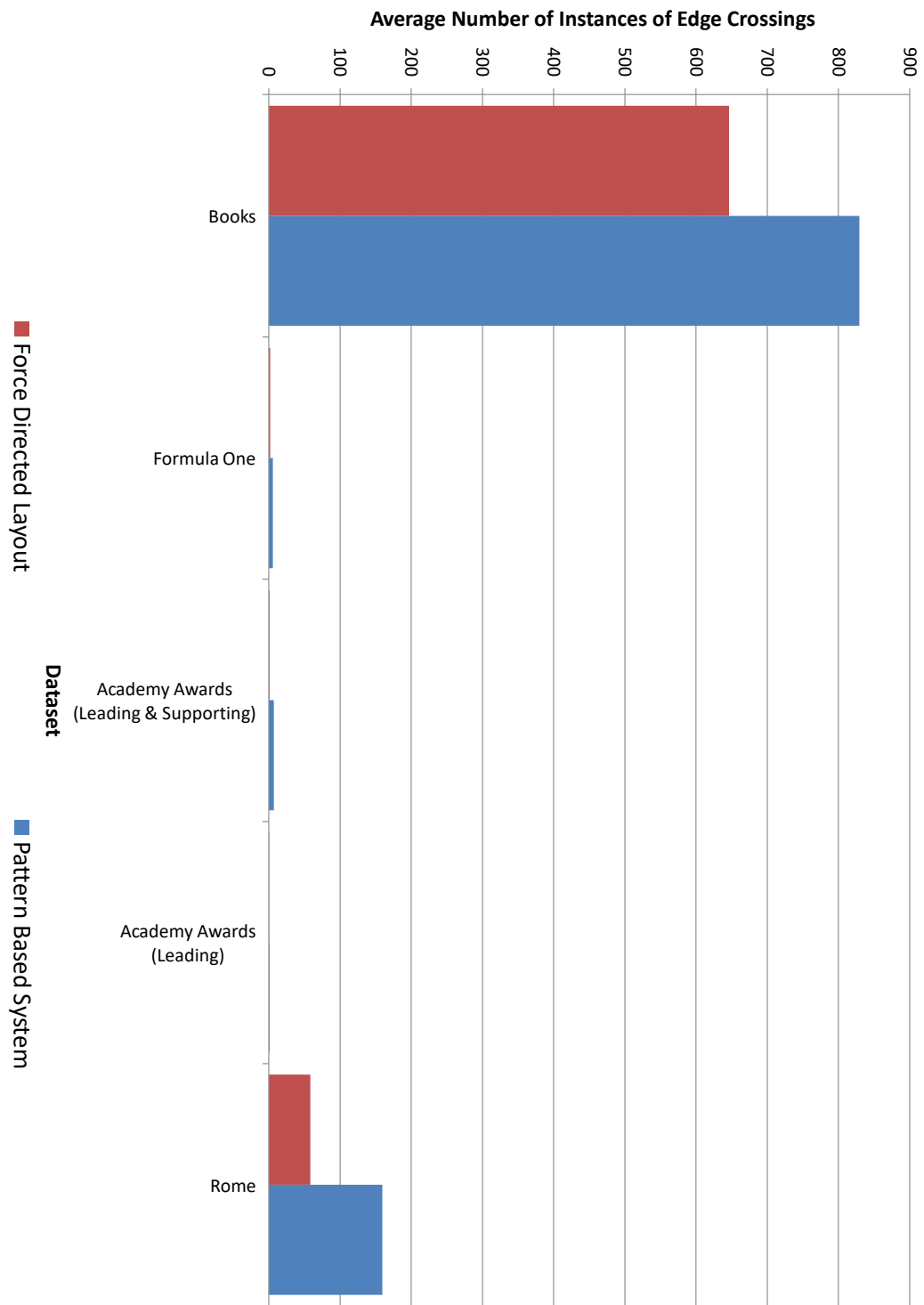


Figure 4.37: Average Number of Edge Crossings

4.5 Summary

This chapter has detailed the methodology used to create a number of datasets for comparing this pattern based layout to a force directed method. Such examples were discussed before a comparison was performed using various metrics.

From the numerous examples in Section 4.3, the pattern based system creates some interesting and aesthetically pleasing graph layouts. Many have more even aspect ratios, although the pattern based method is prone to creating “tighter” layouts that result in unnecessary occlusions or edge crossings. The angles between nodes are also, at times, unnecessarily small and this can also lead to issues. However, the consistent ideal layouts of patterns are very noticeable in most examples, with the regular shape of cliques particularly noticeable on very dense graphs. This allows the user to determine the size of clique even if there are large numbers of edge crossings. The regular spacing and straight lines of paths are also a common feature. The pattern based method does perform well, although it appears to make poor decisions at times that can compromise the rest of the layout.

As can be seen in Section 4.4, the pattern based system performs favourably to a force directed method when comparing node-node occlusion, node-edge occlusion and edge crossings. It is interesting to note that the pattern based system has more node-edge occlusions for *Rome* dataset, but fewer instances in the *Books* dataset. It is also clear the *Books* and *Rome* datasets have a large number of overlapping patterns. This is shown by a large number of discarded patterns (see Figure 4.33), yet a large percentage of nodes were contained within drawn patterns (see Figure 4.34).

Chapter 5

Formal Empirical Study

It is important to investigate the effectiveness of this layout method, and therefore it was compared to the most widely used general graph drawing method: a force based layout. To fairly and accurately measure the relative effectiveness, a lab-based empirical study was performed.

5.1 Introduction to Study

In order to answer the research question

- “Is a pattern based layout more effective than a force based layout?”,

a formal study was completed. The hypothesis for the experiment was that drawing a graph using a pattern based layout rather than a force based layout improves the understanding of the graph. Two ways of calculating the understanding are by measuring the time taken to complete certain tasks and the accuracy of the completion of these tasks.

5.2 Methodology

5.2.1 Methodology Details

The study was conducted on a within-subject basis, with University of Kent students who had replied to an advert. Upon entering the study room, subjects were given a list of instructions, which were also read aloud. Users were asked 8 practice multiple choice questions, 4 of which were simple examples and 4 were examples from medium-sized real world data.

In the pilot study, users completed the practice questions on their own, however there were large error rates for Task Type 2 (around 70%). It is believed this was because a number of participants did not understand the question. Therefore, the methodology was changed such that the study leader would talk through the practice questions with the participant to ensure they understood all questions. This was scripted to minimise the risk of different participants having different instructions (See Appendix B.1.2.1). The answers to the practice questions were shown to the user after each question and the results of these were discarded.

This practice session was followed by 24 multiple choice questions (4 task types, 3 sizes from 12 pieces of data, each drawn with both methods). The order of questions was randomised, with the exception that the graphs used for the same task and size did not appear within 6 questions of each other. This semi-random order and practice questions help to minimise any learning effect that may be present. For each question, any nodes and labels referenced in the question were highlighted in blue, in order to remove any time taken to find nodes, as labels were placed manually. The nodes used in questions were picked at random from a subset that matched certain conditions, as this reduces the likelihood of any bias when selecting questions. For example, for Task Type

1, only nodes with between 3 and 6 connections were considered for selection; this was applied consistently as much as possible so that, for example, all Task Type 1 questions had the answers 3, 4, 5, or 6. A full list of questions and their respective graphs is provided in Appendix B.

Participants were allowed to take as long as they wished to complete each question, although they were aware the time was being recorded (but this was not displayed to the user). The study was completed by participants using custom software, a screenshot of which is shown in Figure 5.1¹. Participants were free to move the mouse, keyboard, monitor and chair to their preferred positions in order to be as comfortable as possible. The software ran using Google Chrome in fullscreen mode, with no other distractions on screen, and was displayed on a large high definition monitor in order to make the examples as clear as possible.

All participants performed the study in the same location in the same room (see Figure 5.2). The lighting was kept to similar levels through the study, by closing blinds and using artificial light, and other environmental conditions were kept as consistent as possible.

¹Available to download from: <http://www.eulerdiagrams.com/pattern>

User Study

Question 6 of 24

How many other people are there in the shortest route between Isaac & Doris (both shown in blue)?

- 3
- 4
- 5
- 6

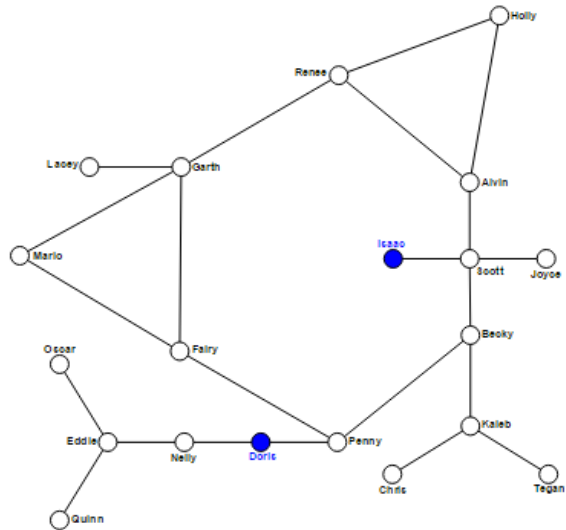


Figure 5.1: Screenshot of Study Software

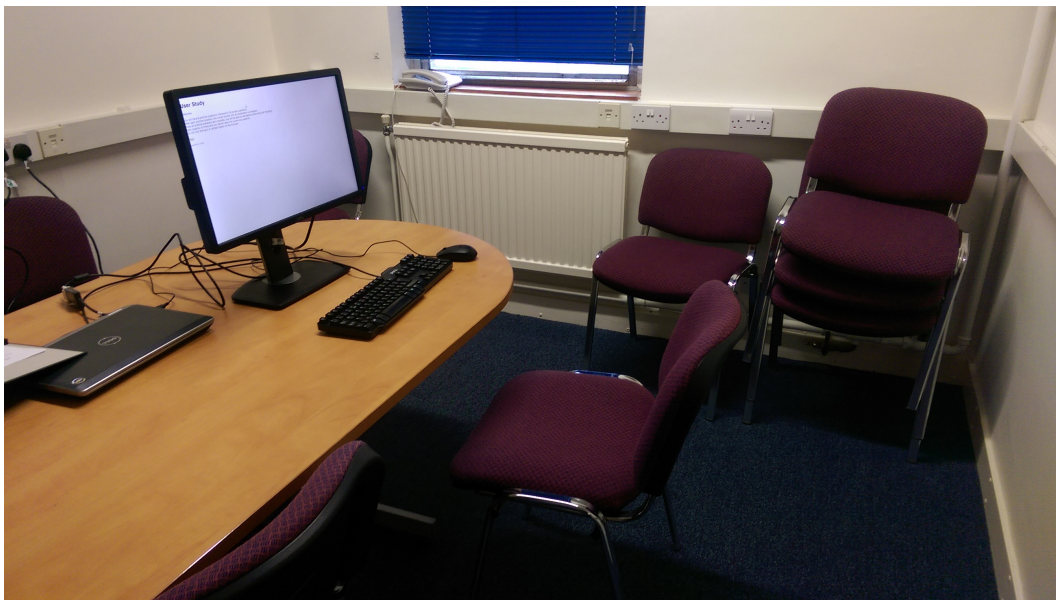


Figure 5.2: Study Room

5.2.2 Task types

Two of the task types were developed upon some existing standard tasks for studies of this type [91]. Task Type 1 was a general question about graph connectivity that is also common in the field, whereas Task Type 4 was more specific to patterns. The questions for the 4 task types were as follows:

1. "How many people directly connected to *Name* (shown in blue)?"
 - One node with 3 or more connections was picked at random.
2. "What is the minimum number of people that must be removed in order to disconnect *Name 1* & *Name 2* (both shown in blue) so that there is no route between them?"
 - One node was picked at random from the graph and chosen for use as *Name 1*. Every node that has a minimum separation of 3 from this node was then identified and one of these was picked at random to be *Name 2*. If there were no possible choices for *Name 2*, then another node to be *Name 1* was chosen. This question was taken from [91].
3. "How many other people are there in the shortest route between *Name 1* & *Name 2* (both shown in blue)?"
 - One node was picked at random from the graph and chosen for use as *Name 1*. Every node that has a minimum separation of 3 from this node was then identified and one of these was picked at random to be *Name 2*. If there were no possible choices for *Name 2*, then another node to be *Name 1* was chosen. This question was taken from [91].
4. "How many triangles are there in the diagram? (A triangle is where three people are connected to each other)"

- There was always at least one triangle in the diagram. This question was chosen to discover if patterns were easier to identify in one layout over another. Triangles were chosen as they have a simple to understand definition compared to the other pattern types.

5.2.3 Survey and Preference Information

Once the main section of the study was complete, participants were asked to complete a short questionnaire. This collected details including age, gender, education level, as well as qualitative data such as previous experience and any additional comments. Once complete the participant was then asked for their preferences. 3 pairs of graphs were shown (1 drawn with the force method, the other with the pattern method) and the participant was asked to select which one they preferred (given as an A or B choice)..

5.2.4 Participant Information

From the 40 participants, 29 (72.5%) were female and 11 (27.5%) were male. All were University of Kent students, with 27 (67.5%) saying they had post-18 education and 13 (32.5%) a degree. The participants were of a similar age range, between 18 and 24, with the mean being 20.2. Participants were given £7 for their time. They were aware they were participating in a study in the School of Computing, but not of the topic nor any other details.

5.2.5 Data Sources

For this, data was obtain from a three sources, Oscar nominees (See Section 4.2.1), Formula One team mates (See Section 4.2.2) and character analysis from classic novels (See Section 4.2.3). Although real data is being used, the labels will be

changed to those of a consistent length to avoid adding any further factors to the experiment [94, 102]. The labels were chosen from a list of 1000 most popular US baby names [3], with an attempt to have an equal number of traditionally male and female names. Consideration was also made to avoid homophonous or similar names, for example, neither Khloë and Chloë, nor Larry and Barry, would appear on the same example.

72 items of data were generated (38 items from the Oscars, 19 Formula One relations and 15 character relations in books) from which 12 were chosen. All the potential datasets were ordered by number of items (the total number of nodes and edges) and from this the estimates for small (40-45 items), medium (60-65 items) and large (80-85 items) graphs were chosen. Some datasets had more items than this, so some nodes (and therefore edges) were removed. In order to remove any element of bias, nodes were sorted according to the following categories and removal continued until the example was in the required range of items:

1. Node contained in the fewest number of patterns
2. Node with the fewest connections

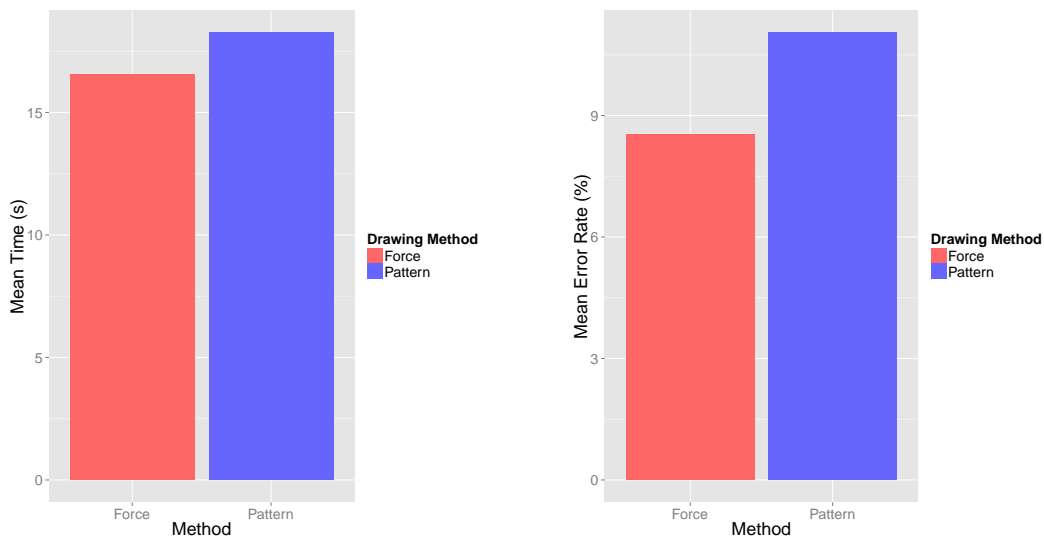
An investigator then selected the 12 datasets needed based on these groupings. Only a meaningless identifier and the number of items were made available - the chooser did not know the type of data, the name of the example, nor how the example drew with either system. This was a deliberate choice to avoid any potential bias towards either system or dataset. Four datasets surrounding the chosen medium-sized datasets were used for the final four practice questions - the first four being trivial examples.

5.3 Results

Table 5.1 details the mean, standard deviation and p -value for the overall time and error rate for each method. As can be seen, the mean time for force examples is lower than for pattern examples. This result is significant as the p -value is lower than 0.05. However, there is no significance for the error rate, although pattern examples have a higher error rate.

Table 5.1: Statistical Analysis of Force vs Pattern

		Mean	SD	p -value
Time (ms)	Pattern	18 249	11 713	3.27×10^{-5}
	Force	16 535	13 950	
Error Rate (%)	Pattern	11.04	0.27	0.149
	Force	8.54	0.3	



(a) Mean Time for Method

(b) Mean Error Rate for Method

Figure 5.3: Data grouped by Method

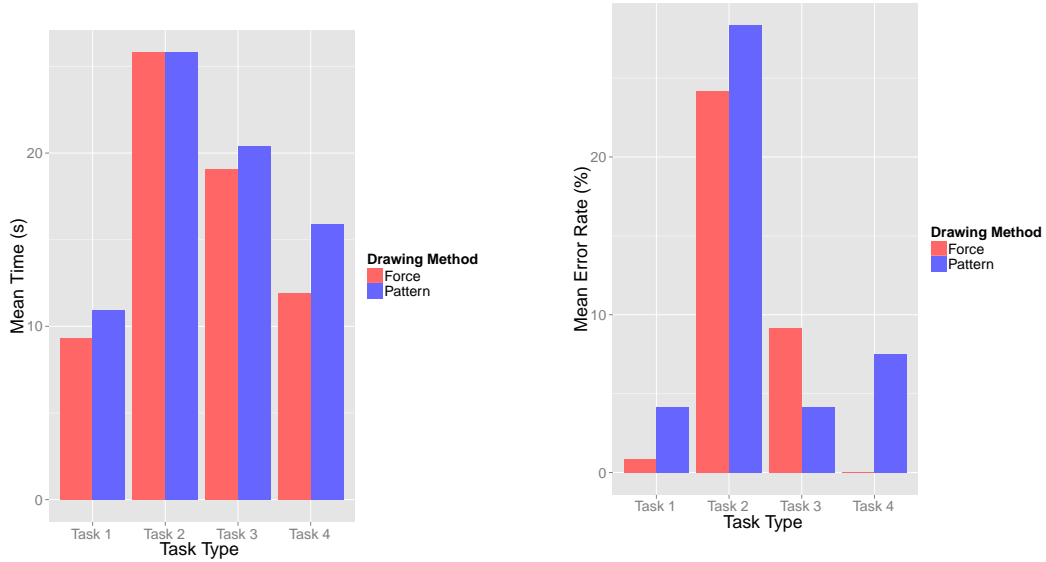
To analyse the results, Friedman two-way analysis was used as the data was not normally distributed. The analysis produces a χ^2 value for task type vs method and size vs method. The results are shown in Table 5.2. For neither pairing are the calculated χ^2 values larger than the critical χ^2 values which suggests there is no significant difference in performance. This is further confirmed by p -values being larger than 0.05.

Table 5.2: χ^2 values from Friedman Analysis for Time and Accuracy

	Critical χ^2 Value	Degrees of Freedom	Calculated Value	p -value
Task Type vs Method (Time)	7.8	3 ($p = 0.05$)	6	0.116
Task Type vs Method (Accuracy)	7.8	3 ($p = 0.05$)	3.9	0.111
Size vs Method (Time)	5.99	2 ($p = 0.05$)	4	0.135
Size vs Method (Accuracy)	5.99	2 ($p = 0.05$)	3	0.223

Figure 5.3 shows the overall mean time taken and error rates for both methods (force method in red, pattern method in blue). Figure 5.4 shows the mean time taken and error rates for both methods (force method in red, pattern method in blue) when grouped by task type. Figure 5.5 shows the mean time taken and error rates for both methods (force method in red, pattern method in blue) when grouped by graph size. Figure 5.6 shows the time taken split by method, task type and size (i.e. by question) in box plot form. This will show any questions that perform better or worse than others. One large outlier for the Large Force

Task Type 2 has been removed from the display of Figure 5.6, as this improves the clarity of the scale for all the sub-charts.



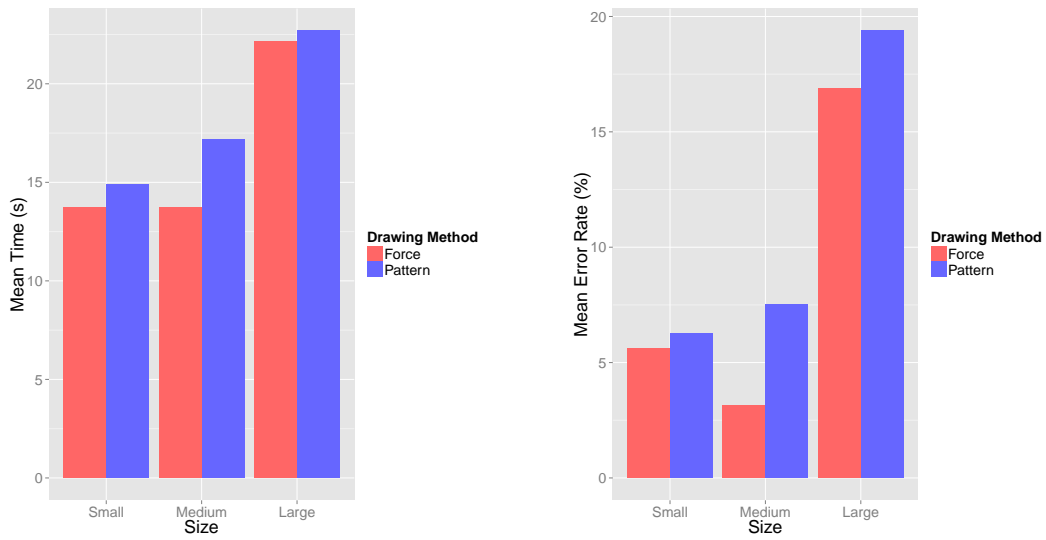
(a) Mean Time for Task Type vs Method

(b) Mean Error Rate for Task Type vs Method

Figure 5.4: Data grouped by Task

Table 5.3: *p*-values for Force vs Pattern when grouped by Task Type

<i>p</i> -values	Task Type 1	Task Type 2	Task Type 3	Task Type 4
Time	1×10^{-5}	0.027	0.02	0.023
Error Rate	0.71	0.005	0.72	No Errors



(a) Mean Time for Size vs Method

(b) Mean Error Rate for Size vs Method

Figure 5.5: Data grouped by Size

Table 5.4: p -values for Force vs Pattern when grouped by Size

p -values	Small	Medium	Large
Time	0.65	0.0005	0.001
Error Rate	0.18	0.64	0.33

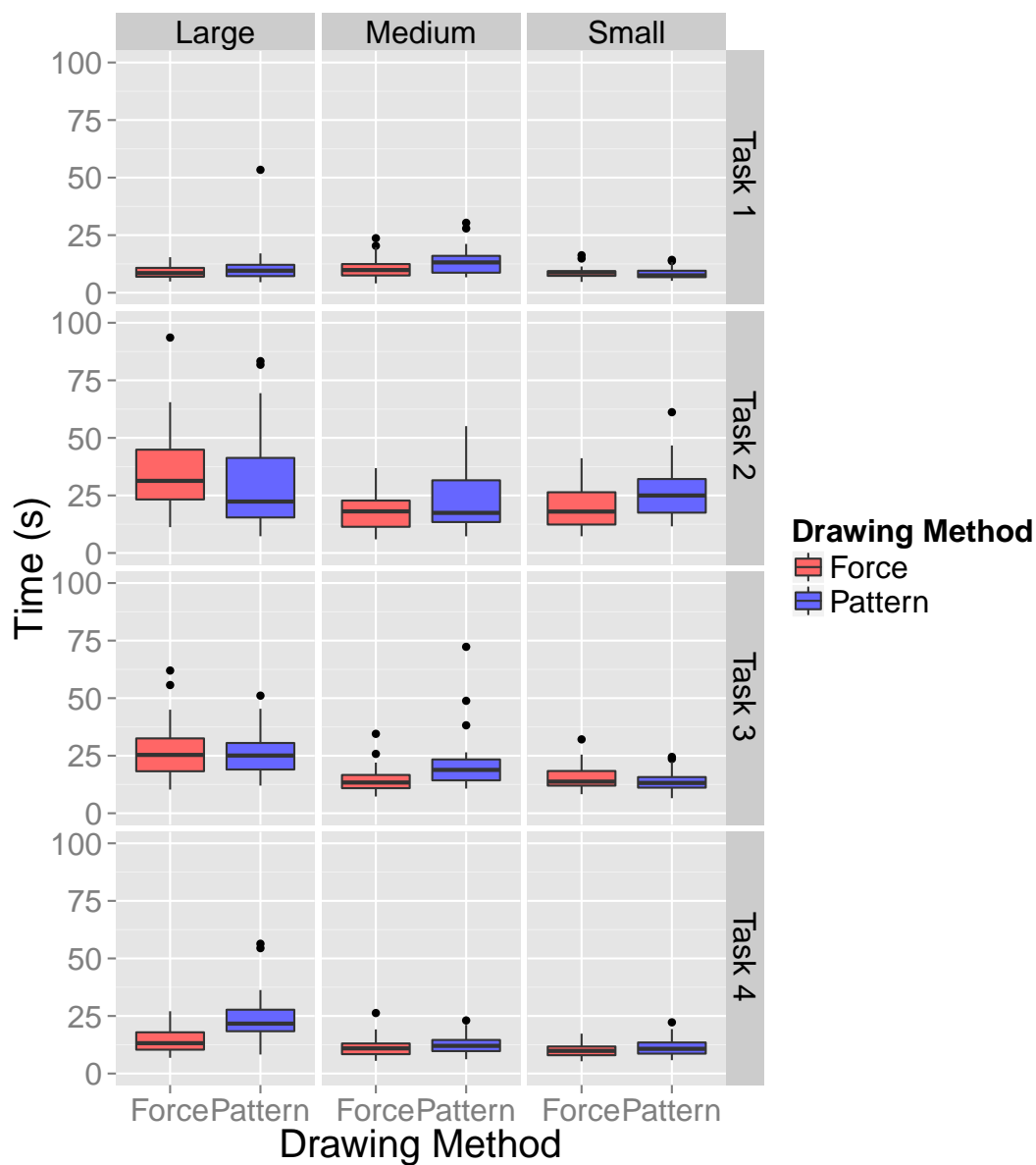


Figure 5.6: Mean Time for Method vs Size vs Time. Note that one large outlier for the Large Force Task Type 2 has been removed from the display to improve the clarity of the scale for all the sub-charts

64% of the preferences were in favour of the force directed layout (with a 0.36 standard deviation) and with a significant p -value of 2.2×10^{-16} . Most

participants had no previous experience of graphs, although a few said they had seen similar graphs “in maths A-Level”², in “chemistry” and “on a map”. One suggested the graphs were “similar to [the] non-verbal reasoning [questions] in [the] 11+ exam”³. The vast majority offered no additional comments, although one remarked the questions were “challenging but good” while another thought the “really webbed ones [were] hard to trace”. The full results of the study, including questions and anonymised user data is in Appendix B and available in CSV format online⁴.

5.4 Analysis

The hypothesis can be rejected due to significance in the time analysis. Our study shows that, with this data and questions, performance with graphs drawn with a force method is significantly quicker than performance with those drawn using a pattern based method. However, the effect size is not large, with the pattern based questions taking on average 1.7 seconds longer to complete than force based questions. There was no significance when comparing the error rates. The time may well imply that the force based method is a better layout technique than the pattern based technique. However, there are some factors to consider. The pattern method is in initial stages of development, and more development may see better performance. Further work in this area is given in Section 6.2.

When performing analysis on time taken or error rate compared to task type & method there is a lack of significance from the large p -value. This could be

²Exams taken at the age of 18 in some parts of the UK and a few other countries

³An exam taken in some parts of the UK by pupils aged 10 or 11 to decide whether they are eligible for a grammar school (i.e. selective based on ability)

⁴<http://www.eulerdiagrams.com/pattern>

due to there being a floor effect for error in all Task Types, except 2, as these all had error rates of less than 10% (and no errors for Force drawn Task Type 4 examples). There is also a floor effect in the time taken for Task 1, with the mean time being around 10 seconds. There is also some issues regarding Task Type 4, in that the choice of pattern used (a triangle) is also typically well drawn by a force directed layout. This means that both layout types draw a triangle in a similar manner, so potentially making any result for this question similar. However, using a different pattern for these questions is problematic as the definitions of the other pattern types are fairly complicated and it would be difficult to ensure that every participant fully understood them. This level of understanding is perhaps also an issue for Task Type 2, although discussing and working through the practice questions with the participant mitigate against this: it should be noted that the error rate for Task Type 2 is substantially reduced from the pilot study.

There was also no significance between time taken or error rate and size & method. However, there is significance between each task type and the time taken, but not for accuracy. There is also significance for time taken on medium and large graphs. It is perhaps unsurprising that the time taken and error rate increases as the size of the graph increases.

5.5 Threats to Validity

Threats to validity are those effects or decisions that may affect the outcome. Purchase [94] describes validity as “the extent to which the experiment correctly addresses the specified research question. Has the experiment been conducted in a manner that allows appropriate conclusions to be made?”.

5.5.1 Internal Validity

Internal validity relates to the experimental design. Purchase [94] defines this as “Has [the experiment] been designed appropriately with respect to the randomisation, controls, data collection methods, and experimental process?”.

Learning effect Participants may improve their performance throughout the study as they learn the techniques and question types. To mitigate this, questions were asked in a semi-random order with practice questions as to reduce any potential learning effect. Questions were selected at random for each participant, with the exception that questions of the same task type and size but different methods must not appear within 6 questions of each other. Participants were given instructions during the practice period to reinforce their understanding.

Carry-over effect Participants may remember questions or data if that particular appeared previously in the study. Each data set was used twice - drawn using a force based method or a pattern method and each of these examples also shared the same labels. However, these two examples would never appear within 6 questions of each other and had a different layout. None of the labels are reused in other questions (except in the practice section). This reduces the risk of a carry-over effect.

5.5.2 Construct Validity

Construct Validity is defined as “the degree to which a test measures what it claims, or purports, to be measuring” [21]. This ensures that the measures used in the study are relevant in answering the research question.

Time Collecting the time taken to complete a task is a valid measure of performance [94]. The time taken is calculated automatically. While it does not take into account the user's reading speed, all labels were 5 characters long, did not have similar written names (e.g. Lenny and Jenny). Labels were also placed in unambiguous positions where possible (names used in the question were always in clear positions). Nodes which were relevant in the question were highlighted in blue, to aid the discovery of these nodes and reduce any effect this may take.

Error Rate Error Rate is also a valid measure of performance [94]. Users had a choice of four answers, with the answers being consistent for each task type (Task Type 3 had one question with a different set of answers). All answers were numeric and in numerical order. The user had to select an answer and click "Next", reducing the chance of them selecting an incorrect answer by mistake.

Data Sources The data used for the questions were taken from 3 real-world datasets (see Section 4.2). These all consisted of social network style data and were only chosen based on their number of nodes and edges. The final 4 practice questions were also chosen from these data sources. The data sources selected were of three sizes, and all of these were of sufficient size to be challenging enough to answer.

5.5.3 External Validity

External validity is concerned with the generalisation of the results. Purchase [94] defines this as "Would these results hold for other participants, or to other experimental objects and tasks?". This ensures that the study could be repeated on other participants, with modified tasks, or a different selection of datasets.

In this study, there were 4 task types, each offering a different cognitive task. One was specifically relating to the identification of patterns. There were 3 sizes of graphs used, which offered a range of complexity. There were 40 participants, none selected by the investigators, and all of whom completed the study.

5.6 Summary

A formal empirical study was performed in order to answer the research question:

- “Is a pattern based layout more effective than a force based layout?”

Two ways of calculating the understanding are by measuring the time taken to complete certain tasks and the accuracy of the completion of these tasks. Forty participants completed four task types (see Section 5.2.2) on 12 real world data examples drawn by both a force directed method and the pattern based system. Participants were presented with an instruction sheet and practice questions before performing the study, and a preference sheet once completed.

There was a significant result for the time taken to complete a task in favour of the force based system (see Section 5.3). There was no such statistical significance for the error rate for completing all tasks. This result in favour of the force based system could be due to the questions asked (which may not have provided enough differentiation between the two systems), or the relative maturity of force based drawing algorithms (see Section 5.4).

Chapter 6

Conclusion and Further Work

This chapter provides a summary of the body of research and the contributions made. A number of potential areas of further research are also discussed.

6.1 Contributions

This thesis has detailed the creation of a new method for drawing graphs by identifying and ordering patterns and then drawing these in a consistent manner. There are a number of patterns used throughout this work that are defined in Section 3.2: Circles, Cliques, Stars, Paths, and Triangles. These patterns are then identified in the starting graph (see Section 3.3). Once identified, the patterns are ordered based on a number of criteria (see Section 3.4): type of pattern, size of pattern and type of connection with the already drawn patterns (such as One Node Shared, Two Nodes Shared, One Edge Shared, and so on). Once ordered, each pattern is then drawn using a particular method dependent on its connection type and pattern type (see Section 3.5). Once all patterns have been drawn, paths may be modified (see Section 3.5.8) and any remaining nodes are drawn (see Section 3.5.9).

To evaluate the effectiveness of this method, it was compared to a force-directed method on a number of real world examples (see Chapter 4). Data obtained from connections between Academy Award nominees (see Section 4.2.1), Formula One teammates (see Section 4.2.2), and character analysis in classic novels (see Section 4.2.3). Numerous examples of these data sets were compared for both visual curiosities (see Section 4.3) and through a number of metrics (see Section 4.4), including node-node occlusion, node-edge occlusion and edge crossings. Metrics relating to patterns (e.g. number of patterns drawn or discarded, types of patterns, and so on) were also detailed and compared.

An empirical study was also performed using the real world datasets (see Chapter 5), in order to answer the research question: “Is a pattern based layout more effective than a force based layout?”. Forty participants were asked questions using four tasks, on small, medium and large examples of graphs drawn in using both a force directed mechanism and this pattern based system (see Section 5.2). Participants performed questions based using the force directed method examples slightly faster than those drawn with this pattern based system (see Sections 5.3 and 5.4). There was no significance with the comparison between the two methods when investigating error rates. There were also no statistically significant findings when the results were broken down into task type versus method, nor size versus method.

6.2 Further Work

There are many areas which could be researched and developed to extend and improve this method, or tailor it to a particular domain. As the method has been designed to draw undirected general graphs, further modifications could be made to generate layouts for directed, planar or orthogonal graphs.

6.2.1 Increased Number of Defined Patterns

It is possible to introduce further pattern types. For example, a clique with one or two missing edges (i.e. a partial clique) may be a pattern to identify and draw in a similar manner to the existing clique method. Such patterns would require justification in the existing literature for their inclusion and would need a sensible ranking in the drawing order. Implementing patterns for a particular domain (see Section 6.2.3) or allowing the user to create custom patterns and layout is also a possible addition (see Section 6.2.6).

6.2.2 Increased Number of Connection Types

It could be possible to add in further connection types. For example, patterns sharing 3 nodes or edges could have individual layout mechanisms. Such strong sharing or connections would, however, result in the ideal layouts of patterns having large amounts of distortion. This would have the effect of reducing the benefits gained from drawing patterns in a consistent manner.

However, other, more specific connection types could be added, for example a star which shares 3 nodes, but not the centre. This would enable more complex connection types to be drawn, without necessarily encompassing all the remaining patterns. Such connection types could be specific to a particular domain or user.

6.2.3 Tailoring to a Particular Domain

It is also possible to tailor the drawing method towards a particular domain. For example, a skeletal formula is often used in chemistry to represent bonding and geometry of molecules, as shown in Figure 6.1. While the drawing method already emphasises cliques, modifications could be made to ensure that edges

with certain properties (e.g. hydrogen bonds) are represented with the correct symbols (in this case, a dotted line). Such modifications could allow the automatic drawing of skeletal formulae using this pattern based technique and comparison to an existing method, such as the one developed by Fricker et al. [47].

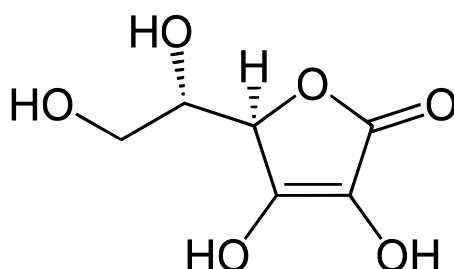


Figure 6.1: Skeletal Formula of vitamin C (ascorbic acid)

Another possible domain is related to the Predicative Toxicology Evaluation Challenge [111] where Borgelt and Berthold developed an algorithm to identify fragments within a set of molecules [18]. Such an algorithm could be incorporated within this pattern based method to layout the resulting graph in such a way to highlight certain fragments and draw these in a consistent manner.

6.2.4 Multi-level approach

A possible area of further work is to introduce a multi-level approach in a similar manner to Archambault et al [5, 6] and Hendrickson and Leland [55]. Patterns would be identified within the graph and reduced to a single node. This process would continue until at most one pattern was found, with each level being drawn in the reverse order of discovery. For example, an original graph such as Figure 6.2a could have each clique condensed into a single node (e.g. clique A becomes node A) resulting in a graph such as Figure 6.2b. This circle would then be drawn and the level complete. The next level consisting of the

four cliques would then be drawn, resulting in a layout similar to Figure 6.2a. This would create a hierarchical approach that may improve the final layout of the graph, but increase the implementation complexity of the method.

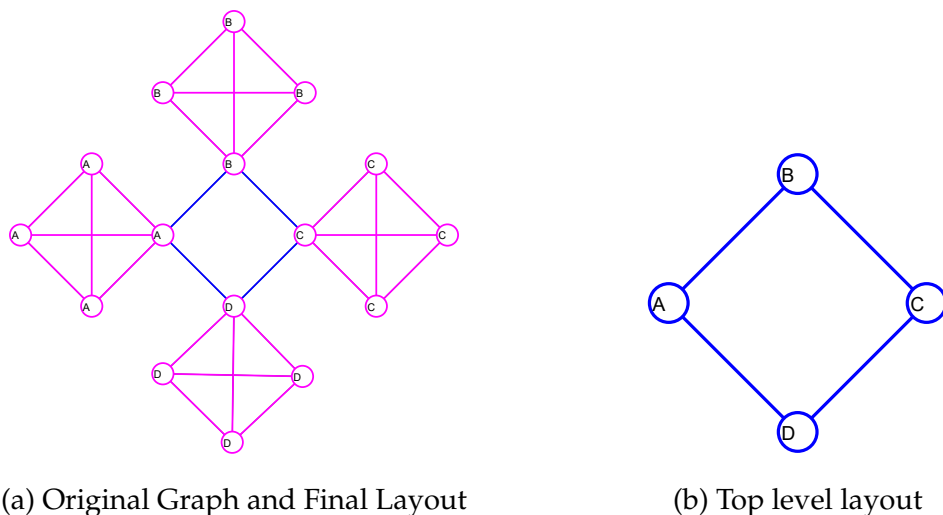


Figure 6.2: Example of a potential multi-level approach

6.2.5 Flexible Drawing Order

A piece of further work that could be implemented is to create a flexible drawing order. Currently, the drawing order is fixed and the order is decided upon the complexity of patterns and their ability to have a flexible layout. However, that could be changed in two ways.

Firstly, a user may want to emphasise a particular pattern, e.g. stars. Therefore, the user may wish that stars are placed higher in the drawing order, and thus are drawn earlier and have their ideal layouts less distorted. A user may also wish that their custom pattern (see Section 6.2.6) appears high within the drawing order.

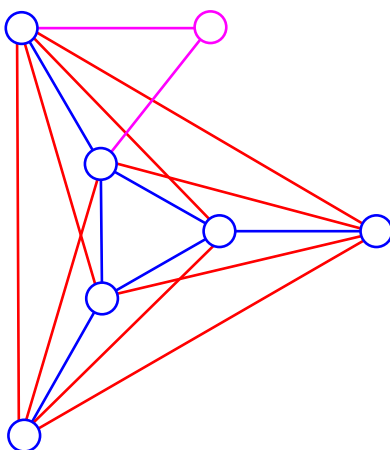
The second concept for a flexible layout is a dynamic drawing order. Based upon a number of heuristics, the algorithm could change the drawing order

during execution. This could be when the graph is drawn, as the system could naïvely try every order and compare occlusion, edge crossing and symmetry scores. Instead, it may identify areas of occlusion in the final layout and change the order based on some knowledge of the problematic area. For example, a star connected to a circle may have spokes that occlude other nodes. The system may then decide to draw stars before circles, in an attempt to remove such occlusion. The drawing method may also abandon particularly bad layouts while in the process of drawing, if it identifies significant improvements may be possible with a different drawing order. However, it would need to ensure that such a bad move will not ultimately end up with a better layout and that the improvement is significant enough to start again.

6.2.6 Custom Patterns and Layouts

One potential piece of further work is to allow a user to create and define a custom pattern. An ideal layout would need to be specified, along with any allowed, or disallowed connection types. Using the example in Figure 6.3a, a user may create the ideal layout for a new pattern, in this case a triangle with one edge radiating from each corner. They may then specify that the blue colour represents nodes and edges that must exist, with red being edges that must not exist (similar to how star spokes may not connect) and pink representing nodes or edges of which the user has no opinion (as are any nodes or edges not explicitly drawn). The user would also then specify what connection types may exist (say, 1 edge sharing but not 2 edge sharing, see Figure 6.3b), with designs for the modified layout if needed. The user would then also specify where in the drawing order such a pattern should appear (see Figure 6.3c).

Using such a design, the system would have to identify all of these patterns



(a) Custom pattern definition: a triangle with nodes radiating from each corner

Share 1 Node	Yes
Share 2 Nodes	No
Share 1 Edge	Yes
Share 2 Edges	No
Connected by Edges	Yes
No connection	Yes

(b) Custom pattern connection types

Circle	1
Clique	2
Star	3
Triangle-Star	-
Path	4
Triangle	5

(c) Custom pattern drawing order

Figure 6.3: Suggestion for the definition of a user defined pattern

which exist in the graph. This would be a big challenge. Currently the identification methods are very specific to the pattern definitions - with no such definitions existing for the custom layouts, nor a requirement from the user to implement any algorithms, another approach must be taken. Subgraph isomorphism could be used to identify all the custom subgraphs that exist within the main graph and use them in the drawing method. However, subgraph isomorphism is an NP-complete problem, and therefore approximate subgraph isomorphism would be required.

There are also issues surrounding the judgement of the user, regarding the pattern's relevance, definition, ideal layout, and order. The user may well select suboptimal choices for each of these and this would have an adverse effect on

the final layout. However, if techniques such as those described in Section 6.2.5 were implemented, this would mitigate such poor choices.

6.2.7 Hybrid System

One piece of further work could be to combine this pattern based system with a force directed layout. For example, when drawing a pattern which connected to the drawn set by one of more edges (or has no connection), the current search based strategy could be replaced by a force directed model. In this case, attractive forces (shown in Figure 6.4 in green) could exist between the nodes in pattern about to be drawn and the nodes in the drawn set at the opposite end of the connecting edges. Repulsive forces (shown in Figure 6.4 in red) could exist between the nodes in pattern about to be drawn and all (or those within a certain distance) nodes in the drawn set. Each iteration would have to ensure that the pattern about to be drawn maintained its correct layout.

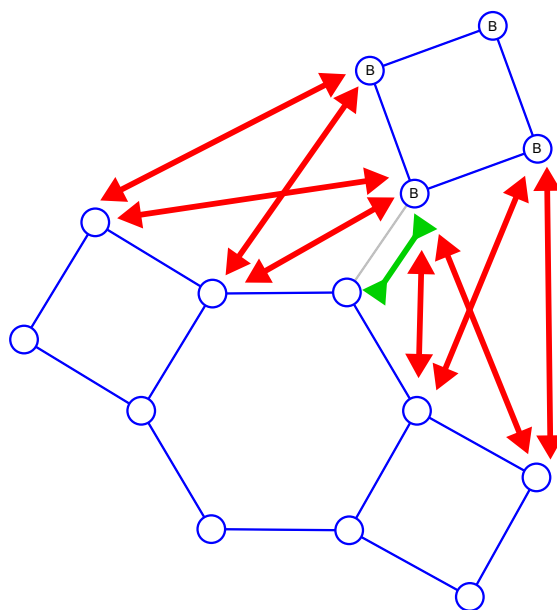
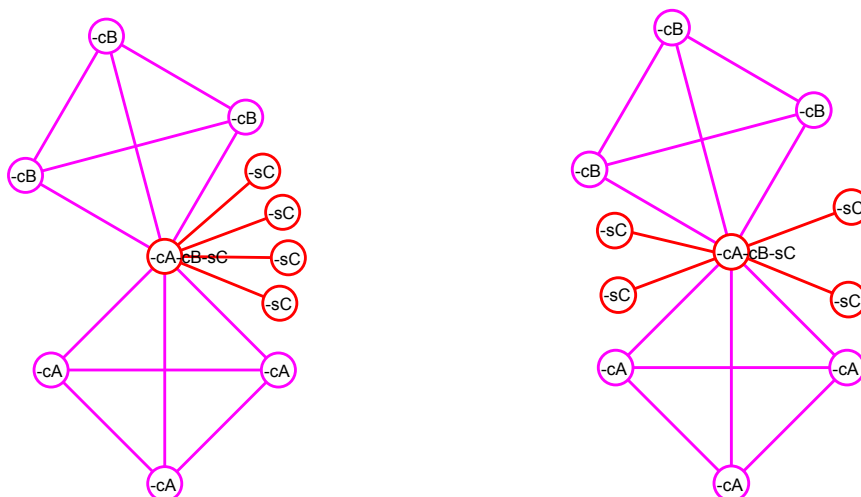


Figure 6.4: Potential Force-Directed Placement of Patterns

Other uses could be in placing loose nodes, or the spokes of stars using a force directed system. While this may weaken the regular layouts of stars, it could allow better placement in that spokes could be spread amongst open spaces, rather than all being clustered in one location. For example, Figure 6.5a shows a star sharing a centre node as drawn by the current pattern based system. Under a hybrid system, the nodes could be more evenly distributed into the available space (see Figure 6.5b), even perhaps having different edge lengths if required. However, the force directed system does not optimise to avoid edge crossings, so extra forces or penalty scores may need to be added.



(a) Current Pattern Based Layout

(b) Possible Force-Directed Layout

Figure 6.5: Example of a potential multi-level approach

Bibliography

- [1] GD Data. Accessed: 2017-05-12.
<http://www.graphdrawing.org/data.html>
- [2] Project Gutenberg. Accessed: 2015-06-19.
<https://www.gutenberg.org/>
- [3] Baby Names 1000 - Five Letters, 2016. Accessed: 2015-11-04.
<http://www.babynames1000.com/five-letter/>
- [4] Steven Seagal - Yahoo Search, 2016. Accessed: 2016-09-02.
<https://uk.search.yahoo.com/search?p=steven+seagal>
- [5] Daniel Archambault, Tamara Munzner, and David Auber. TopoLayout: Multilevel graph layout by topological features. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):305–316, 2007. ISSN 10772626.
- [6] Daniel Archambault, Tamara Munzner, and David Auber. GrouseFlocks: Steerable exploration of graph hierarchy space. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):900–913, 2008. ISSN 10772626.
- [7] Alessio Arleo, Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. A Distributed Multilevel Force-directed Algorithm. In *Proceedings of the 24th International Symposium on Graph Drawing and Network Visualization (GD 2016)*, 2016.

- [8] Robert Baker, Peter Rodgers, Simon Thompson, and Huiqing Li. Multi-level Visualization of Concurrent and Distributed Computation in Erlang. In *The 19th International Conference on Distributed Multimedia Systems (DMS 2013)*, pages 156–161, 2013.
- [9] Josh Barnes and Piet Hut. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, 324(6096):446–449, 1986. ISSN 0028-0836.
- [10] N. P. Barradas, C. Jeynes, and R. P. Webb. Simulated annealing analysis of Rutherford backscattering data. *Applied Physics Letters*, 71(2):291, 1997. ISSN 00036951.
- [11] Harry G Barrow, A P Ambler, and Rodney Matineau Burstall. Some techniques for recognizing structures in pictures. *Frontiers of Pattern Recognition*, pages 1–29, 1972.
- [12] Skye Bender-deMoll and Daniel McFarland. The art and science of dynamic network visualization. *Journal of Social Structure*, 7(2):1–38, 2006. ISBN 1529-1227, 1529-1227.
- [13] C. Bennett, J. Ryall, L. Spalteholz, and A. Gooch. The aesthetics of graph visualization. *Proceedings of the 2007 Computational Aesthetics in Graphics, Visualization, and Imaging*, pages 1–8, 2007. ISBN 978-3-905673-43-2.
- [14] Irving Biederman. Recognition by components: A theory of human image understanding. *Psychological Review*, 94(2):115–117, 1987. ISBN 0033-295X.
- [15] Therese Biedl, J Marks, K Ryall, and S Whitesides. Graph Multidrawing: Finding Nice Drawings Without Defining Nice. *Graph Drawing*, 1547:347–355, 1998. ISBN 3-540-65473-9.

- [16] Karl-Friedrich Böhringer and Frances Newbery Paulisch. Using Constraints To Achieve Stability In Automatic Layout. In *CHI '90 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 43–51, 1990.
- [17] Vincenzo Bonnici, Rosalba Giugno, Alfredo Pulvirenti, Dennis Shasha, and Alfredo Ferro. A subgraph isomorphism algorithm and its application to biochemical data. *BMC bioinformatics*, 14 Suppl 7(Suppl 7):S13, apr 2013. ISSN 1471-2105.
- [18] Christian Borgelt and M.R. Berthold. Mining molecular fragments: finding relevant substructures of molecules. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 51–58. IEEE Comput. Soc, 2002. ISBN 0-7695-1754-4.
- [19] Franz J Brandenburg, Michael Himsolt, and Christoph Rohrer. An experimental comparison of force-directed and randomized graph drawing algorithms. In *International Symposium on Graph Drawing*, pages 76–87. Springer, 1996.
- [20] J Breithaupt. *Essential AS Physics for OCR*. Nelson Thornes Limited, 2004. ISBN 9780748785070.
- [21] James Dean Brown. *Testing in language programs: a comprehensive guide to English language assessment*. Prentice Hall Regents, 2005. ISBN 9780072948363.
- [22] Horst Bunke. Attributed Programmed Graph Grammars and Their Application to Schematic Diagram Interpretation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(6):574–582, 1982.

- [23] Tanya Cashorali. Music Roamer, 2008. Accessed: 2015-10-14.
<http://musicroamer.com/{#}/search?artist=franksinatra{&}nodes=1{&}nodes=5>
- [24] Gary Chartrand. *Introductory Graph Theory*. Courier Corporation, 1977. ISBN 0486247759.
- [25] Chaomei Chen. *Information Visualization: Beyond the Horizon*. Springer Science & Business Media, 2006. ISBN 1846285798.
- [26] Uroš Čibej and Jurij Mihelič. Search strategies for subgraph isomorphism algorithms. In *International Conference on Applied Algorithms*, pages 77–88, 2014. ISBN 9783319041254.
- [27] Douglas H. Clements and Julie Sarama. Young Children’s Ideas about Geometric Shapes. *Teaching Children Mathematics*, 6(8):482–88, 1999. ISSN ISSN-1073-5836.
- [28] Michael K. Coleman and D. Stott Parker. Aesthetics-based Graph Layout for Human Consumption. *Software: Practice and Experience*, 26(12):1415–1438, 1996. ISBN 1097-024X.
- [29] Thomas F Coleman and Jorge J Moré. Estimation of sparse Jacobian matrices and graph coloring blems. *SIAM journal on Numerical Analysis*, 20(1):187–209, 1983.
- [30] Luigi P Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1367–72, 2004. ISSN 0162-8828.

- [31] Isabel F Cruz and Roberto Tamassia. Graph Drawing Tutorial, 2003. ISBN 0012200603. Accessed: 2015-04-03.
<https://cs.brown.edu/courses/csci2520/misc/slides/gd-constraints.pdf>
- [32] Ron Davidson and David Harel. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics*, 15(4):301–331, oct 1996. ISSN 07300301.
- [33] Fred DePiero and David Krout. An algorithm using length-r paths to approximate subgraph isomorphism. *Pattern recognition letters*, 24(1):33–46, 2003.
- [34] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry*, 4(5):235–282, oct 1994. ISSN 09257721.
- [35] Josep Díaz, Jordi Petit, and Maria Serna. A survey of graph layout problems. *ACM Computing Surveys*, 34(3):313–356, 2002. ISBN 0360-0300.
- [36] Cody Dunne and Ben Shneiderman. Motif Simplification: Improving Network Visualization Readability with Fan, Connector, and Clique Glyphs. In *CHI '13: Proc. 2013 International Conference on Human Factors in Computing Systems*, pages 3247–3256, 2013. ISBN 9781450318990.
- [37] Tim Dwyer, Bongshin Lee, Danyel Fisher, Kori Inkpen Quinn, Petra Isenberg, George Robertson, and Chris North. A comparison of user-generated and automatic graph layouts. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):961–968, 2009. ISSN 10772626.
- [38] Peter Eades. A Heuristic for Graph Drawing. *Congressus Numerantium*, 42:149–160, 1984.

- [39] Peter Eades, Wei Lai, Kazuo Misue, and Kozo Sugiyama. Preserving the Mental Map of a Diagram. Technical report, International Institute for Advanced Study of Social Information Science, 1991.
- [40] Peter Eades and Xuemin Lin. Spring algorithms and symmetry. *Theoretical Computer Science*, 240(2):379–405, 2000. ISSN 03043975.
- [41] Peter Eades, Xuemin Lin, and Roberto Tamassia. An Algorithm For Drawing A Hierarchical Graph. *International Journal of Computational Geometry & Applications*, 6(2):145–56, 1996.
- [42] Hartmut Ehrig, Annegret Habel, and Hans-Jörg Kreowski. Introduction to Graph Grammars with Applications to Semantic Networks. *Computers & Mathematics with Applications*, 23(6):557–572, 1992.
- [43] David Eppstein. Subgraph isomorphism in planar graphs and related problems. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, volume 95, pages 632–640, 1995.
- [44] Leonhard Euler. Solutio problematis ad geometriam situs pertinentis (The solution of a problem relating to the geometry of position). *Commentarii academiae scientiarum Petropolitanae*, 8:128–140, 1741.
- [45] Clifford J Fisk, David L Caskey, and Leslie E West. ACCEL: Automated circuit card etching layout. *Proceedings of the IEEE*, 55(11):1971–1982, 1967. ISBN 0018-9219.
- [46] Arne Frick, Andreas Ludwig, and Heiko Mehldau. A fast adaptive layout algorithm for undirected graphs. In *Graph Drawing*, pages 388–403, 1995.

- [47] Patrick C Fricker, Marcus Gastreich, and Matthias Rarey. Automated drawing of structural molecular formulas under constraints. *Journal of chemical information and computer sciences*, 44(3):1065–1078, 2004.
- [48] Thomas M J Fruchterman and Edward M Reingold. Graph Drawing by Force-directed Placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.
- [49] Pawel Gajer, Michael T Goodrich, and Stephen G Kobourov. A multi-dimensional approach to force-directed layouts of large graphs. *Computational Geometry: Theory and Applications*, 29(1):3–18, 2004. ISBN 3-540-41554-8.
- [50] Emden R Gansner, Yehuda Koren, and Stephen North. Graph Drawing by Stress Majorization. In *12th International Symposium, Graph Drawing*, pages 239–250, 2004.
- [51] H. Gibson, J. Faith, and P. Vickers. A survey of two-dimensional graph layout techniques for information visualisation. *Information Visualization*, 12(3-4):324–357, 2012. ISSN 1473-8716.
- [52] Frank Harary. *Graph Theory*. Addison-Wesley, 3rd edition, 1969.
- [53] David Harel. On the Aesthetics of Diagrams (Summary of Talk). In *MPC '98 Proceedings of the Mathematics of Program Construction*, pages 1–5, London, 1998. Springer-Verlag. ISBN 3-540-64591-8.
- [54] David Harel and Yehuda Koren. A Fast Multi-Scale Method for Drawing Large Graphs. *Journal of Graph Algorithms and Applications*, 6(3):179–202, 2002. ISBN 1581132522.

- [55] Bruce Hendrickson and Robert Leland. A Multi-Level Algorithm For Partitioning Graphs. In *Proceedings of the IEEE/ACM SC95 Conference*, pages 1–14. IEEE, 1995. ISBN 0-89791-816-9.
- [56] Ivan Herman, Guy Malançon, and M. Scott Marshall. Graph Visualization and Navigation in Information Visualization: a Survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.
- [57] M H W Hobbs and Peter Rodgers. Representing Space: A Hybrid Genetic Algorithm for Aesthetic Graph Layout. In *FEA 1998, Frontiers in Evolutionary Algorithms, Proceedings of the Fourth Joint Conference on Information Sciences*, pages 415–418, 1998.
- [58] Danny Holten and Jarke J Van Wijk. Force-Directed edge bundling for graph visualization. *Computer Graphics Forum*, 28(3):983–990, 2009. ISBN 01677055.
- [59] Yifan Hu. Efficient, High-Quality Force-Directed Graph Drawing. *Mathematica Journal*, 10(1):37–71, 2005. ISBN 1047-5974.
- [60] Yifan Hu. Algorithms for visualizing large networks. *Combinatorial Scientific Computing*, 5(3):180–186, 2011.
- [61] Jun Huan, Wei Wang, and Jan Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *Third IEEE International Conference on Data Mining, 2003. ICDM 2003*, pages 549–552, 2003.
- [62] Weidong Huang. Establishing aesthetics based on human graph reading behavior: Two eye tracking studies. *Personal and Ubiquitous Computing*, 17(1):93–105, 2013. ISBN 1617-4909.

- [63] Weidong Huang, Peter Eades, and Seok-Hee Hong. Beyond time and error: A Cognitive Approach to the Evaluation of Graph Drawings. In *Proceedings of the 2008 Conference on Beyond time and errors: Novel evaluation methods for Information Visualization - BELIV '08*, 2008. ISBN 9781605580166.
- [64] Weidong Huang, Peter Eades, and Seok-Hee Hong. Measuring effectiveness of graph visualizations: A cognitive load perspective. *Information Visualization*, 8(3):139–152, 2009. ISSN 1473-8716.
- [65] Weidong Huang, Seok-hee Hong, and Peter Eades. *Layout Effects : Comparison of Sociogram Drawing Conventions*. School of Information Technologies, University of Sydney, 2005.
- [66] Weidong Huang, Seok-hee Hong, and Peter Eades. Predicting graph reading performance: A cognitive approach. In *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation*, pages 207–216, 2006.
- [67] Chuntao Jiang, Frans Coenen, and Michele Zito. A Survey of Frequent Subgraph Mining Algorithms. *The Knowledge Engineering Review*, 28(2):75–105, 2013. ISSN 0269-8889. arXiv preprint arXiv:1201.3011.
- [68] Tomihisa Kamada and Satoru Kawai. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15, 1989.
- [69] Stephen G Kobourov. Spring Embedders and Force Directed Graph Drawing Algorithms. *CoRR*, pages 1–23, 2012.
- [70] Pierre-Yves Koenig, Faraz Zaidi, and Daniel Archambault. Interactive searching and visualization of patterns in attributed graphs. In *Proceedings of Graphics Interface 2010*, pages 113–120, 2010.

- [71] Denes König. *Theory of Finite and Infinite Graphs*. Springer Science & Business Media, 2013. ISBN 1468489712.
- [72] Vincent Lacroix, Cristina G. Fernandes, and Marie-France Sagot. Motif search in graphs: Application to metabolic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):360–368, 2006. ISBN 15455963.
- [73] Antoine Lambert, Romain Bourqui, F. Queyroi, and Romain Bourqui. Visualizing patterns in node-link diagrams. In *Proceedings of the International Conference on Information Visualisation*, pages 48–53, 2012. ISBN 9780769547718.
- [74] Seongwhan Lee and Jin H. Kim. Attributed stroke graph matching for seal imprint verification. *Pattern Recognition Letters*, 9(2):137–145, 1989. ISSN 01678655.
- [75] Olav Lenz, Frank Keul, Sebastian Bremm, Kay Hamacher, and Tatiana von Landesberger. Visual Analysis of Patterns in Multiple Amino Acid Mutation Graphs. In *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 93–102. IEEE, 2014. ISBN 978-1-4799-6227-3.
- [76] Chun-Cheng Lin and Hsu-Chun Yen. A new force-directed graph drawing method based on edgeedge repulsion. *Journal of Visual Languages & Computing*, 23(1):29–42, 2012.
- [77] Fabian Lipp, Alexander Wolff, and Johannes Zink. Faster Force-Directed Graph Drawing with the Well-Separated Pair Decomposition. *Algorithms*, 9(3):53, 2016. ISSN 1999-4893.
- [78] Jens Lischka and Holger Karl. A virtual network mapping algorithm based on subgraph isomorphism detection. In *Proceedings of the 1st*

ACM workshop on Virtualized Infrastructure Systems and Architectures - VISA '09, pages 81–88, New York, New York, USA, 2009. ACM Press. ISBN 9781605585956.

- [79] Si Wei Lu, Ying Ren, and Ching Y Suen. Hierarchical attributed graph representation and recognition of handwritten Chinese characters. *Pattern Recognition*, 24(7):617–632, 1991.
- [80] Sonja Maier and Mark Minas. A Pattern-Based Layout Algorithm for Diagram Editors. *Electronic Communications of the EASST*, 7, 2007.
- [81] David McCandless. *The Middle East - Key Players and Notable Relationships*, 2016. Accessed: 2016-01-12.
<http://www.informationisbeautiful.net/visualizations/the-middle-east-key-players-notable-relationships/>
- [82] Cathleen McGrath and Jim Blythe. Do you see what I want you to see? The effects of motion and spatial layout on viewers' perceptions of graph structure. *Journal of Social Structure*, 5(2):2, 2004. ISSN 1529-1227.
- [83] Cathleen McGrath, Jim Blythe, and David Krackhardt. The effect of spatial arrangement on judgments and errors in interpreting graphs. *Social Networks*, 19(3):223–242, 1997. ISBN 0378-8733.
- [84] Bruno T Messmer and Horst Bunke. Subgraph Isomorphism in Polynomial Time. In *Recent Developments in Computer Vision*, pages 373–382. Springer, 1996.
- [85] Bruno T Messmer and Horst Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):493–504, 1998.

- [86] Bruno T Messmer and Horst Bunke. A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recognition*, 32(12):1979–1998, 1999. ISSN 00313203.
- [87] Bruno T Messmer and Horst Bunke. Efficient subgraph isomorphism detection: a decomposition approach. *IEEE Transactions on Knowledge and Data Engineering*, 12(2):307–323, 2000. ISSN 10414347.
- [88] Ron Milo, Shai Shen-Orr, S Itzkovitz, N Kashtan, D Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science (New York, N.Y.)*, 298(5594):824–827, 2002. ISSN 00368075.
- [89] Casper Pleijsier. 2013 F1 Results & Standings, 2016. Accessed: 2016-09-02. <http://www.f1-fansite.com/f1-results/results-2013-formula-1-season/>
- [90] Casper Pleijsier. Results 2012 Formula 1 Season, 2016. Accessed: 2016-09-02. <http://www.f1-fansite.com/f1-results/results-2012-formula-1-season/>
- [91] Helen Purchase. Which Aesthetic Has the Greatest Effect on Human Understanding? In *Graph Drawing - 5th International Symposium*, pages 248–261, 1997. ISBN 978-3-540-69674-2.
- [92] Helen Purchase. Performance of Layout Algorithms: Comprehension, not Computation. *Journal of Visual Languages and Computing*, 9(6):647–657, 1998. ISSN 1045-926X.
- [93] Helen Purchase. Effective information visualisation: a study of graph drawing aesthetics and algorithms. *Interacting with Computers*, 13(2):147–162, 2000. ISBN 0953-5438.

- [94] Helen Purchase. *Experimental human-computer interaction: a practical guide with visual examples*. Cambridge University Press, 2012.
- [95] Helen Purchase, Jo-Anne Alder, and David Carrington. User Preference of Graph Layout Aesthetics: A UML Study. In *International Symposium on Graph Drawing*, pages 5–18, 2000. ISBN 978-3-540-41554-1.
- [96] Helen Purchase, David Carrington, and Jo-Anne Alder. Empirical evaluation of aesthetics-based graph layout. *Empirical Software Engineering*, 7(3):233–255, 2002. ISSN 13823256.
- [97] Helen Purchase, R F Cohen, and M James. Validating Graph Drawing Aesthetics. In *International Symposium on Graph Drawing*, pages 435–446, 1995. ISBN 3-540-60723-4.
- [98] Helen Purchase, R F Cohen, and M James. An experimental study of the basis for graph drawing algorithms. *Journal of Experimental Algorithmics*, 2(4):1–17, 1997. ISSN 10846654.
- [99] Helen Purchase, Eve Hoggan, and Carsten Görg. How important is the “Mental Map”? - An empirical investigation of a dynamic graph layout algorithm. In *International Symposium on Graph Drawing*, pages 184–195, 2006. ISBN 978-3-540-70903-9.
- [100] Helen Purchase, Matthew McGill, Linda Colpoys, and David Carrington. Graph drawing aesthetics and the comprehension of UML class diagrams: an empirical study. In *Proceedings of the 2001 Asia-Pacific symposium on Information visualisation*, pages 129–137. Australian Computer Society, Inc., 2001. ISBN 0-909925-87-9.
- [101] Helen C Purchase. Metrics for graph drawing aesthetics. *Journal of Visual Languages and Computing*, 13(5):501–516, 2002. ISBN 1045-926X.

- [102] Helen C Purchase and Amanjit Samra. Extremes are better: Investigating mental map preservation in dynamic graphs. In *International Conference on Theory and Application of Diagrams*, pages 60–73, 2008.
- [103] N. Quinn and M. Breuer. A forced directed component placement procedure for printed circuit boards. *IEEE Transactions on Circuits and Systems*, 26(6):377–388, 1979. ISSN 0098-4094.
- [104] Peter Rodgers, Gem Stapleton, Bilal Alsallakh, Luana Michallef, Robert Baker, and Simon Thompson. A Task-Based Evaluation of Combined Set and Network Visualization. *Information Sciences*, 367-368:58–79, 2016. ISBN 4401273642410.
- [105] Dennis H Rouvray and Alexandru T Balaban. Chemical applications of graph theory. *Applications of Graph Theory*, pages 177–221, 1979.
- [106] Amalia Adrian Rusu, Andrew J. Fabian, and Radu Jianu. Using the Gestalt Principle of Closure to Alleviate the Edge Crossing Problem in Graph Drawings. In *2011 15th International Conference on Information Visualisation*, pages 488–493. IEEE, 2011. ISBN 978-1-4577-0868-8.
- [107] Falk Schreiber and Henning Schwöbbermeyer. Towards motif detection in networks: Frequency concepts and flexible search. In *Proceedings of the International Workshop on Network Tools and Applications in Biology*, pages 91–102, 2004.
- [108] Falk Schreiber and Henning Schwöbbermeyer. MAVisto: A tool for the exploration of network motifs. *Bioinformatics*, 21(17):3572–3574, 2005. ISSN 13674803.

- [109] Shai Shen-Orr, Ron Milo, Shmoolik Mangan, and Uri Alon. Network motifs in the transcriptional regulation network of *Escherichia coli*. *Nature Genetics*, 31(1):64–68, 2002. ISSN 10614036.
- [110] Robert Spence. *Information visualization*. Springer, 1st edition, 2001.
- [111] A. Srinivasan, S. H. Muggleton, R. D. King, and M. J. E. Sternberg. The predictive toxicology evaluation challenge. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 4–9. Morgan Kaufmann Publishers Inc., 1997. ISBN 1-555860-480-4.
- [112] Harald Storrle. On the Impact of Layout Quality to Understanding UML Diagrams: Diagram Type and Expertise. In *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC*, pages 49–56, 2012. ISBN 9781467308502.
- [113] Kozo Sugiyama and Kazuo Misue. Graph drawing by the magnetic spring model. *Journal of Visual Languages and Computing*, 6(3):217–231, 1995.
- [114] Roberto Tamassia. Constraints in graph drawing algorithms. *Constraints*, 120:87–120, 1998.
- [115] Roberto Tamassia. *Handbook of Graph Drawing and Visualization*. CRC Press, 2013. ISBN 1420010263.
- [116] Roberto Tamassia, Giuseppe Di Battista, and C Batini. Automatic graph drawing and readability of diagrams. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):61–79, 1988.

- [117] Martyn Taylor and Peter Rodgers. Applying Graphical Design Techniques to Graph Visualisation. In *Ninth International Conference on Information Visualisation (IV'05)*, pages 651–656. IEEE, 2005. ISBN 0-7695-2397-8.
- [118] Daniel Tunkelang. Carnegie Mellon School of Computer Science. Technical report, DTIC Document, 1994. CMU-CS-94-161.
- [119] Daniel Tunkelang. *A Numerical Optimization Approach to General Graph Drawing*. PhD thesis, Carnegie Mellon University, 1999.
- [120] W T Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 8(May 1962):743–767, 1963. ISSN 0024-6115.
- [121] W T Tutte. *Graph Theory*. Cambridge University Press, 2001. ISBN 0521794897.
- [122] J R Ullmann. An Algorithm for Subgraph Isomorphism. *Journal of the ACM*, 23(1):31–42, 1976.
- [123] J Utech, J Branke, H Schmeck, and Peter Eades. An evolutionary algorithm for drawing directed graphs. In *Proceedings of the International Conference on Imaging Science, Systems and Technology*, pages 154–160, 1998.
- [124] Corinna Vehlow, Fabian Beck, and Daniel Weiskopf. The State of the Art in Visualizing Group Structures in Graphs. In *Eurographics Conference on Visualization*, 2015.
- [125] Tatiana Von Landesberger, Arjan Kuijper, Tobias Schreck, Jörn Kohlhammer, Jarke J van Wijk, Jean Fekete, Dieter W Fellner, Jarke J van Wijk, Jean Fekete, and Dieter W Fellner. Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011. ISSN 1467-8659.

- [126] C Walshaw. A Multilevel Algorithm for Force-Directed Graph Drawing. In *International Symposium on Graph Drawing*, pages 171–182. Springer, 2000.
- [127] Colin Ware. *Information visualization: perception for design*. Elsevier, 2012.
- [128] Colin Ware, Helen Purchase, Linda Colpoys, and Matthew McGill. Cognitive Measurements of Graph Aesthetics. *Information Visualization*, 1(2):103–110, 2002. ISBN 1473-8716.
- [129] J. Mark Ware, Christopher B. Jones, and Nathan Thomas. Automated map generalization with multiple operators: a simulated annealing approach. *International Journal of Geographical Information Science*, 17(8):743–769, dec 2003. ISSN 1365-8816.
- [130] Eric W. Weisstein. “Maximal Clique”. From MathWorld—A Wolfram Web Resource, 2017. Accessed: 2017-05-25.
<http://mathworld.wolfram.com/MaximalClique.html>
- [131] Sebastian Wernicke and Florian Rasche. FANMOD: A tool for fast network motif detection. *Bioinformatics*, 22(9):1152–1153, may 2006. ISSN 13674803.
- [132] C. Wetherell and A. Shannon. Tidy Drawings of Trees. *IEEE Transactions on Software Engineering*, SE-5(5):514–520, 1979. ISSN 0098-5589.
- [133] Kenny Wong and Dabo Sun. On evaluating the layout of UML diagrams for program comprehension. *Software Quality Journal*, 14(3):233–259, 2006. ISBN 0769522548.

- [134] K.P. Wong and C.C. Fung. Simulated annealing based economic dispatch algorithm. *IEE Proceedings C-Generation, Transmission and Distribution*, 140(6):509, 1993. ISSN 01437046.
- [135] Xiaoru Yuan, Limei Che, Yifan Hu, and Xin Zhang. Intelligent Graph Layout Using Many Users' Input. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2699–2708, 2012. ISSN 1077-2626.

Appendix A

Full Analysis of Generated Layouts

A.1 Size of graphs

The following charts show the number of nodes and edges in each graph.

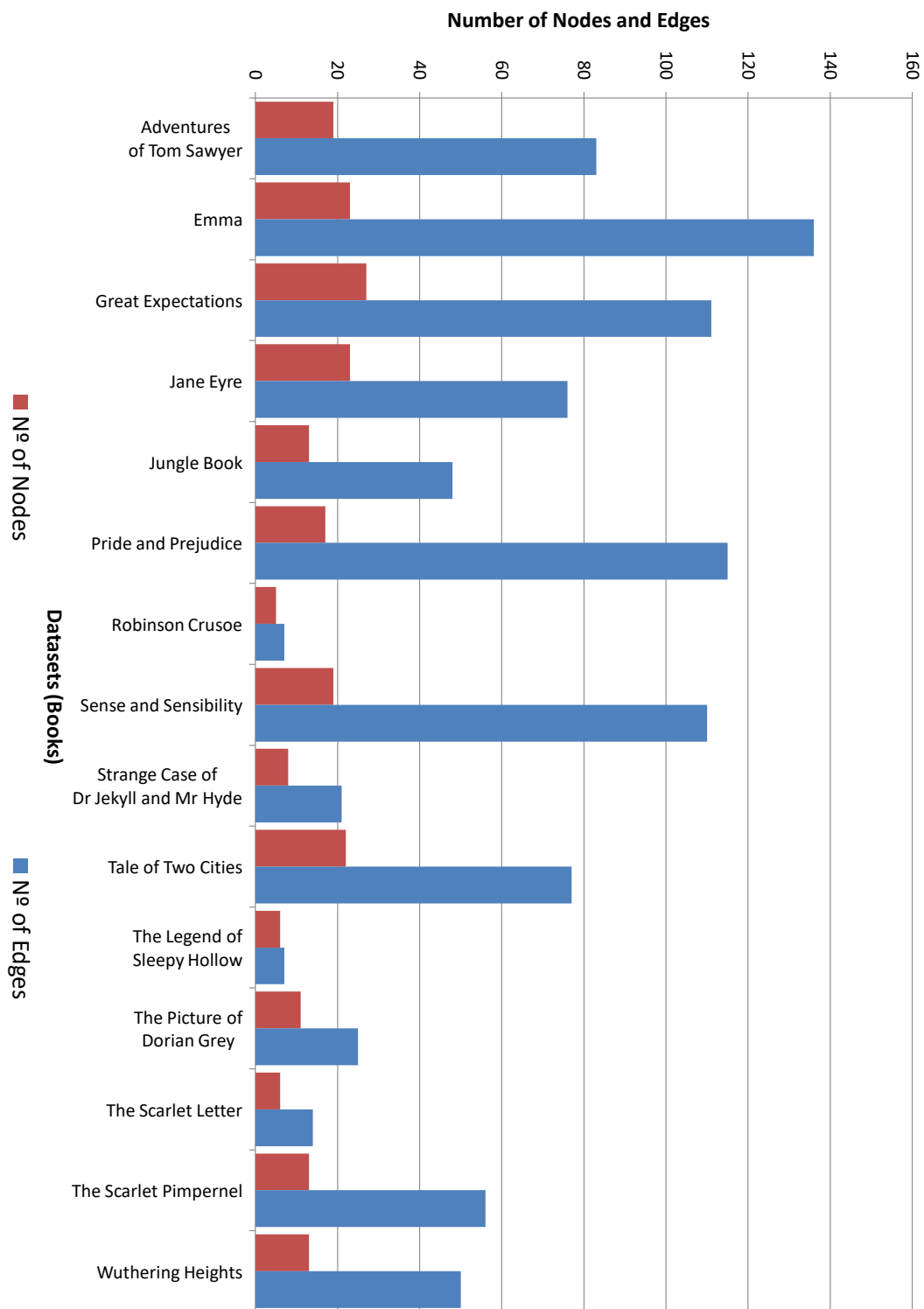


Figure A.1: Number of Nodes and Edges (Books)

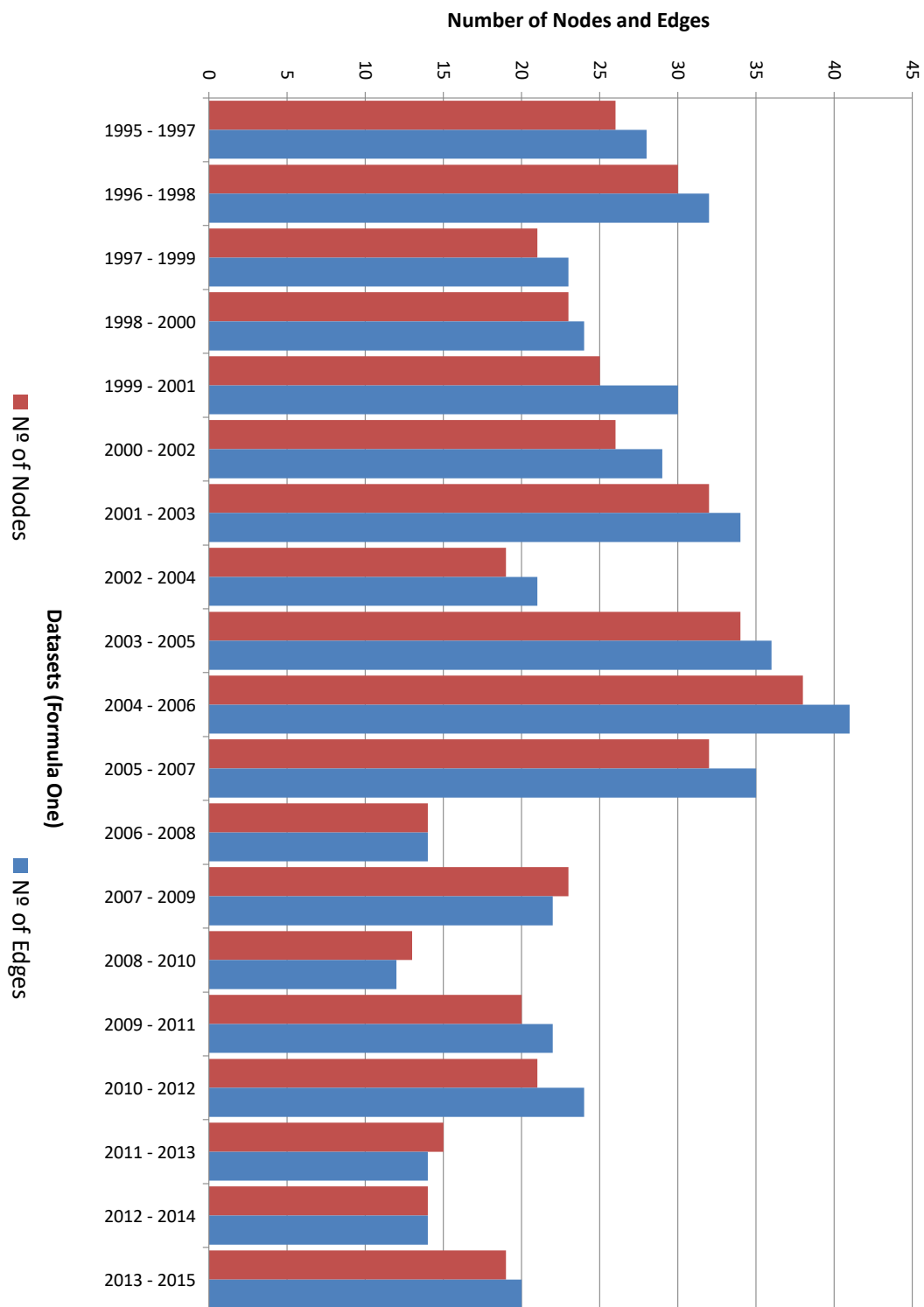


Figure A.2: Number of Nodes and Edges (Formula One)

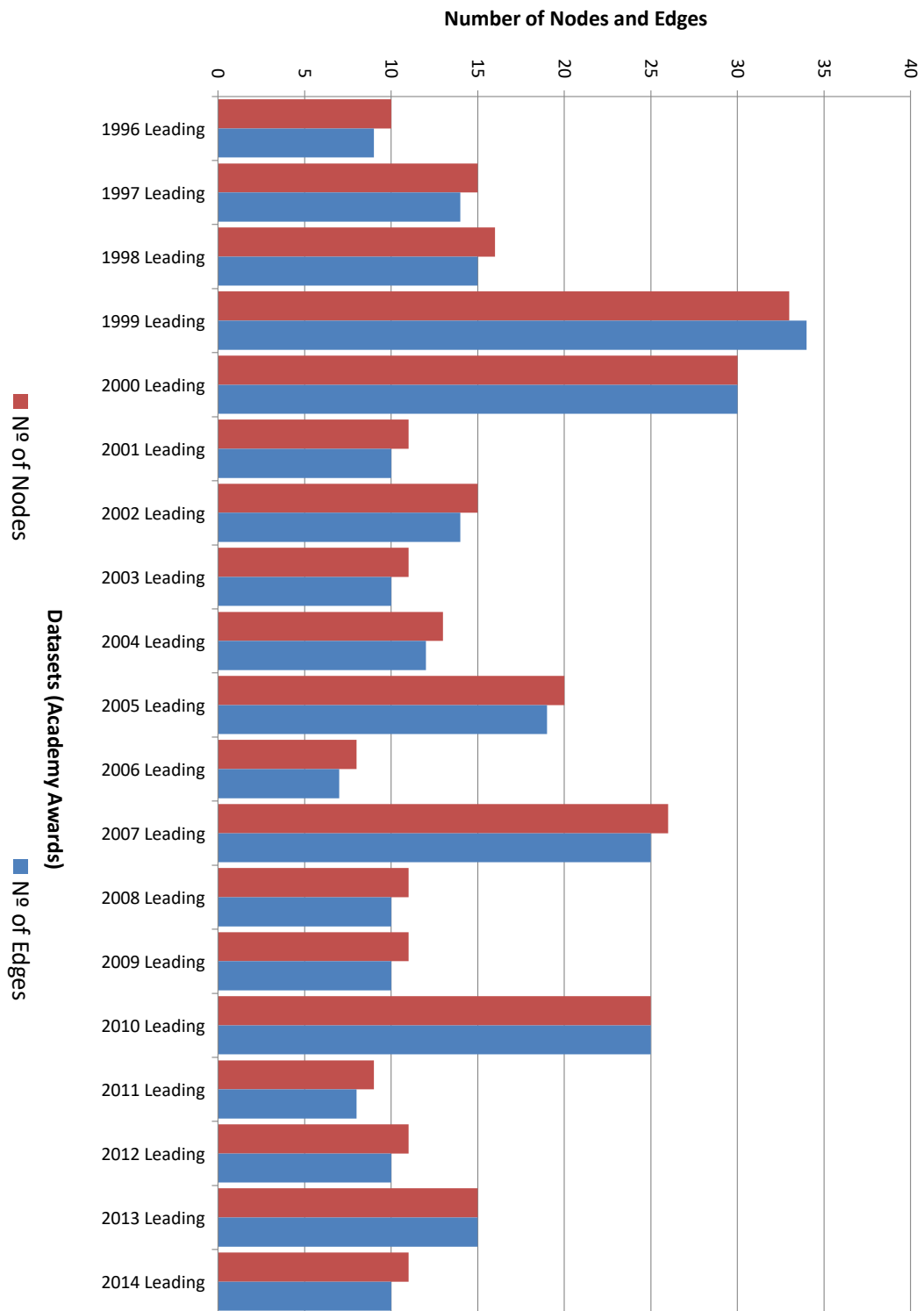


Figure A.3: Number of Nodes and Edges (Academy Awards) (1)

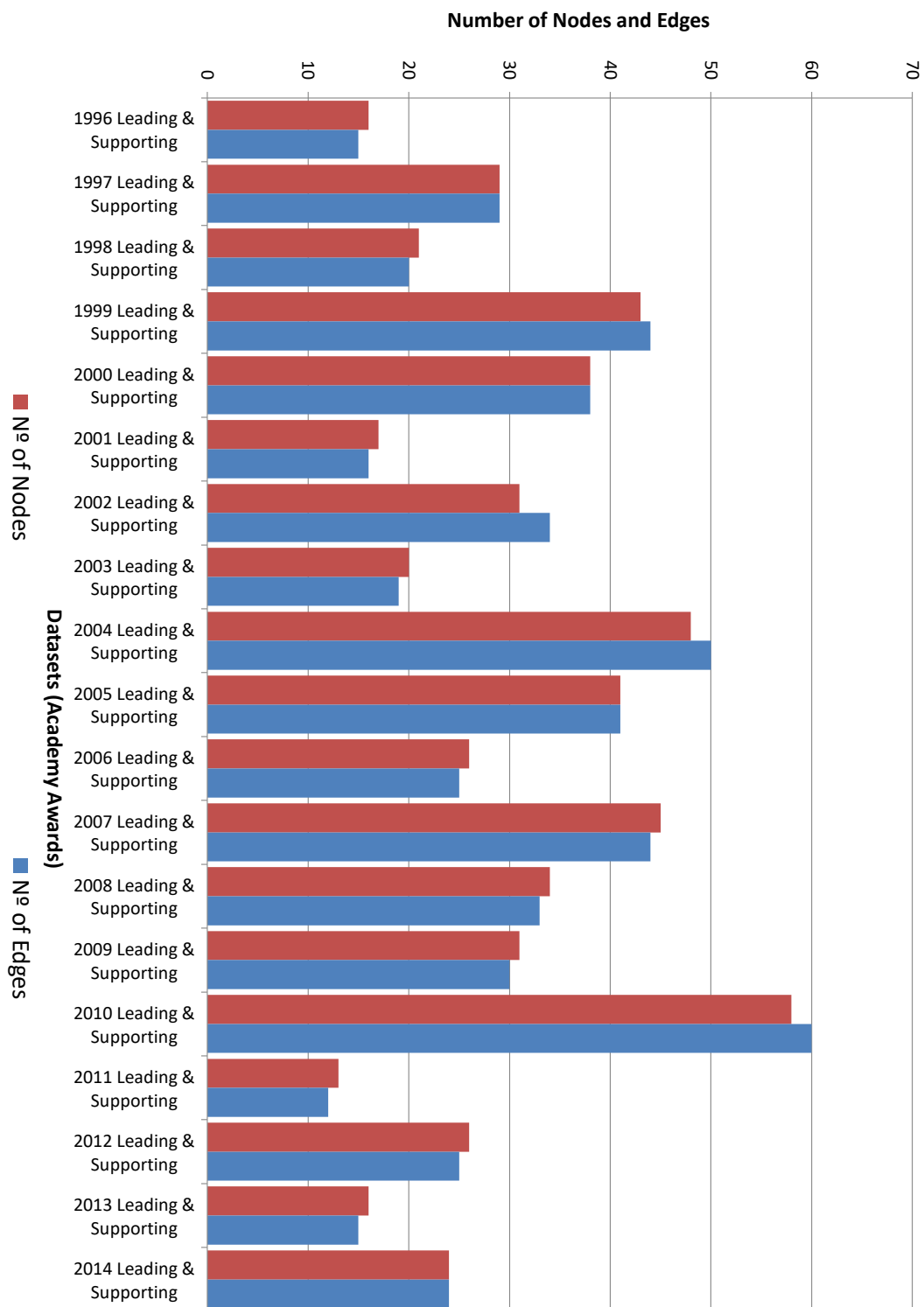


Figure A.4: Number of Nodes and Edges (Academy Awards) (2)

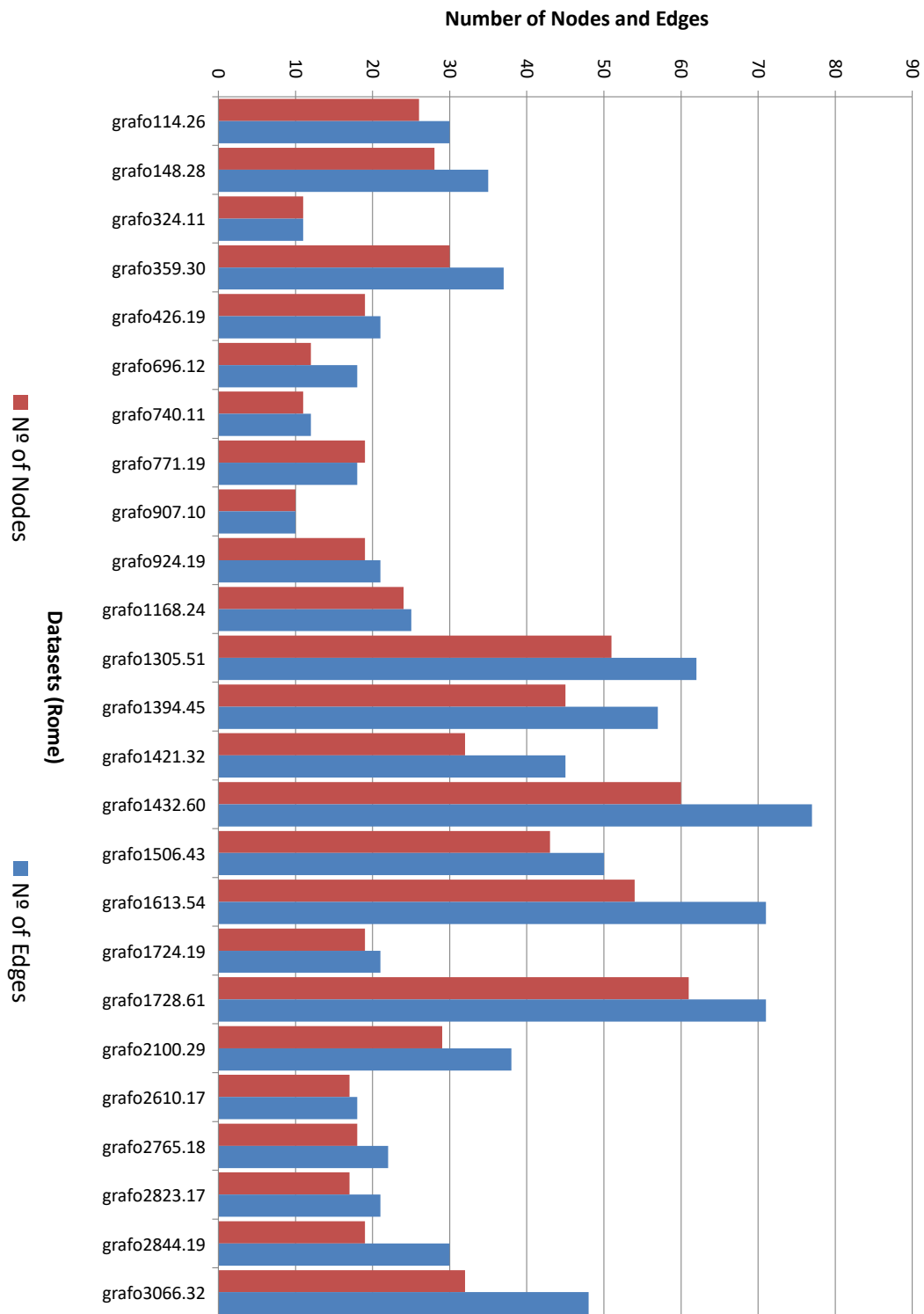


Figure A.5: Number of Nodes and Edges (Rome) (1)

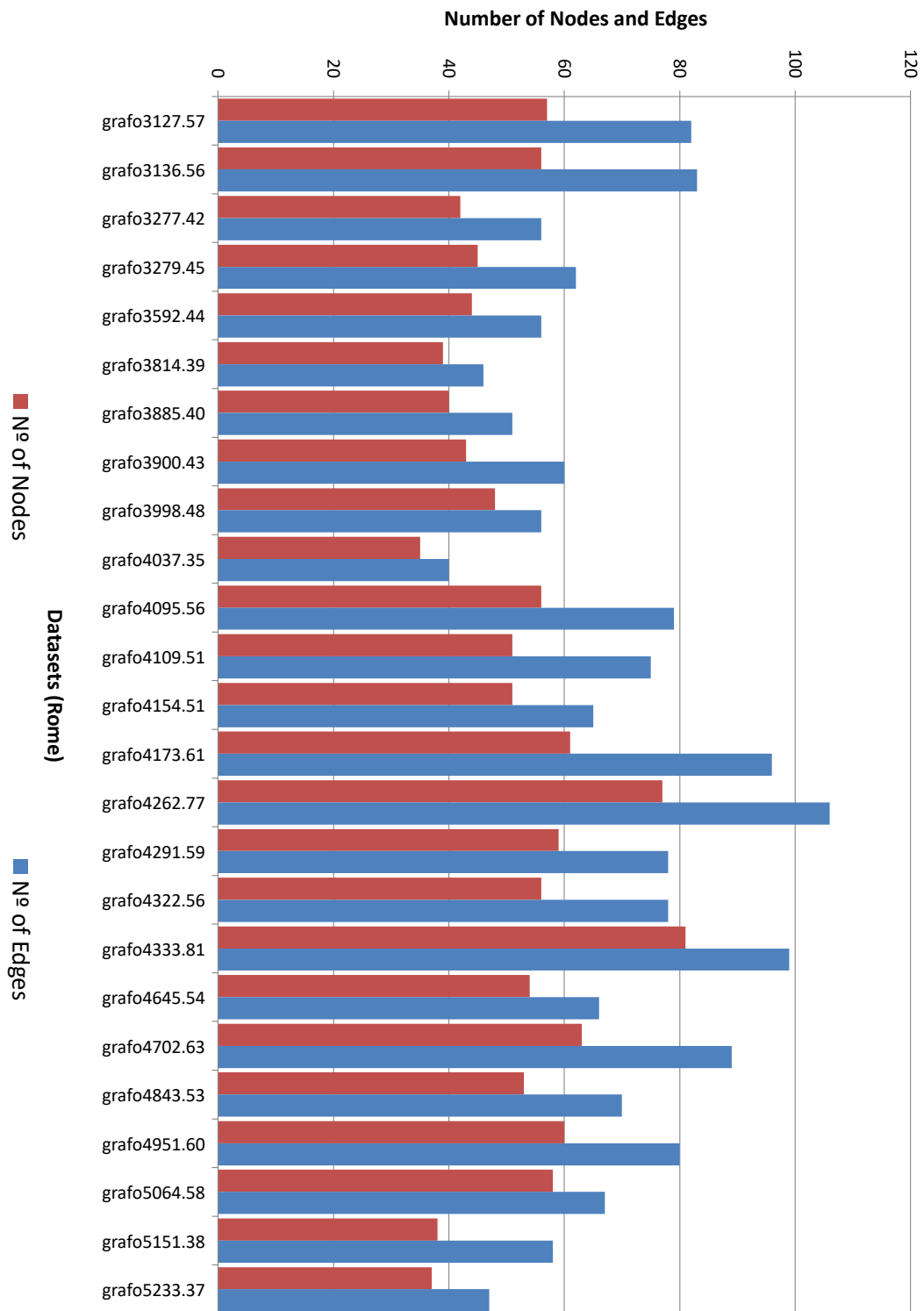


Figure A.6: Number of Nodes and Edges (Rome) (2)

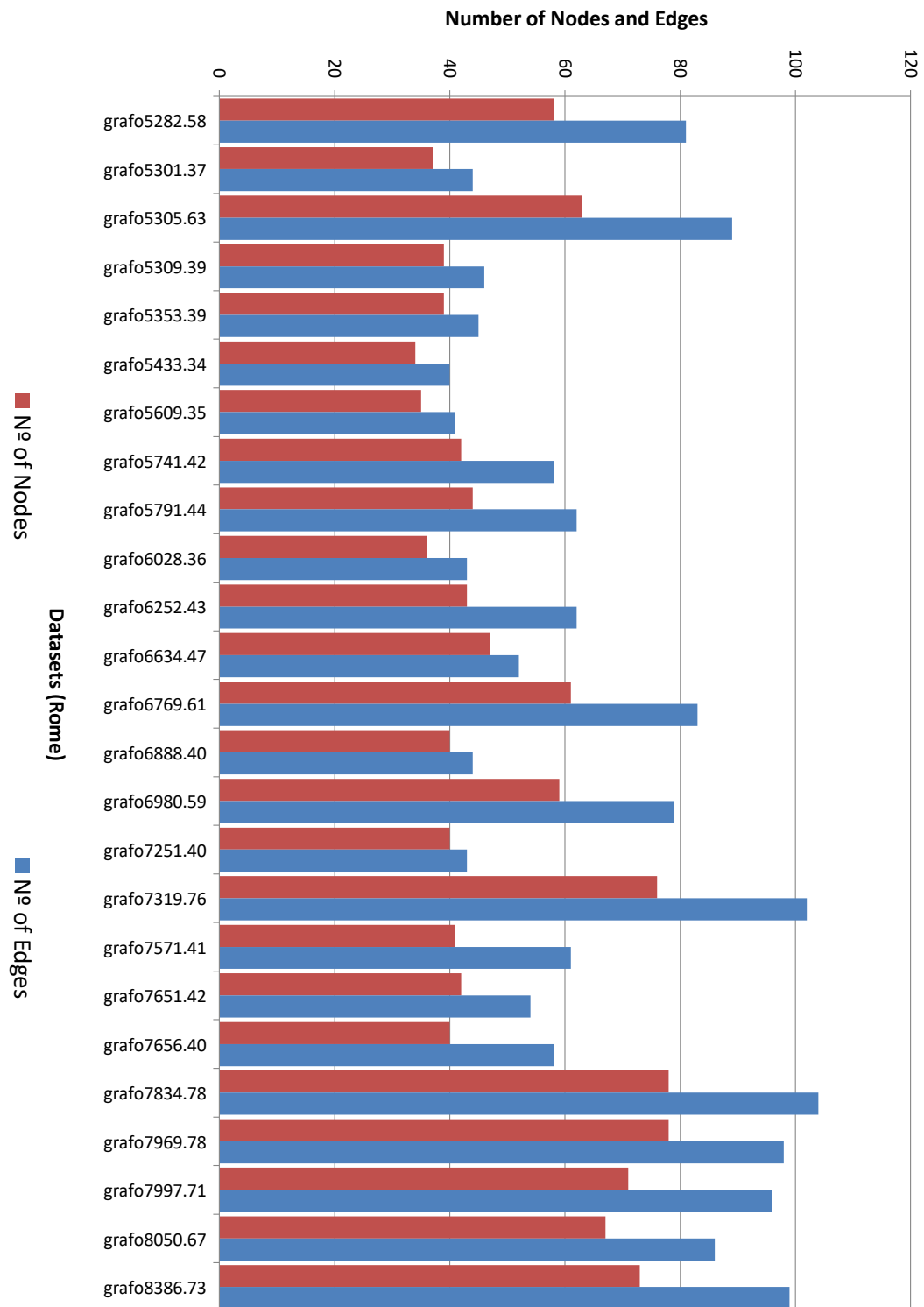


Figure A.7: Number of Nodes and Edges) (Rome) (3)

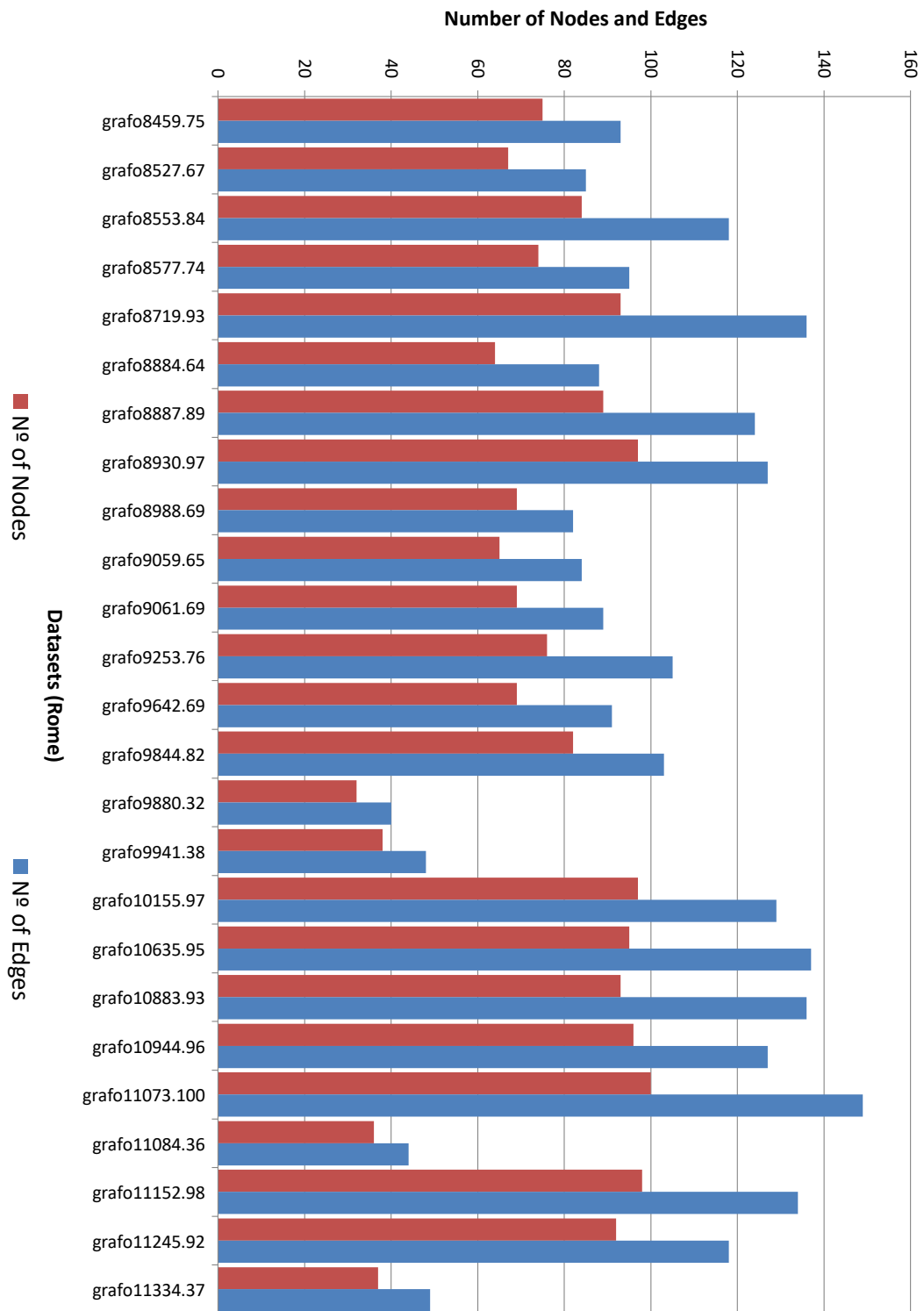


Figure A.8: Number of Nodes and Edges (Rome) (4)

A.2 Time taken

The following charts show the time taken vs the number of nodes and edges, split by dataset.

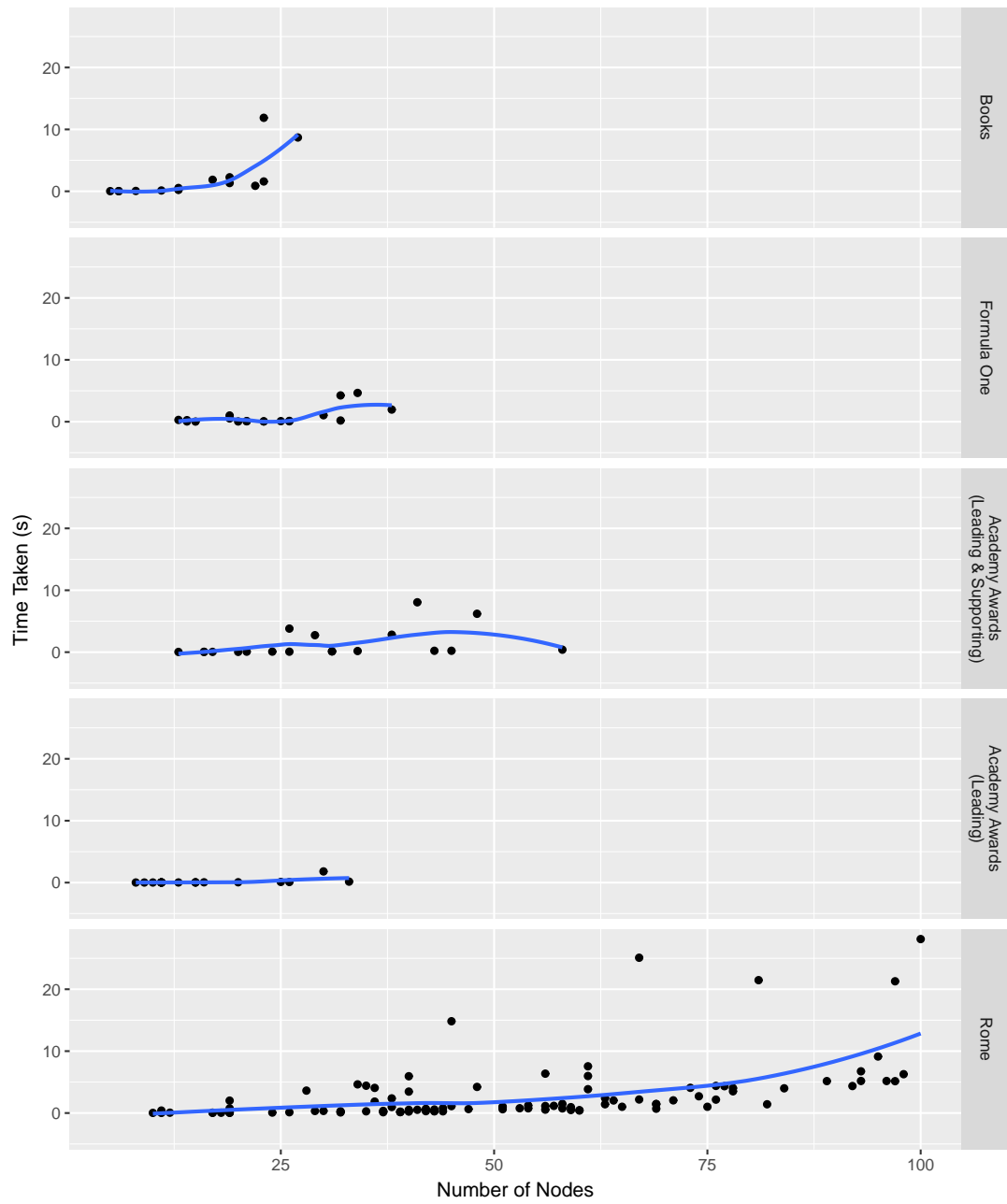


Figure A.9: Time Taken (s) vs Number of Nodes By Dataset

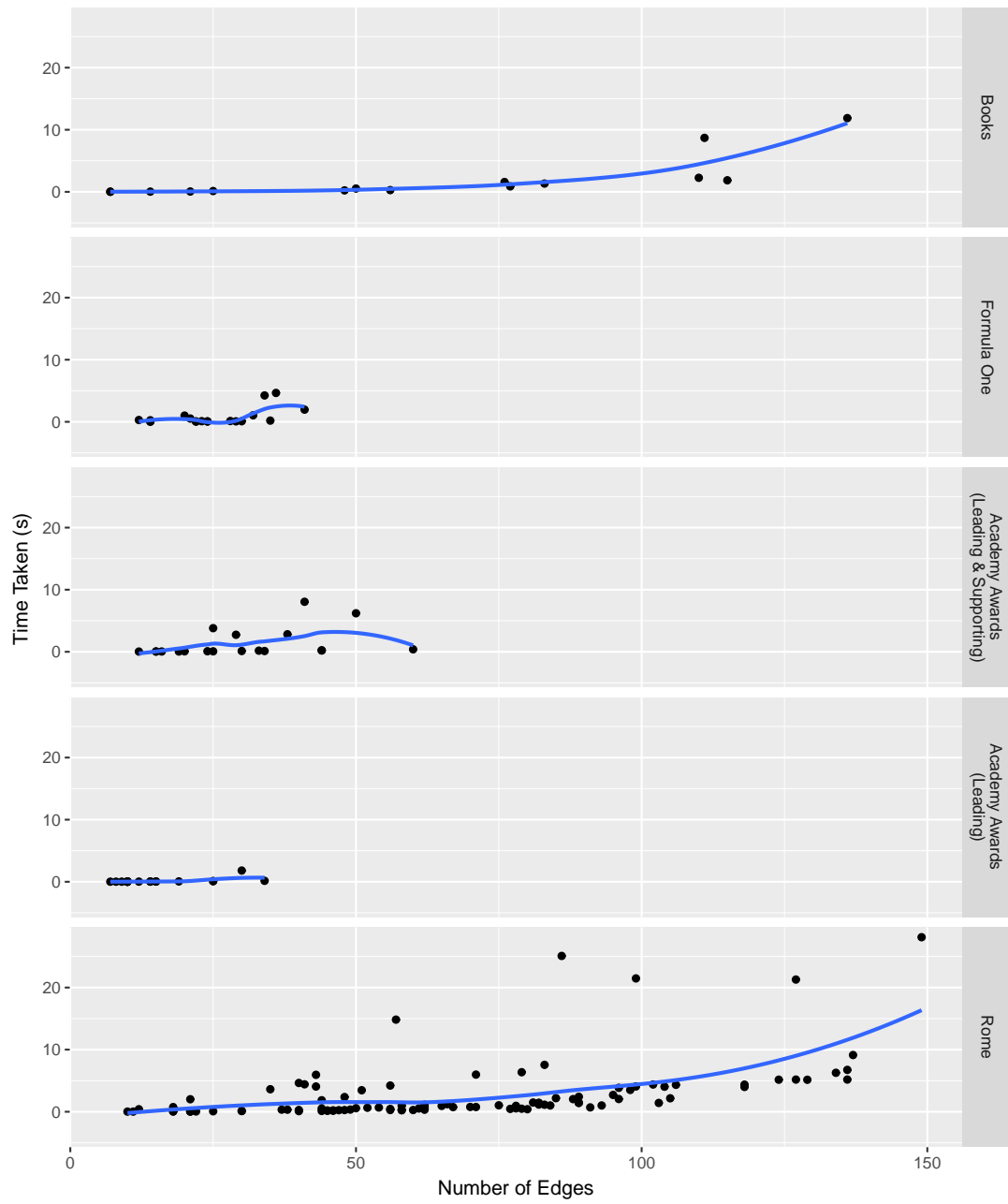


Figure A.10: Time Taken (s) vs Number of Edges By Dataset

A.3 Density of graphs

The following charts show the edge density.

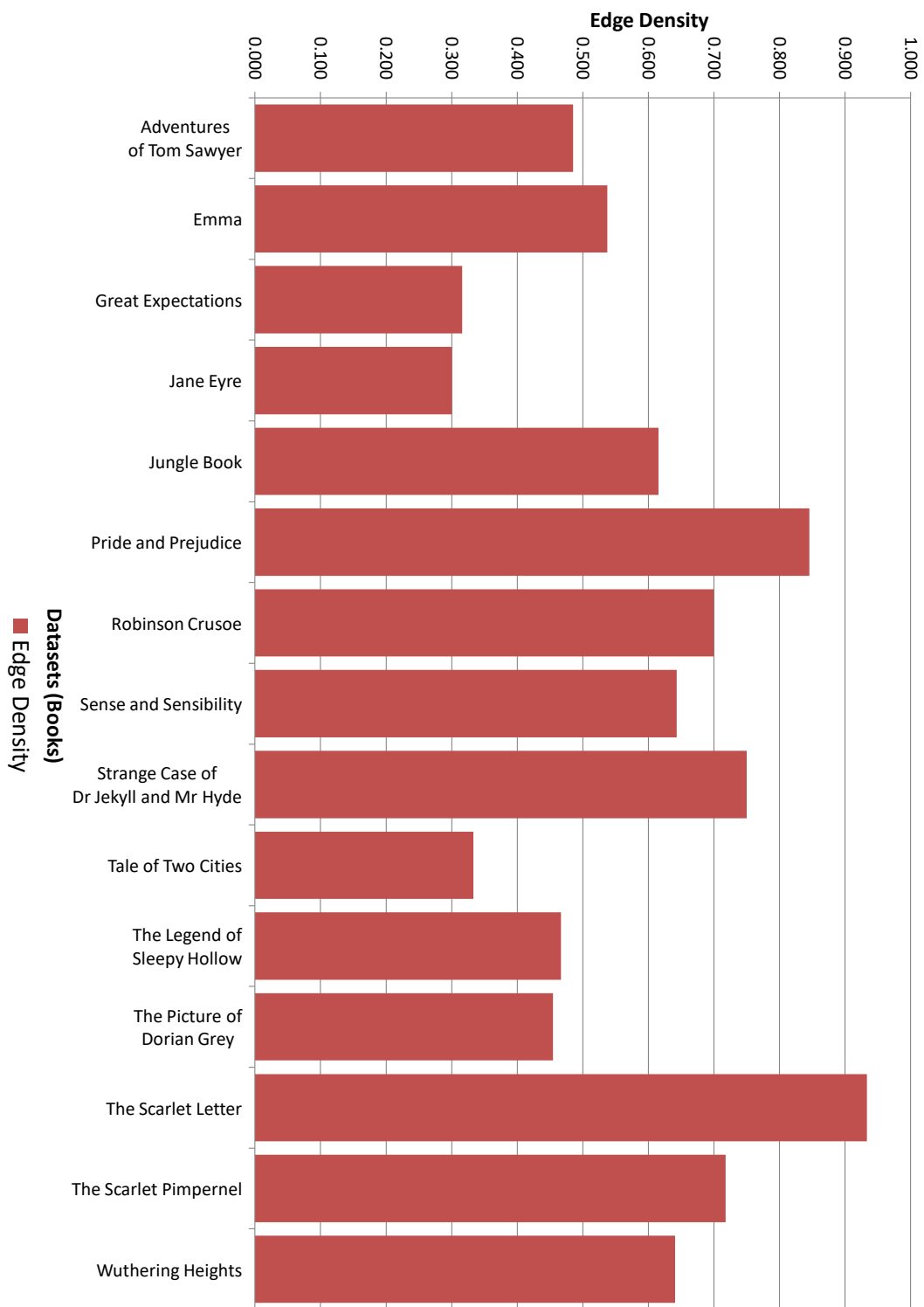


Figure A.11: Edge Density (Books)

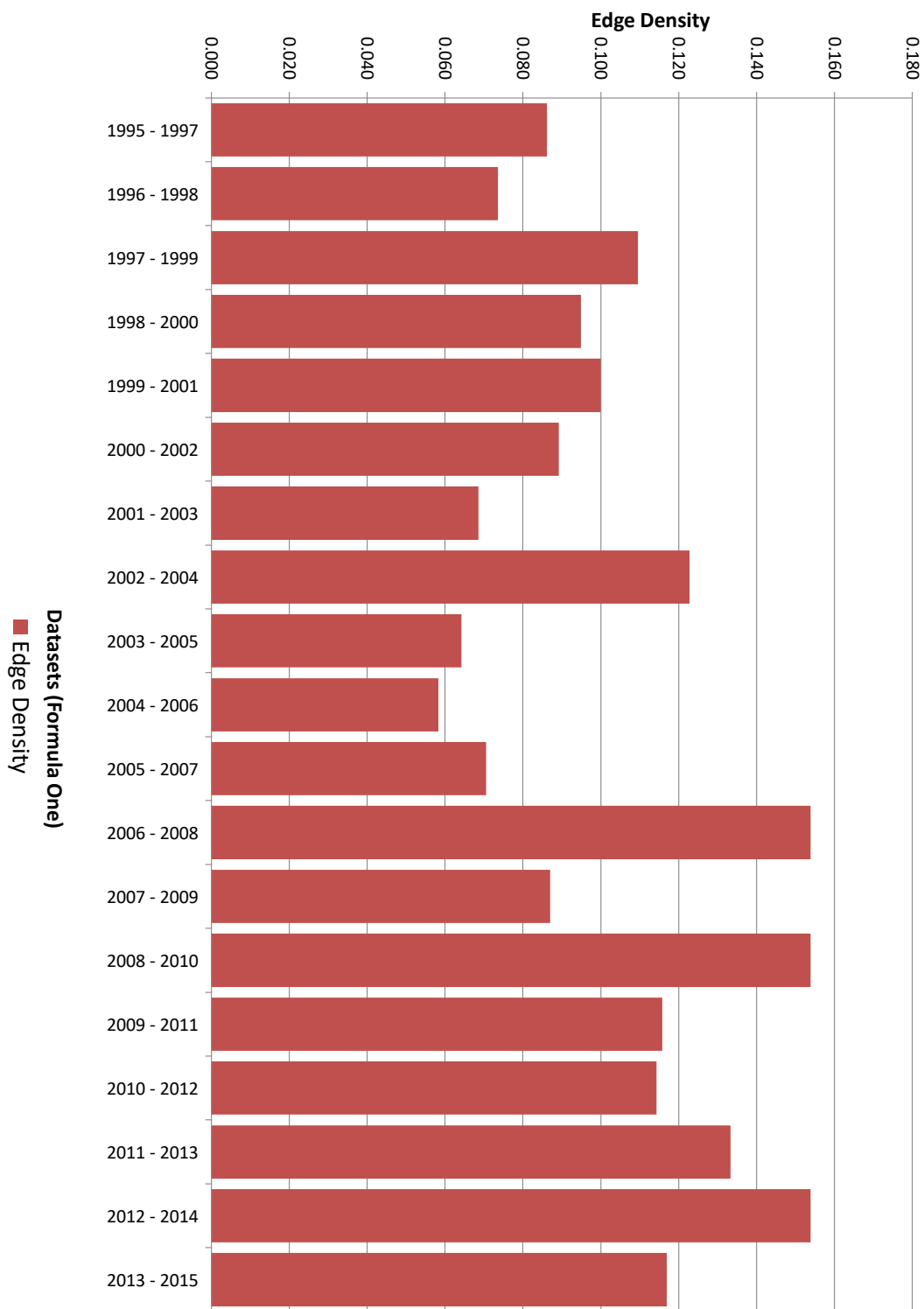


Figure A.12: Edge Density (Formula One)

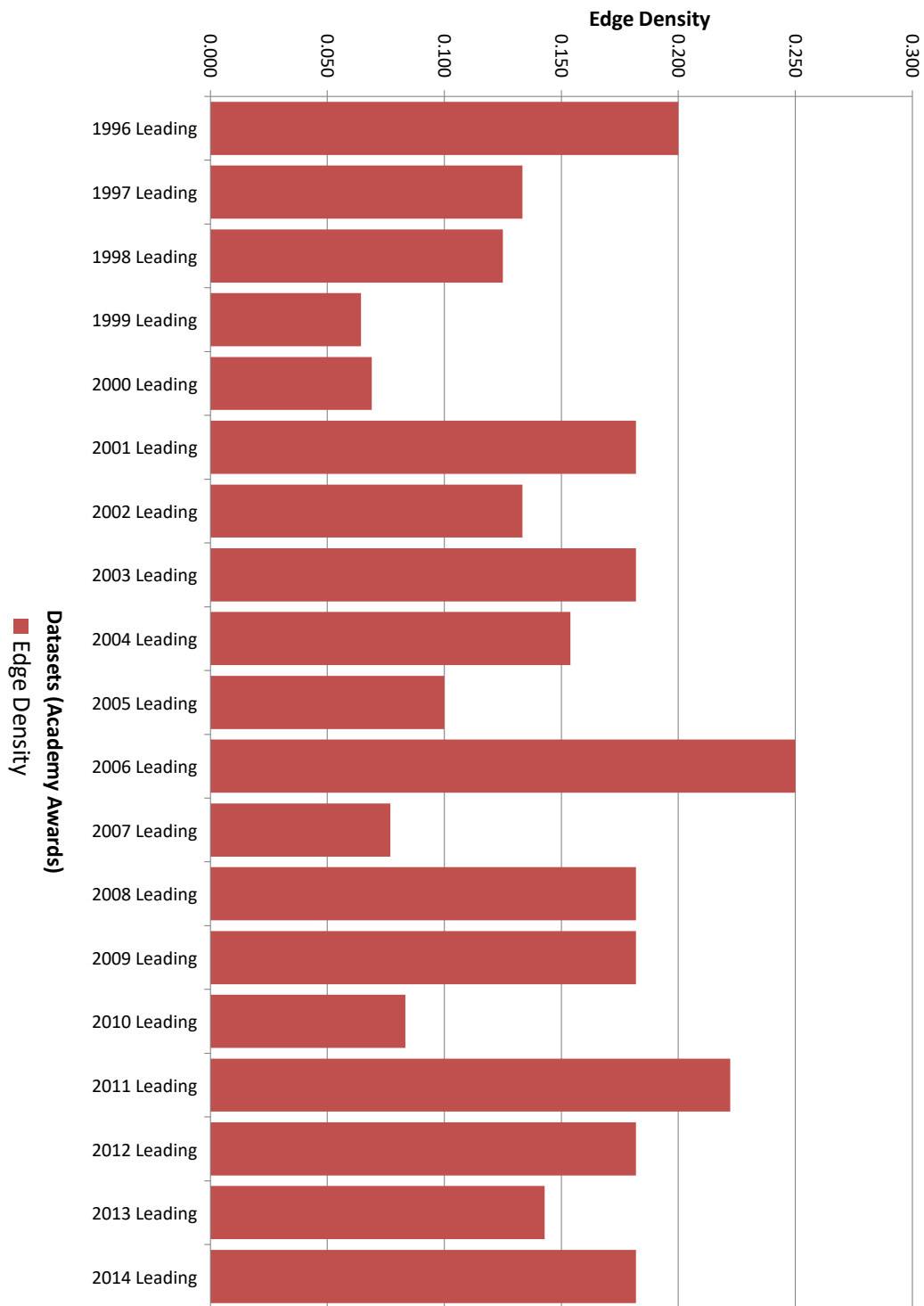


Figure A.13: Edge Density (Academy Awards) (1)

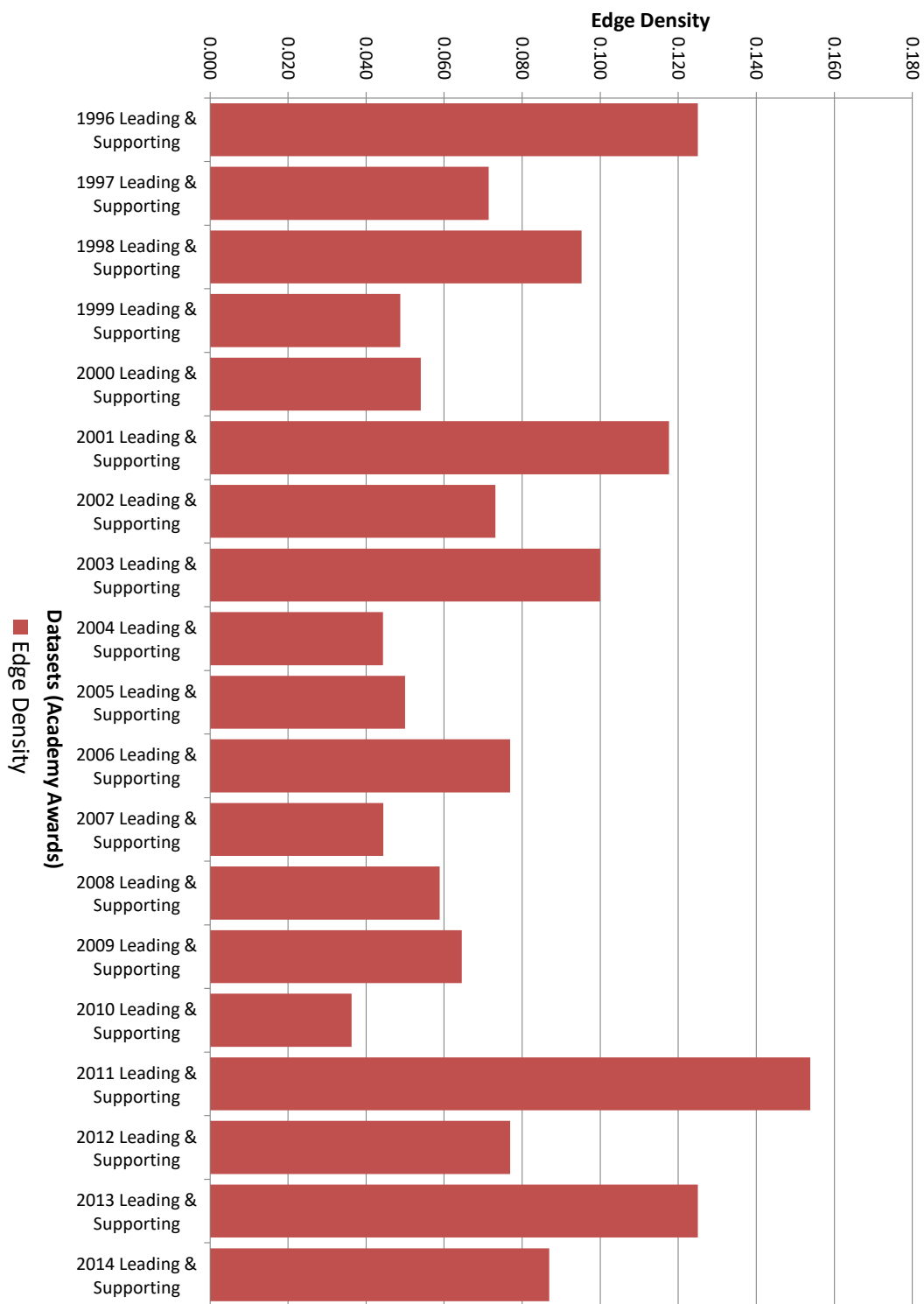


Figure A.14: Edge Density (Academy Awards) (2)

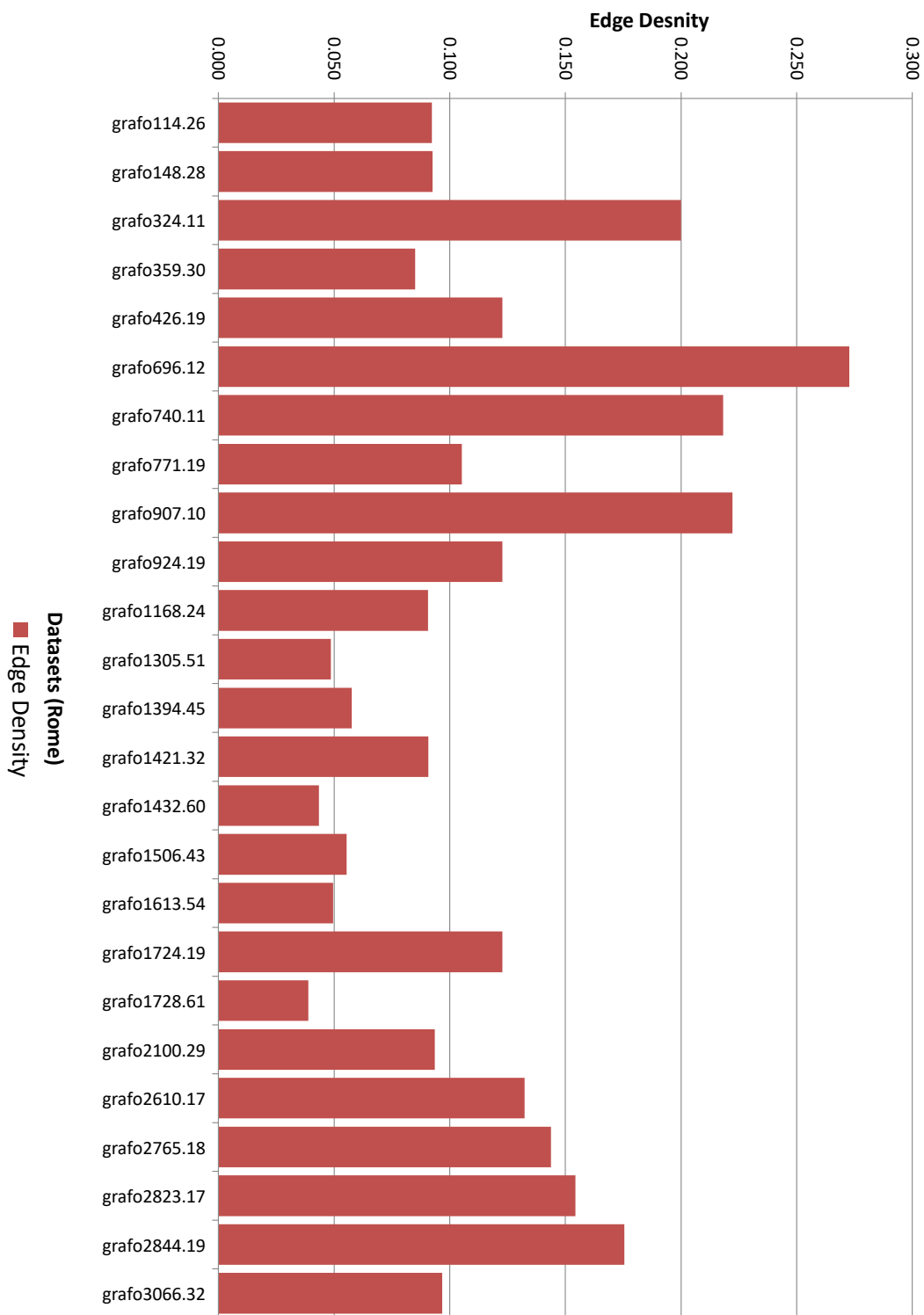


Figure A.15: Edge Density (Rome) (1)

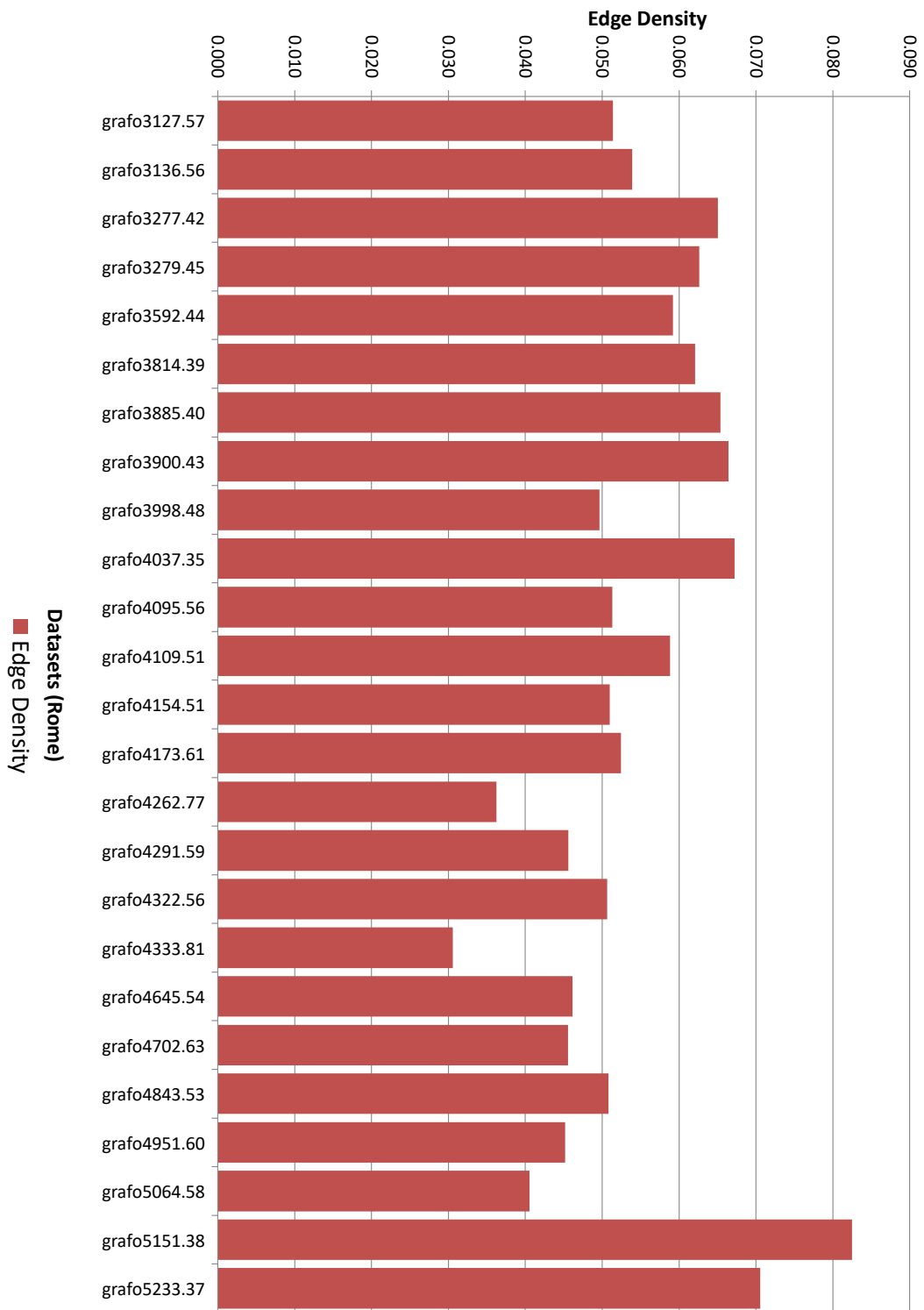


Figure A.16: Edge Density (Rome) (2)

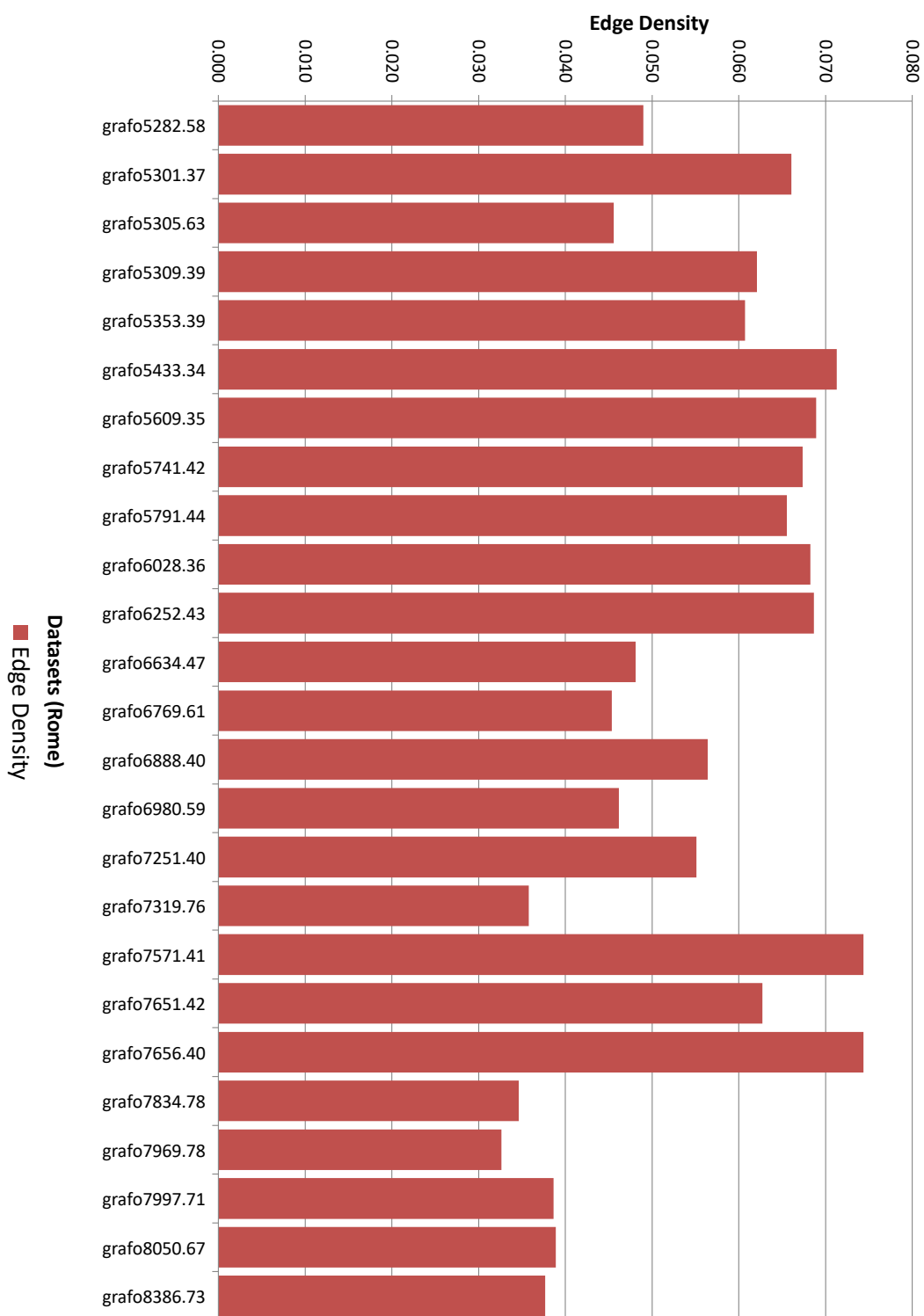


Figure A.17: Edge Density (Rome) (3)

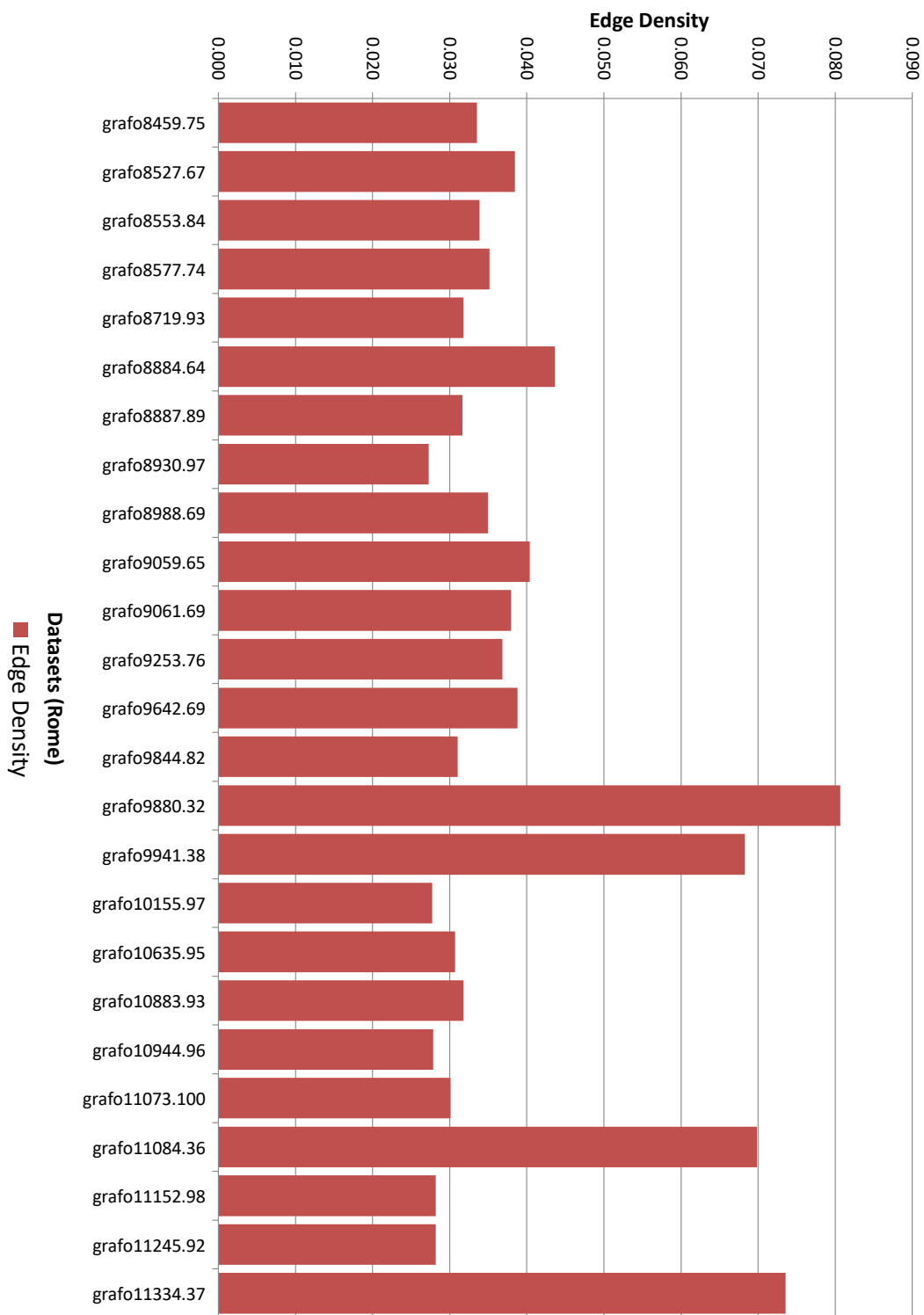


Figure A.18: Edge Density (Rome) (4)

A.4 Size of Largest Clique

Only one dataset contains cliques, and the largest clique in each example is shown in the following chart.

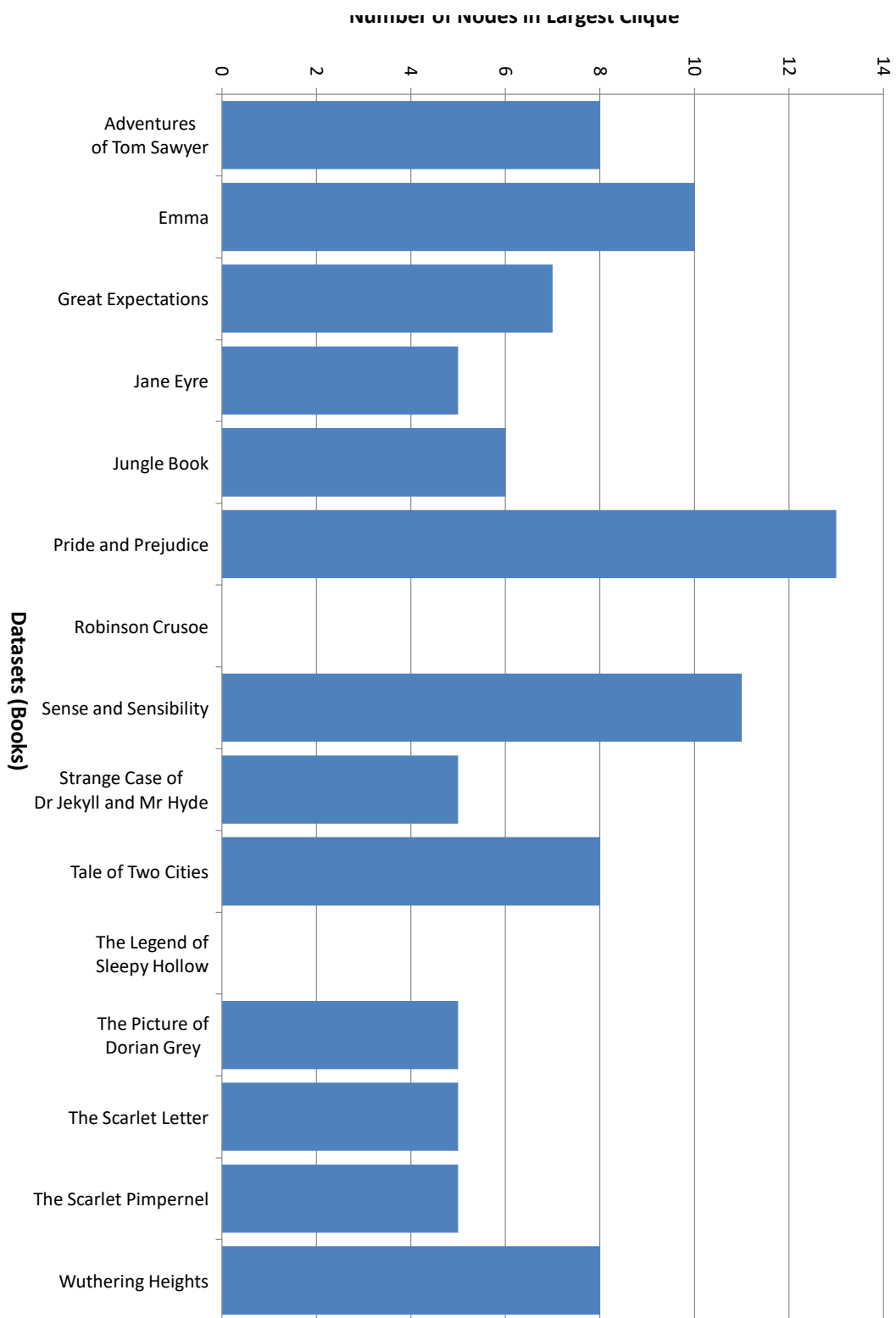


Figure A.19: Size of Largest Clique (Books)

A.5 Patterns by Type

The following charts detail the types of pattern found within each dataset.

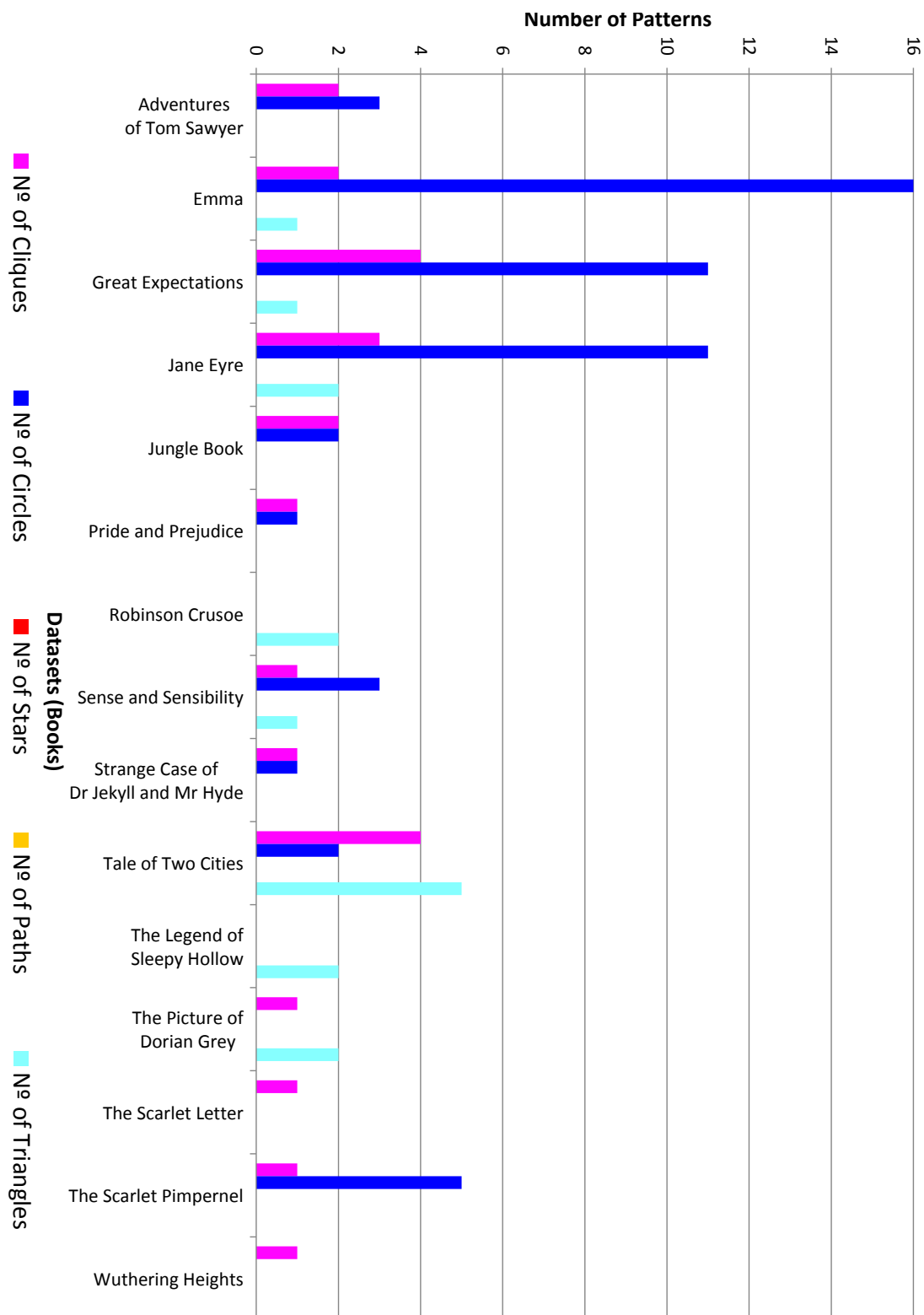


Figure A.20: Number of patterns by type (Books)

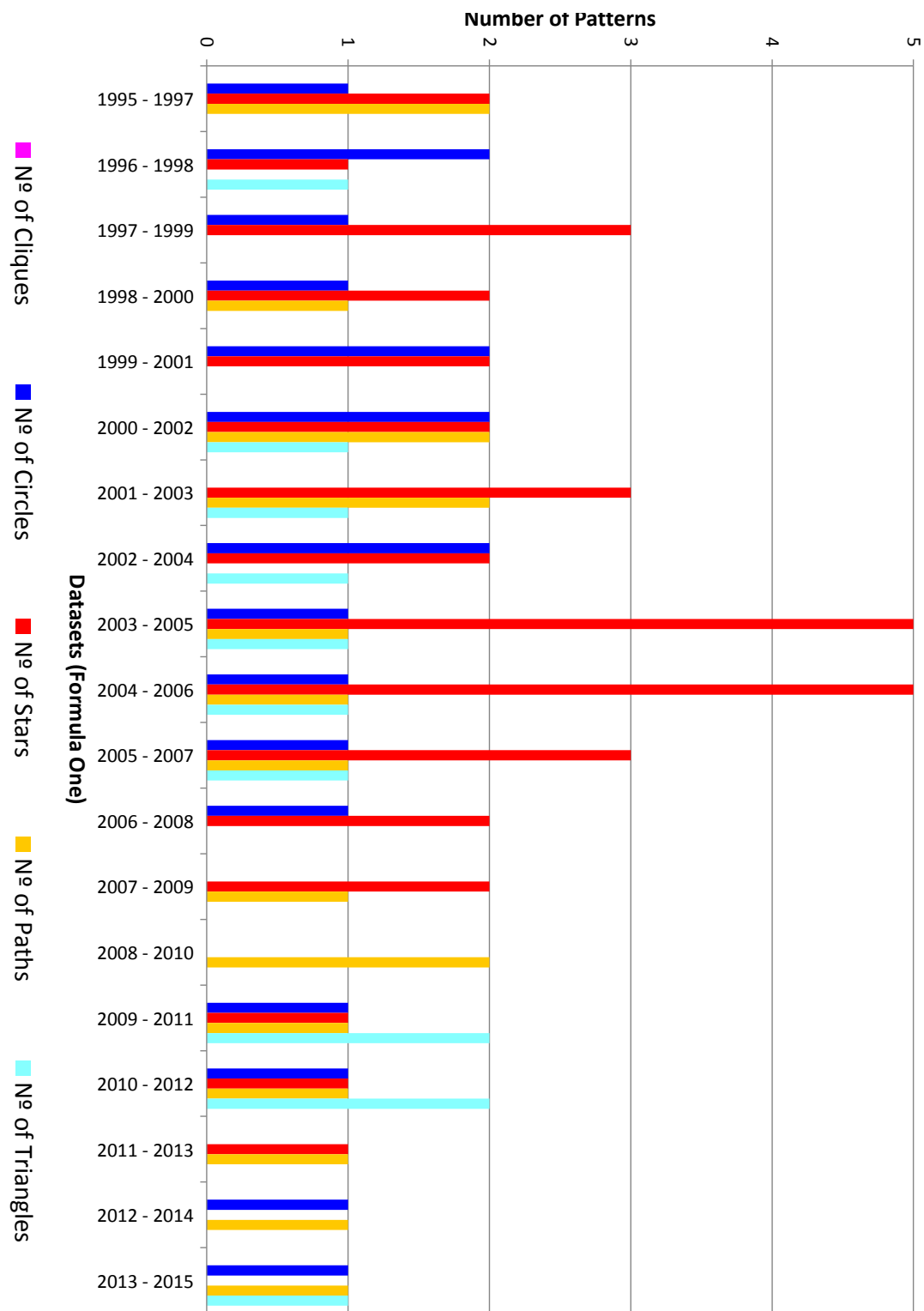


Figure A.21: Number of patterns by type (Formula One)

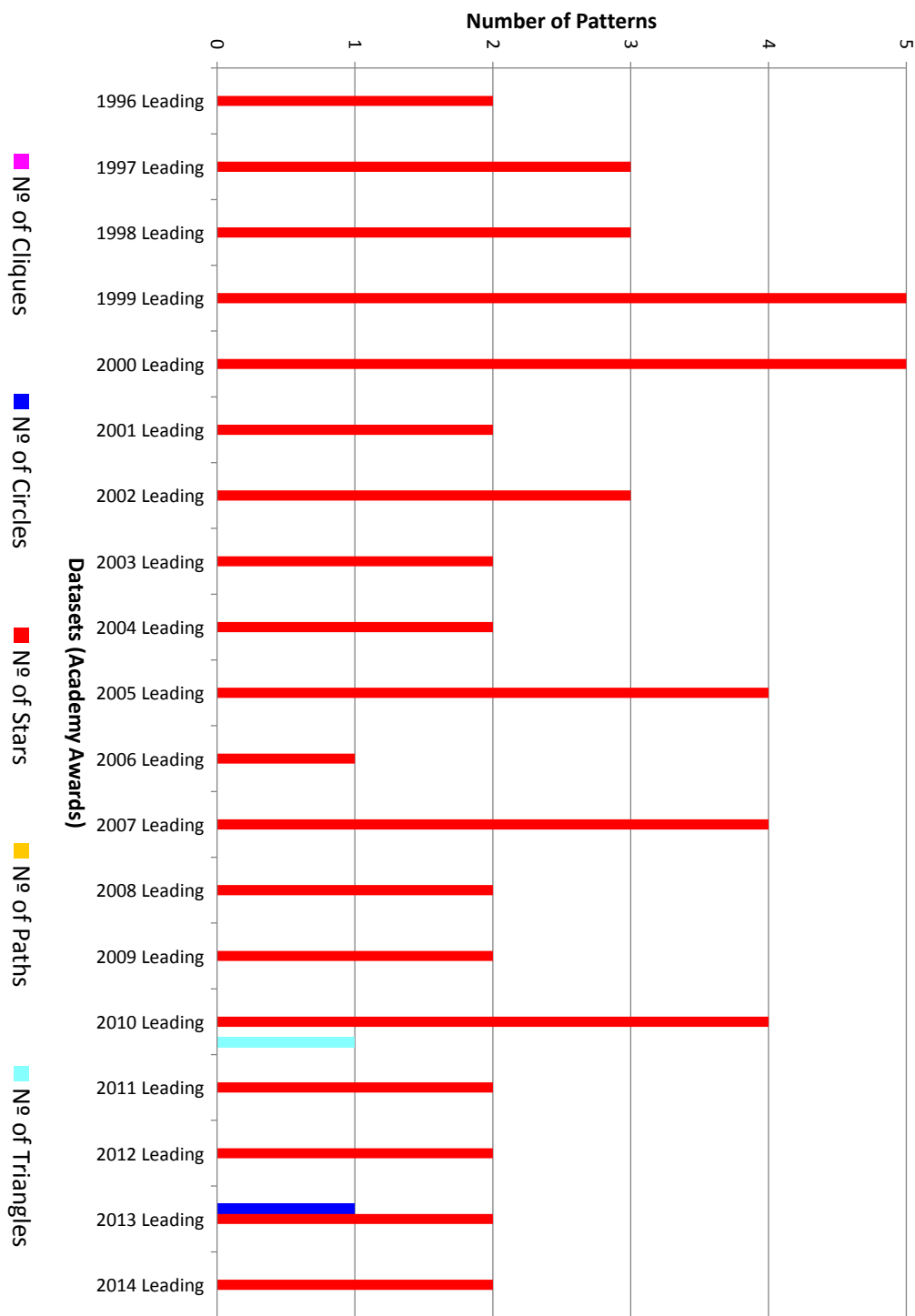


Figure A.22: Number of patterns by type (Academy Awards) (1)

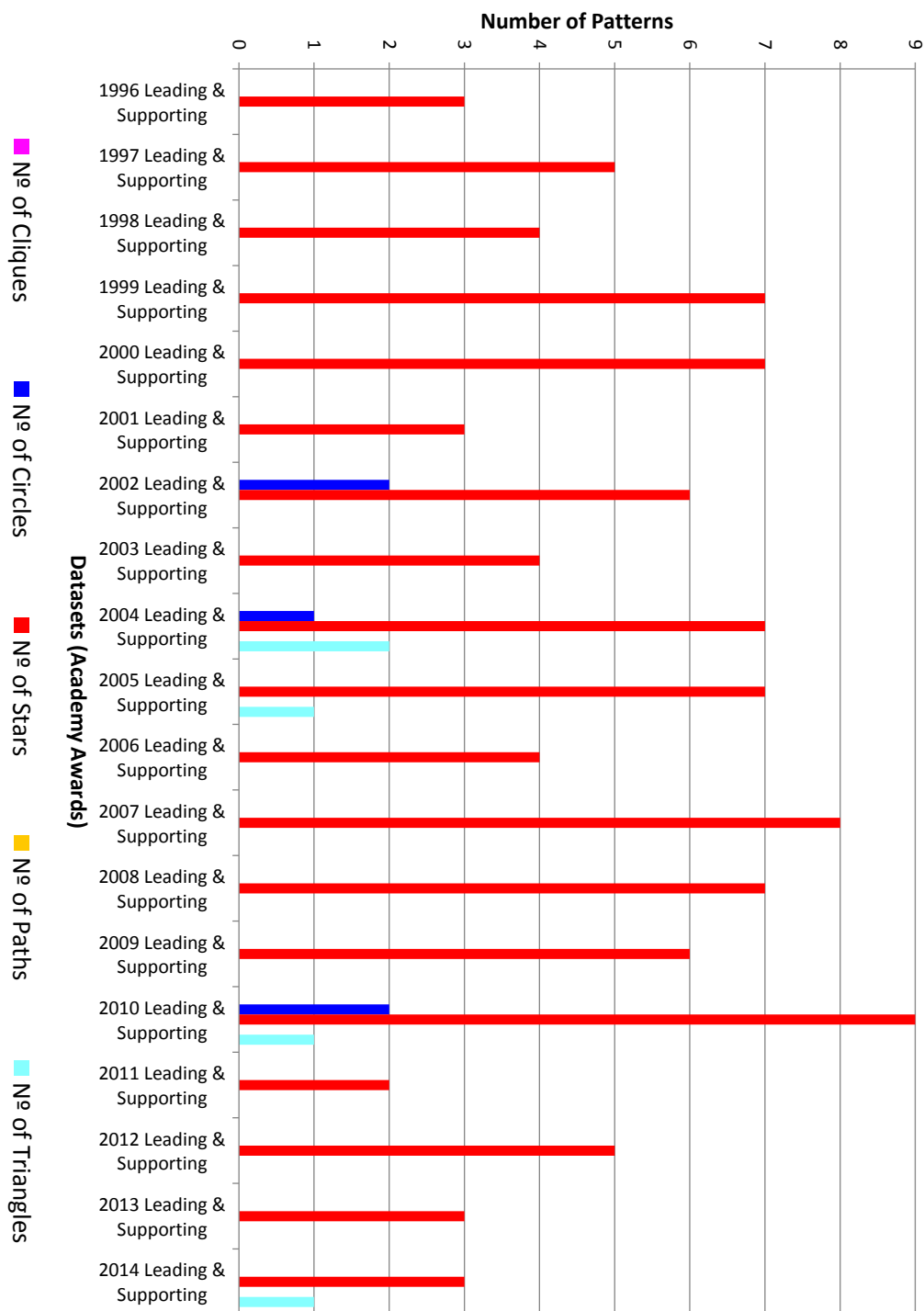


Figure A.23: Number of patterns by type (Academy Awards) (2)

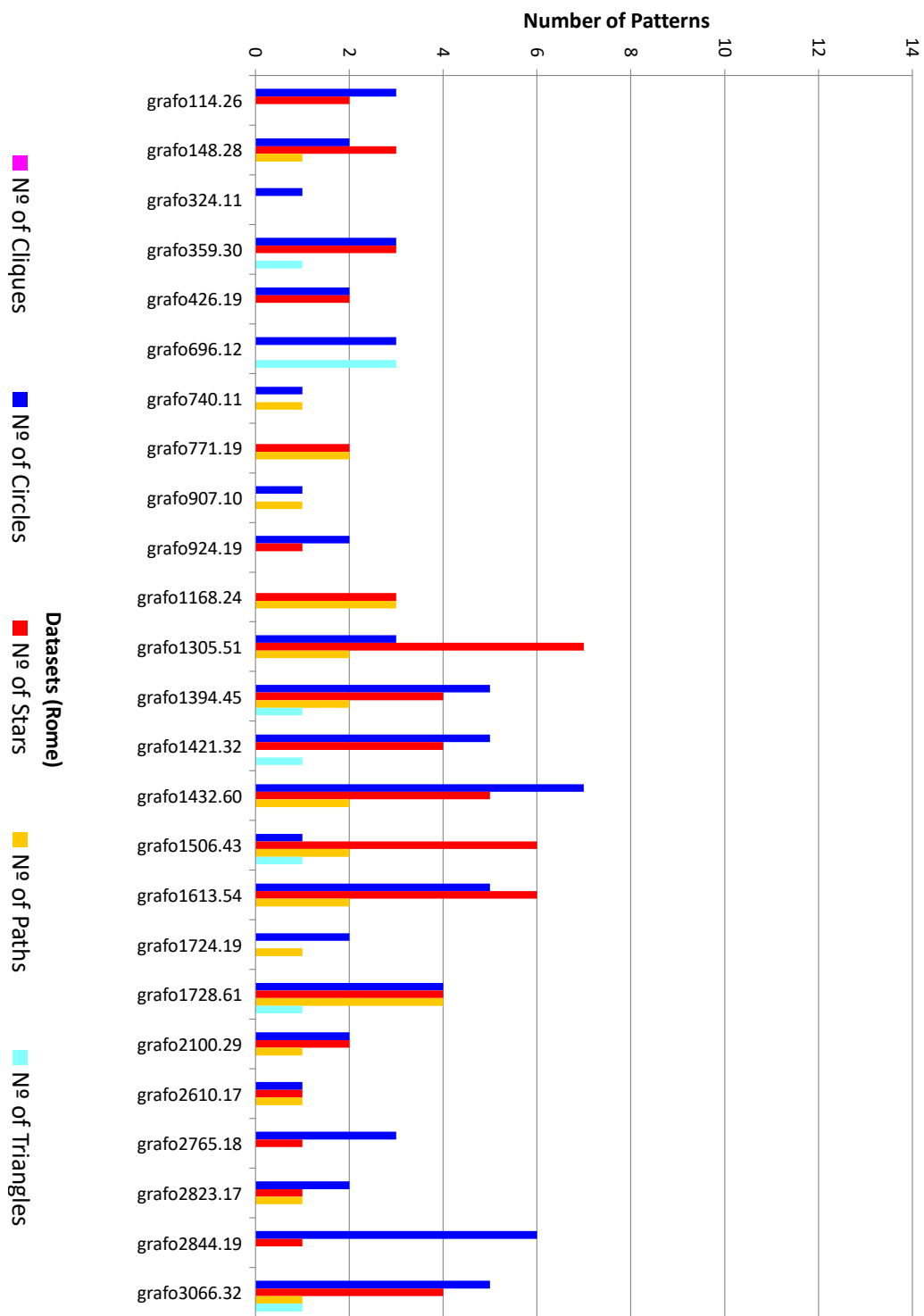


Figure A.24: Number of patterns by type (Rome) (1)

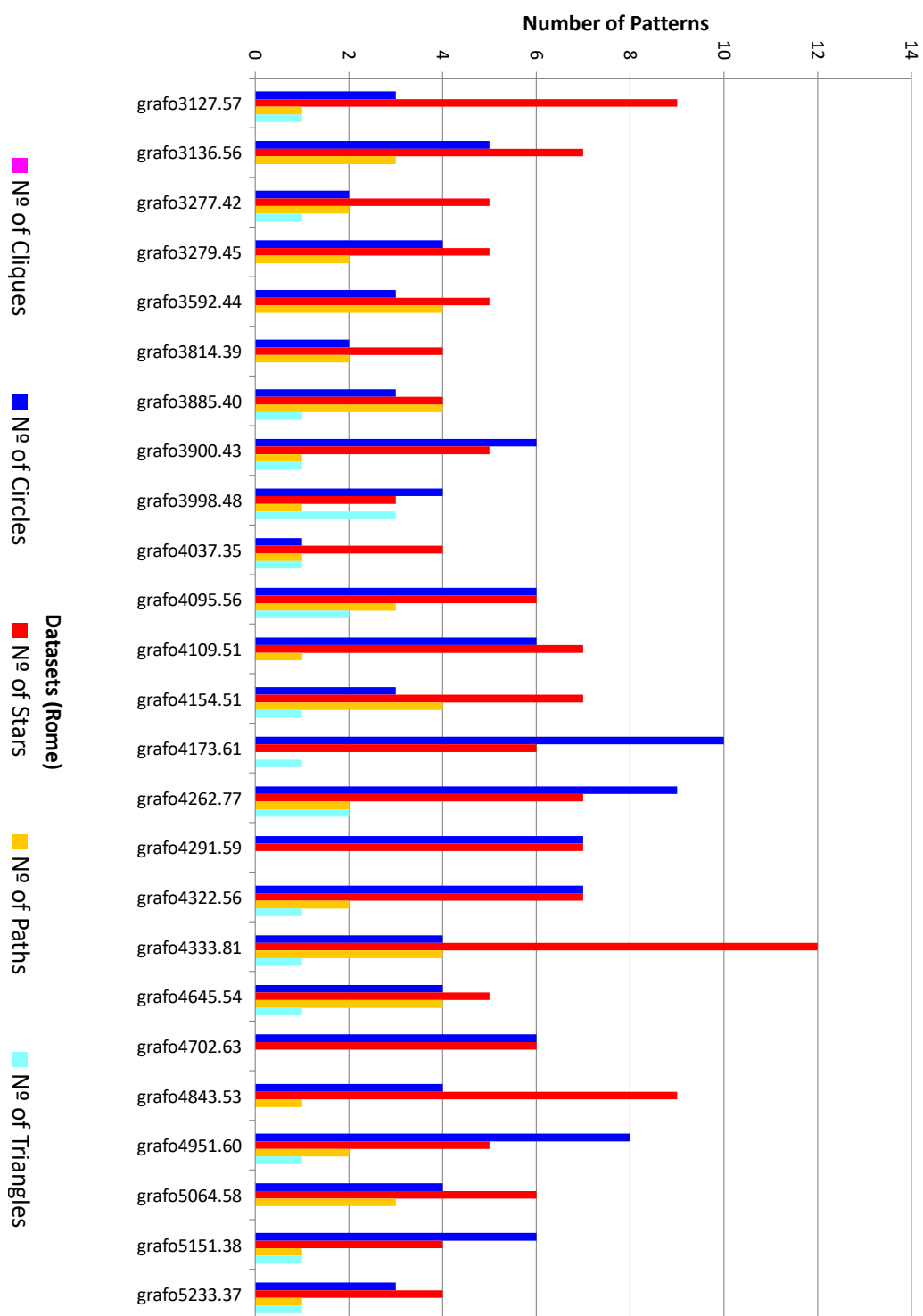


Figure A.25: Number of patterns by type (Rome) (2)

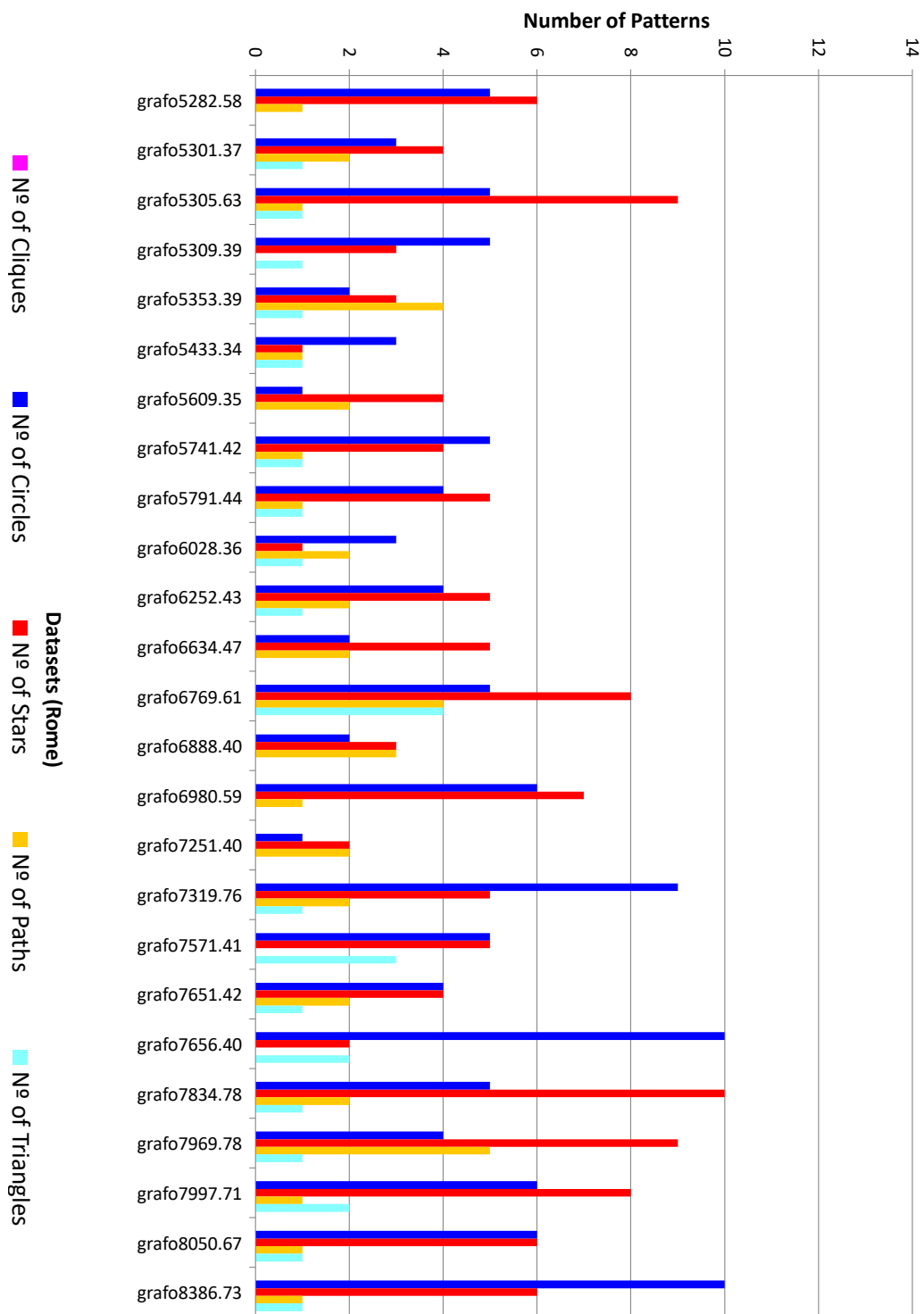


Figure A.26: Number of patterns by type (Rome) (3)

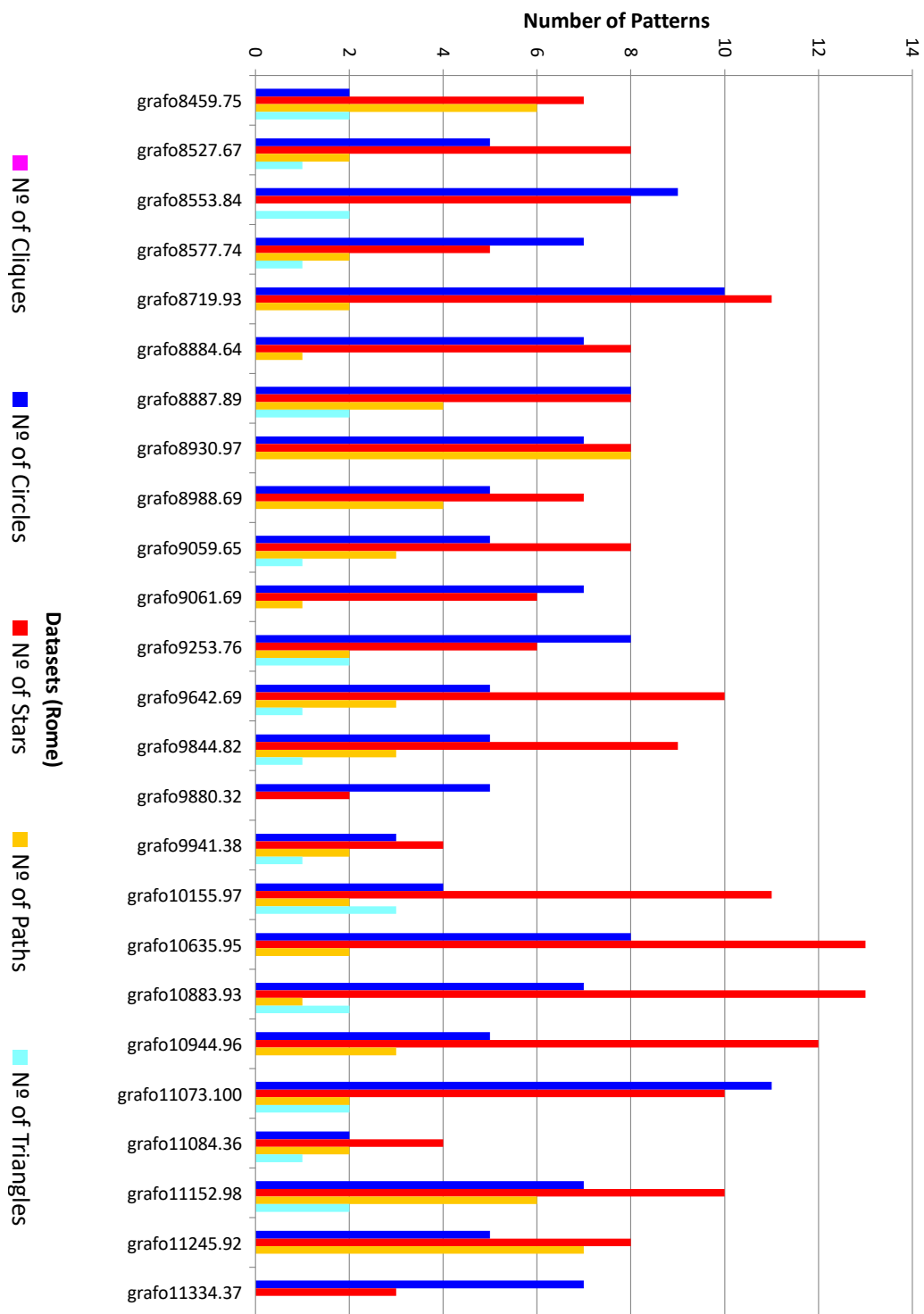


Figure A.27: Number of patterns by type (Rome) (4)

A.6 Discarded and Drawn Patterns

The following charts display the number of patterns that have been drawn and discarded.

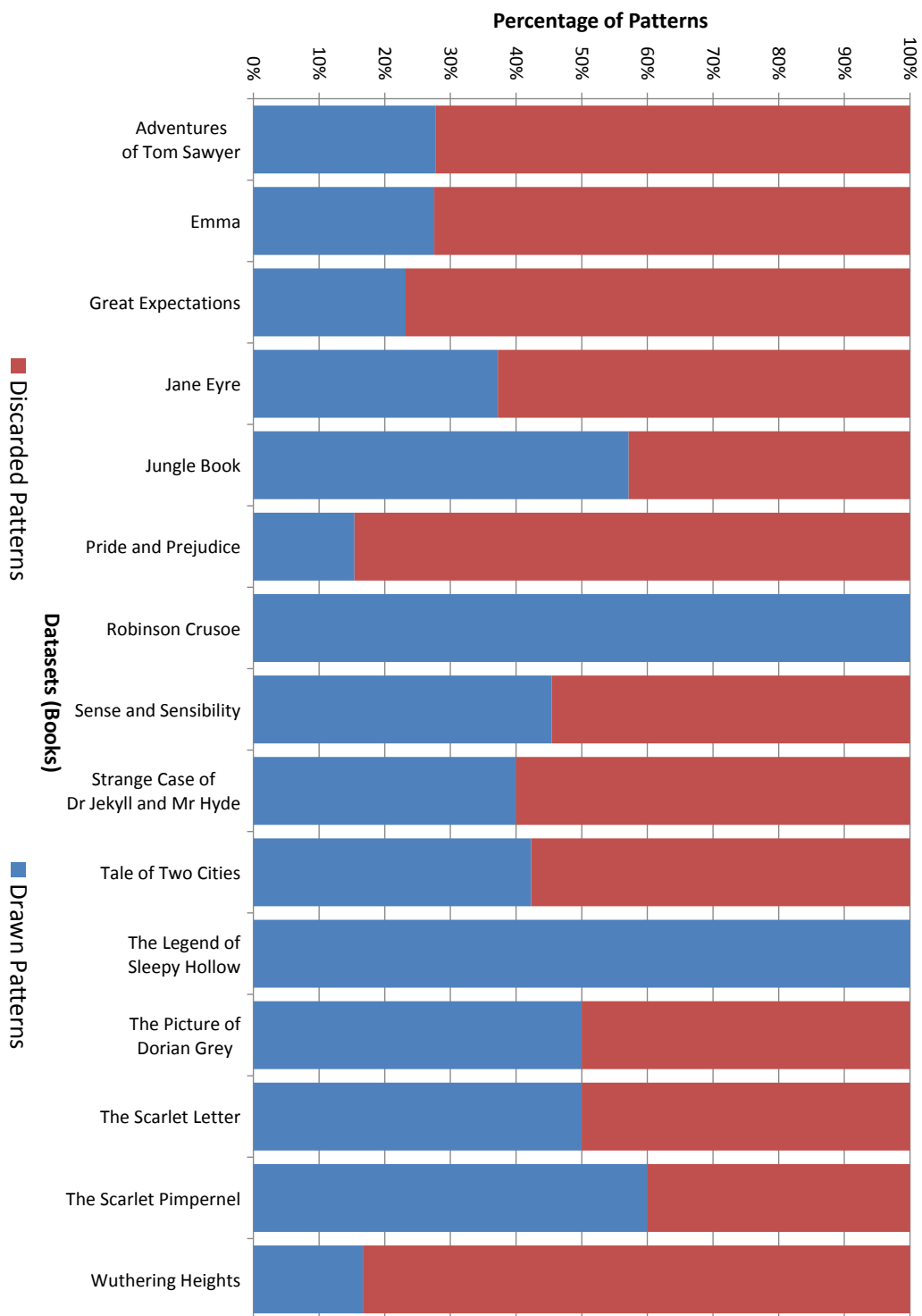


Figure A.28: Percentage of Patterns Drawn & Discarded (Books)

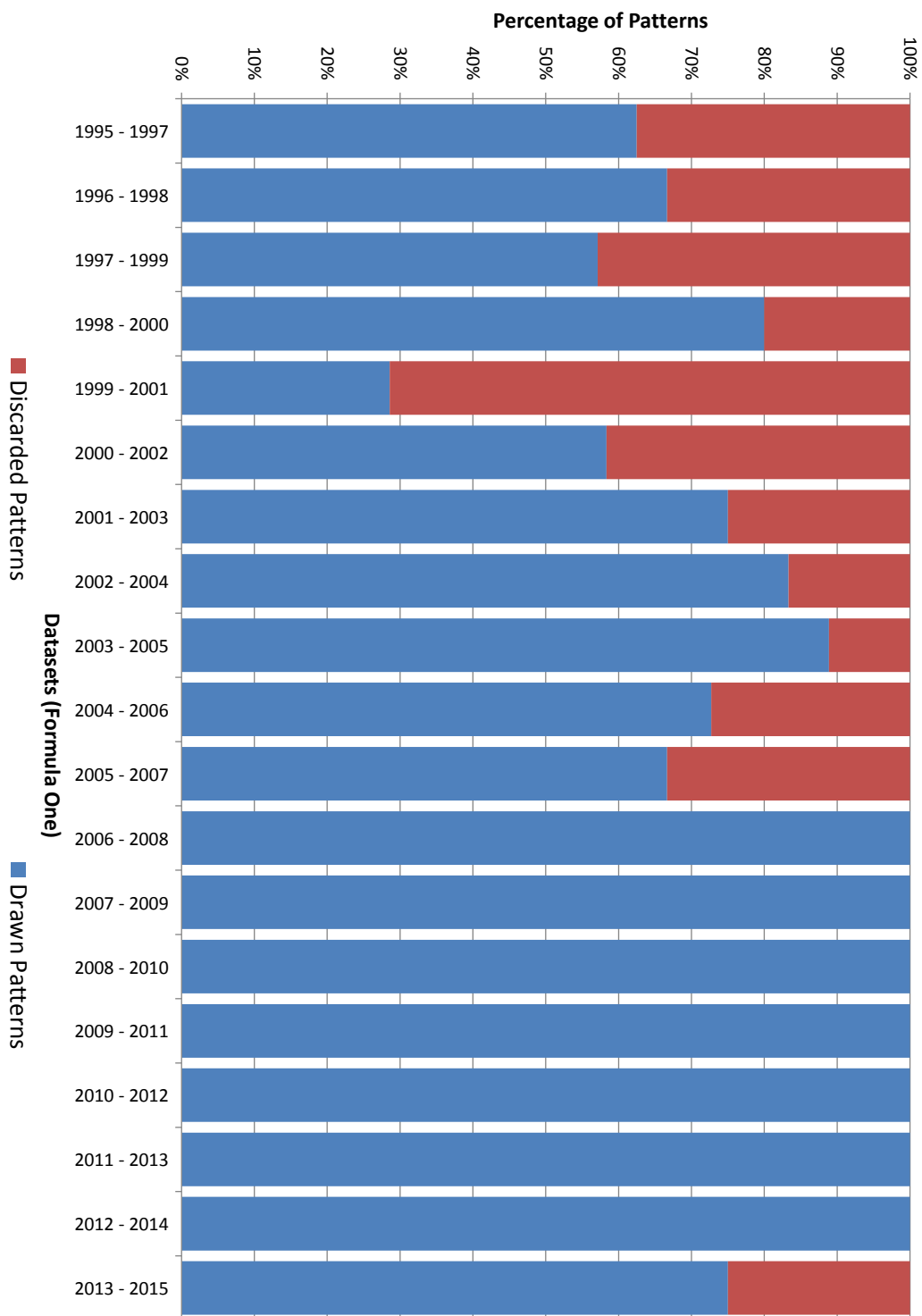


Figure A.29: Percentage of Patterns Drawn & Discarded (Formula One)

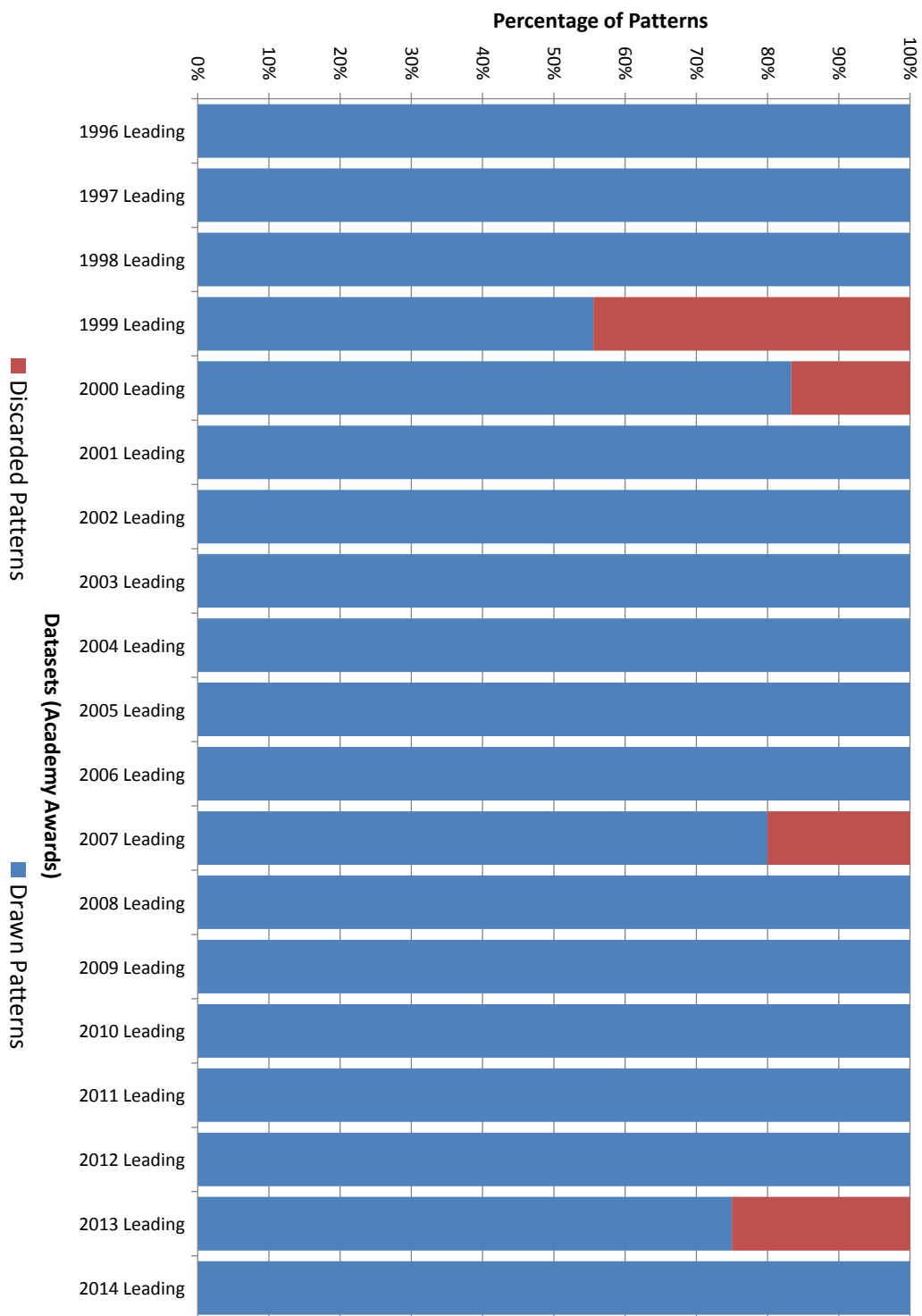


Figure A.30: Percentage of Patterns Drawn & Discarded (Academy Awards) (1)

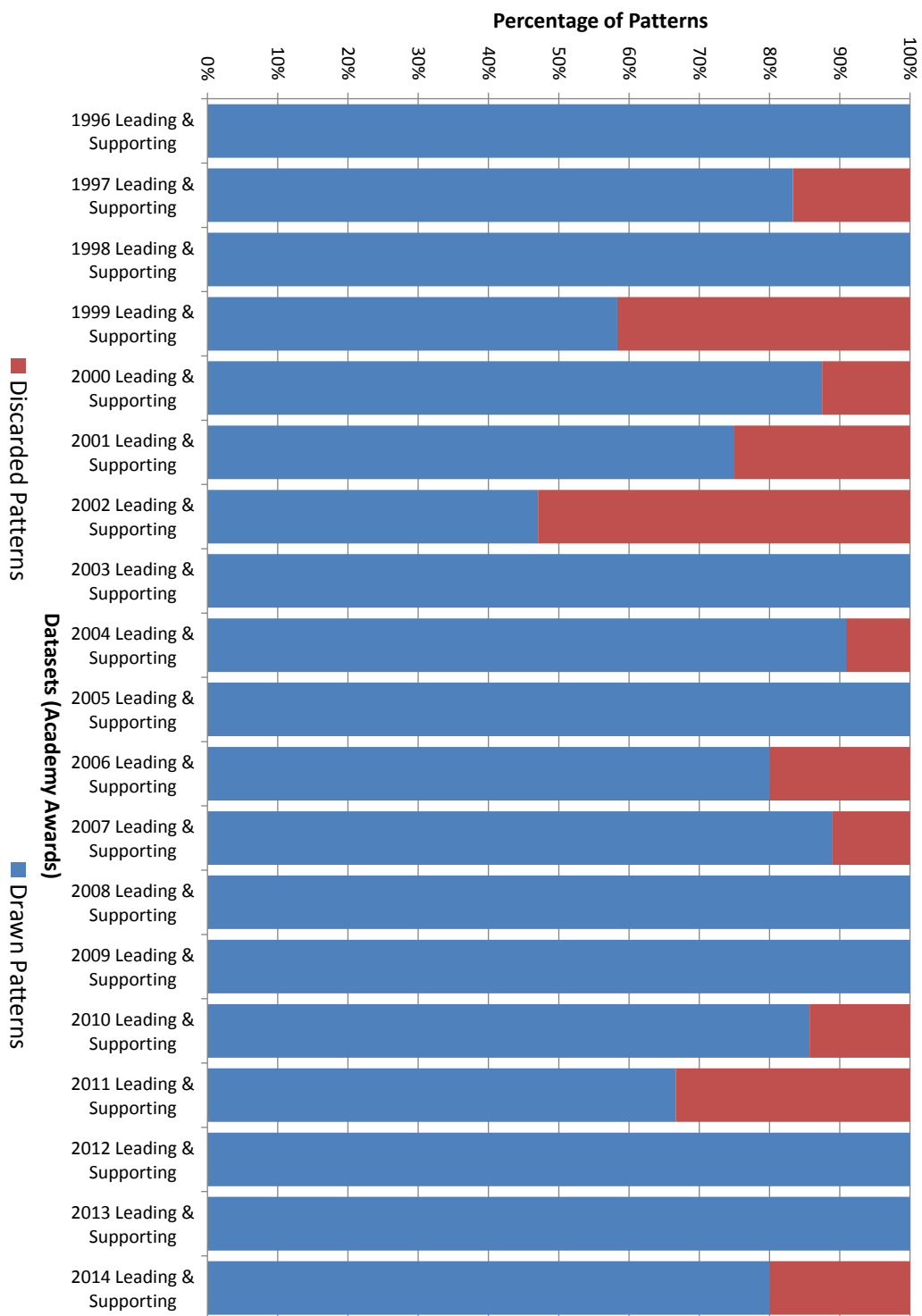


Figure A.31: Percentage of Patterns Drawn & Discarded (Academy Awards) (2)

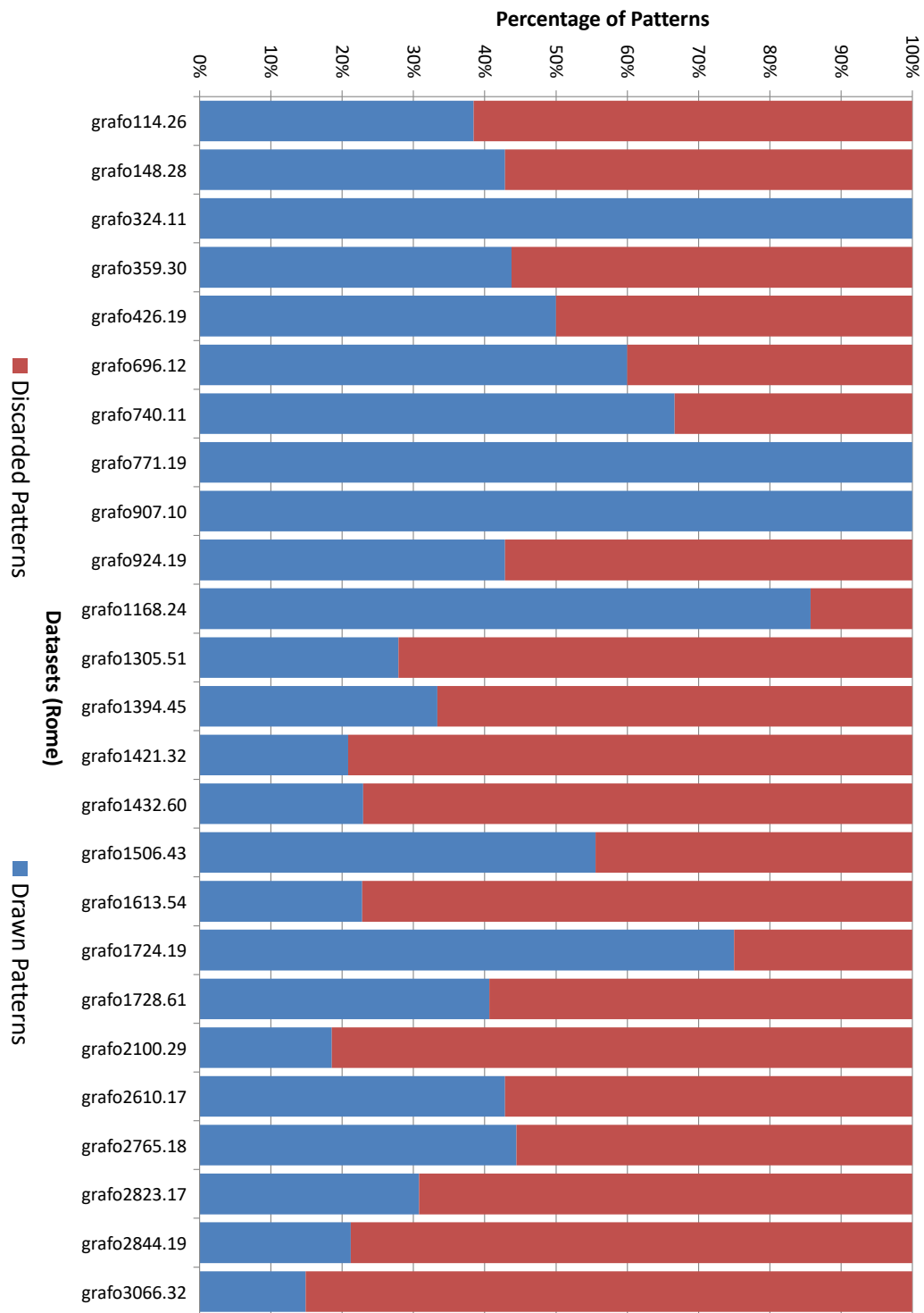


Figure A.32: Percentage of Patterns Drawn & Discarded (Rome) (1)

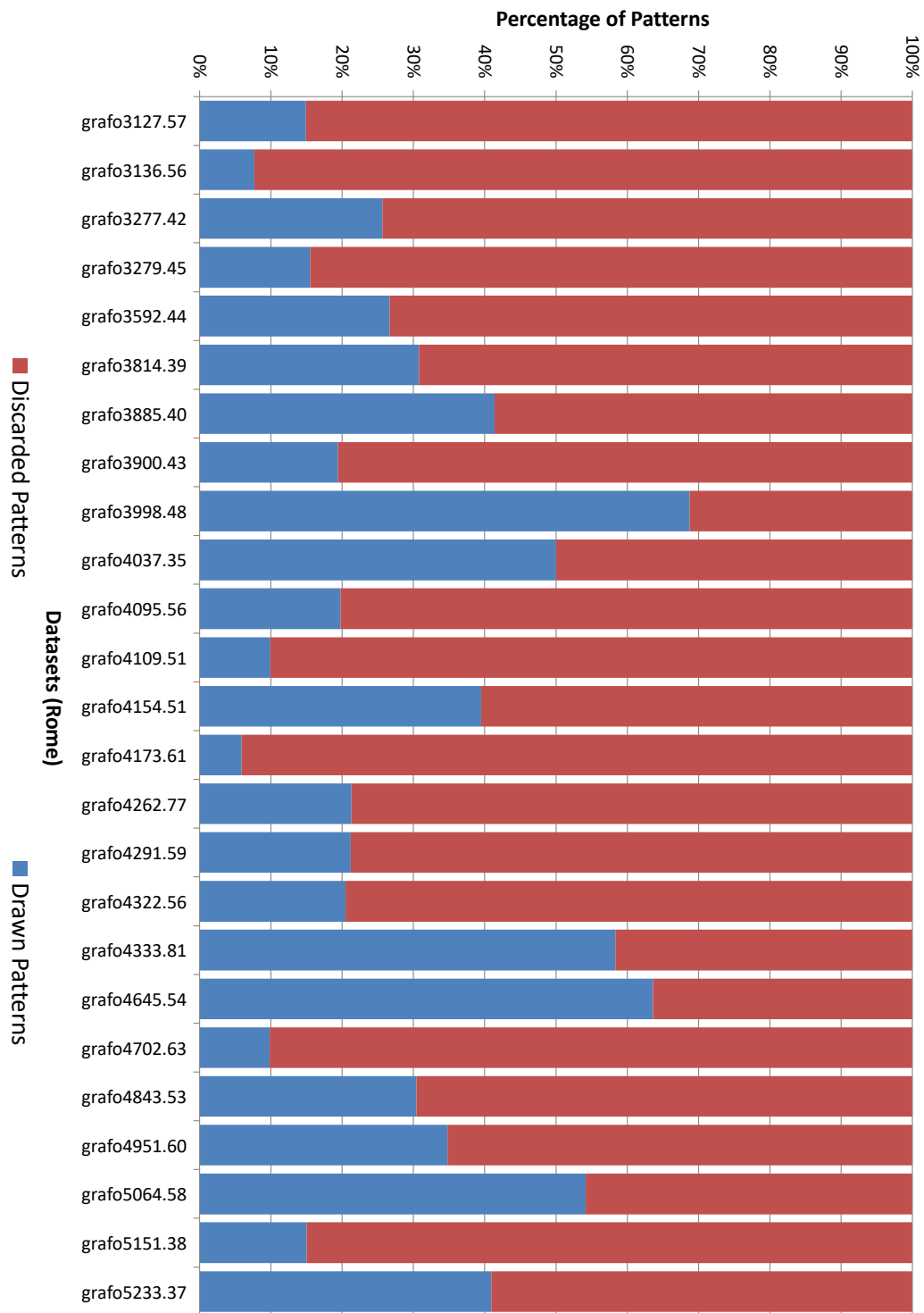


Figure A.33: Percentage of Patterns Drawn & Discarded (Rome) (2)

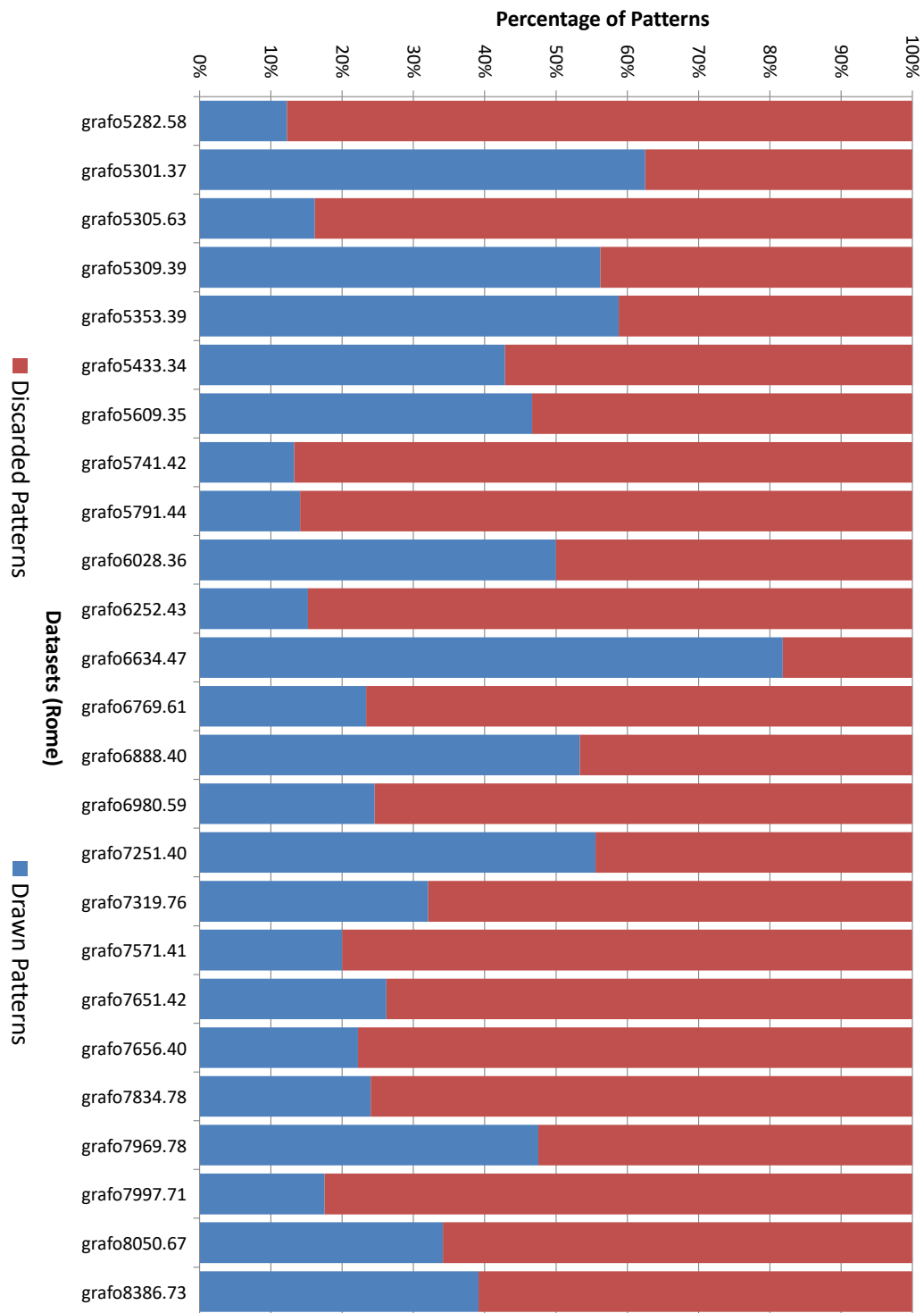


Figure A.34: Percentage of Patterns Drawn & Discarded (Rome) (3)

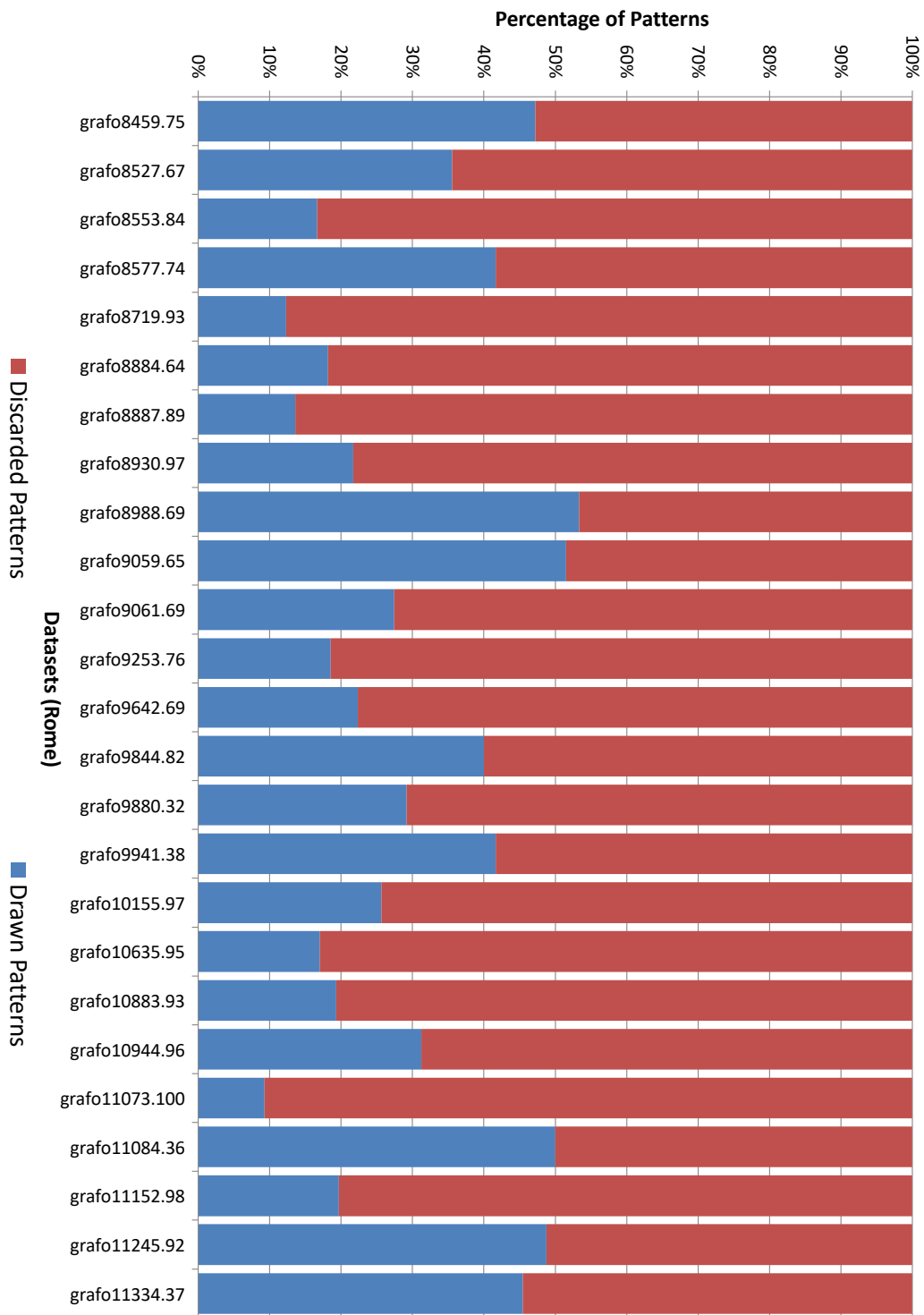


Figure A.35: Percentage of Patterns Drawn & Discarded (Rome) (4)

A.7 Discarded and Drawn Nodes

The following charts display the number of nodes which have been drawn, are only within discarded patterns, or are in no pattern.

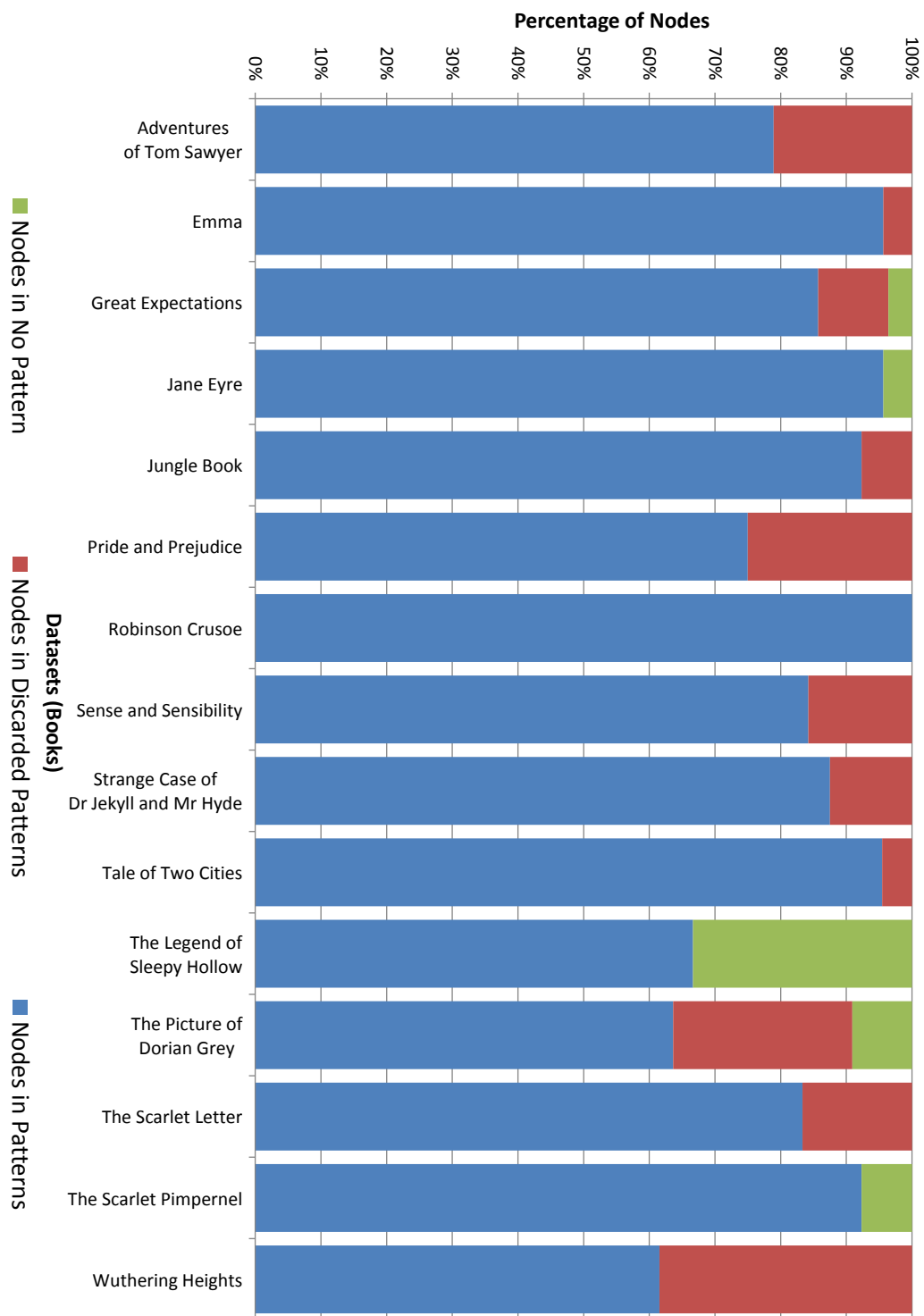


Figure A.36: Percentage of Nodes Drawn, in Discarded Patterns and in No Pattern (Books)

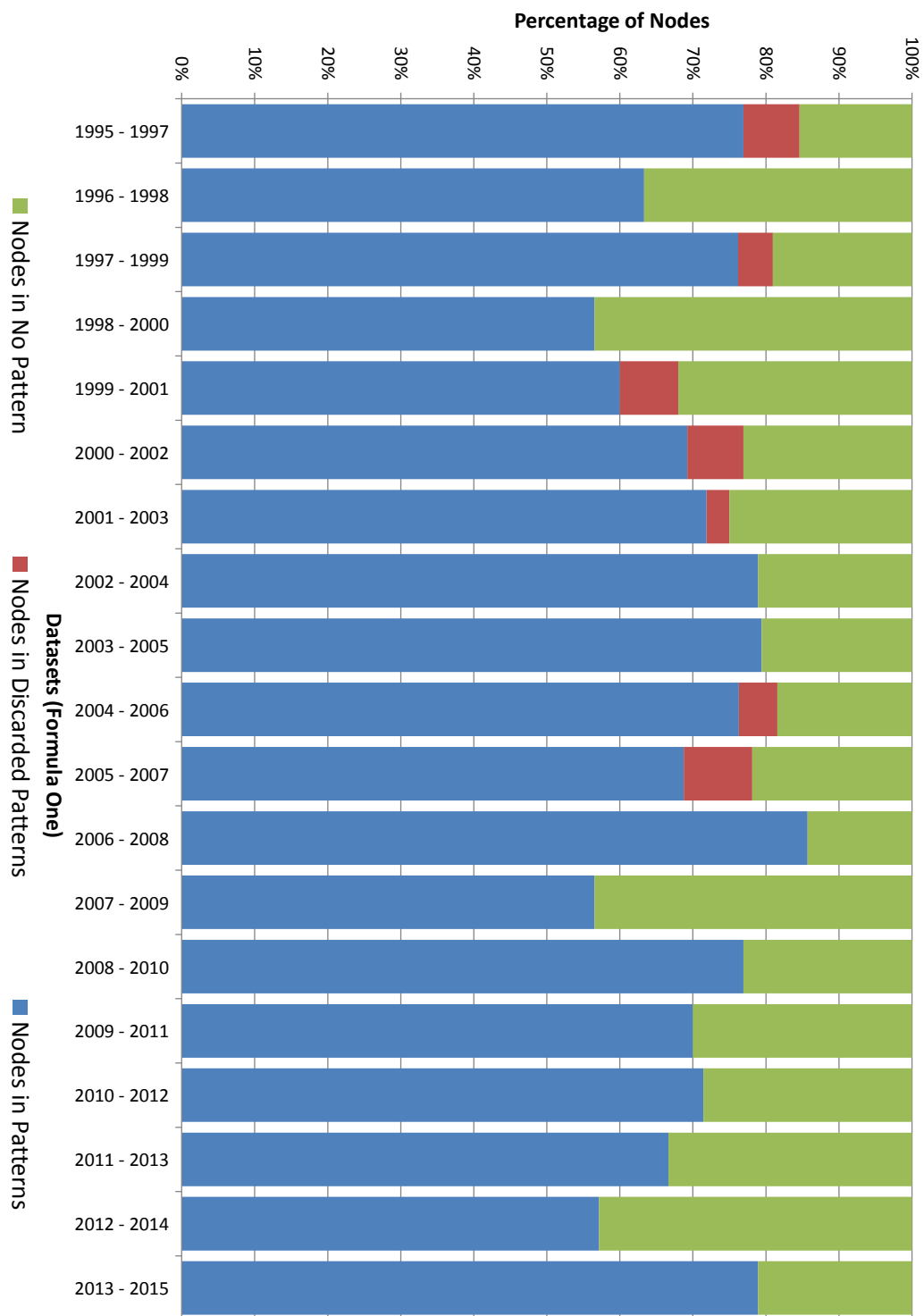


Figure A.37: Percentage of Nodes Drawn, in Discarded Patterns and in No Pattern (Formula One)

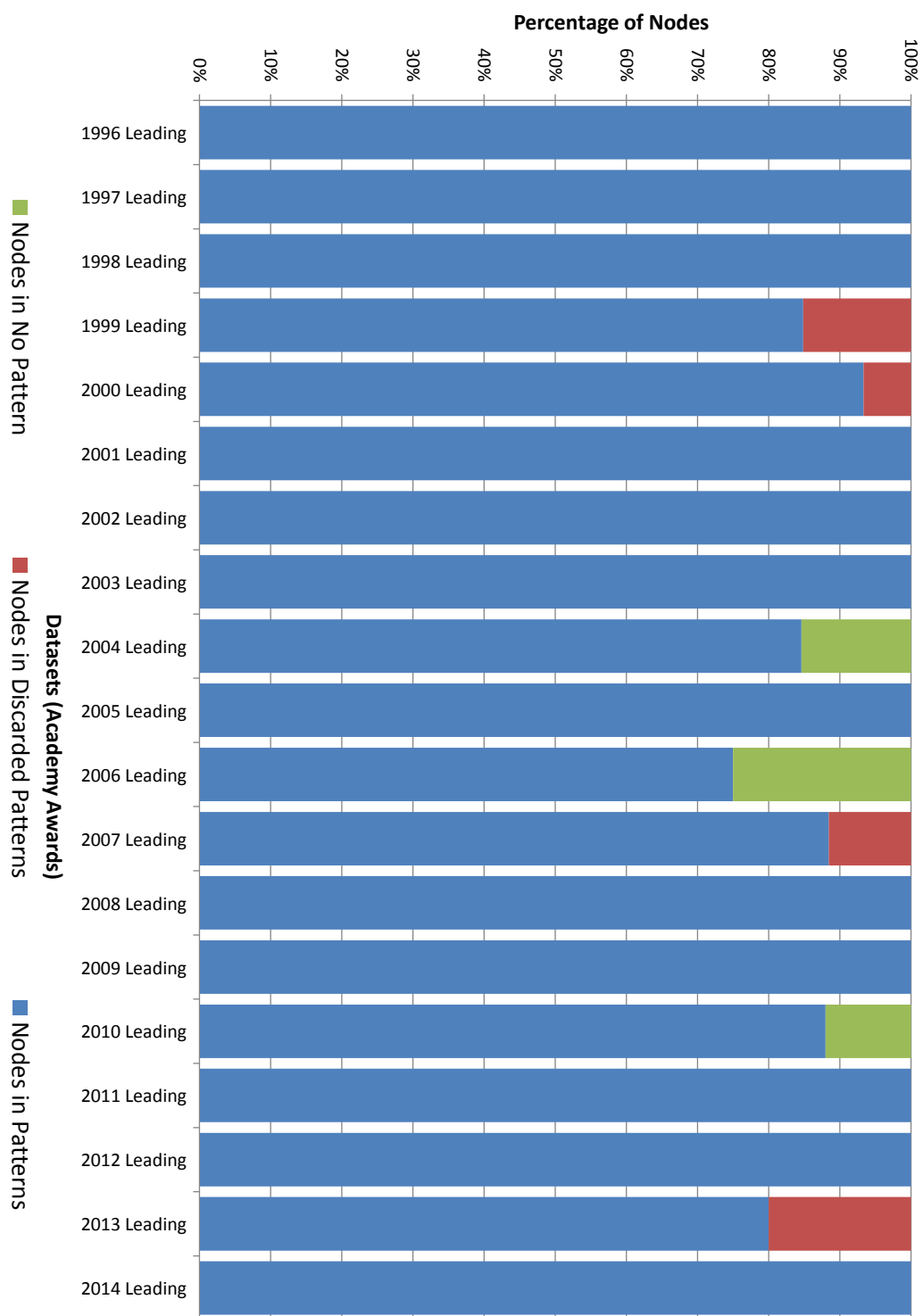


Figure A.38: Percentage of Nodes Drawn, in Discarded Patterns and in No Pattern (Academy Awards) (1)

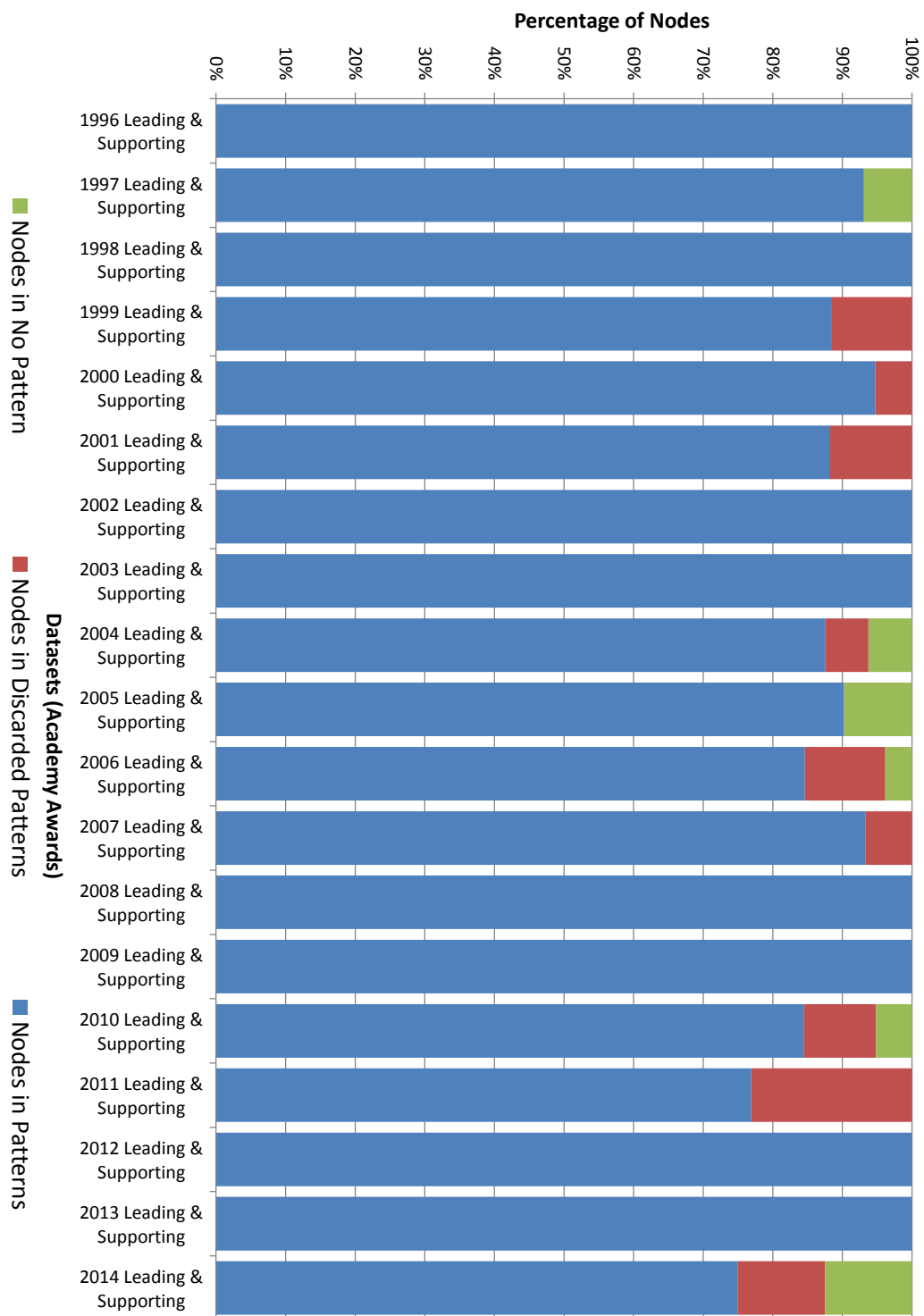


Figure A.39: Percentage of Nodes Drawn, in Discarded Patterns and in No Pattern (Academy Awards) (2)

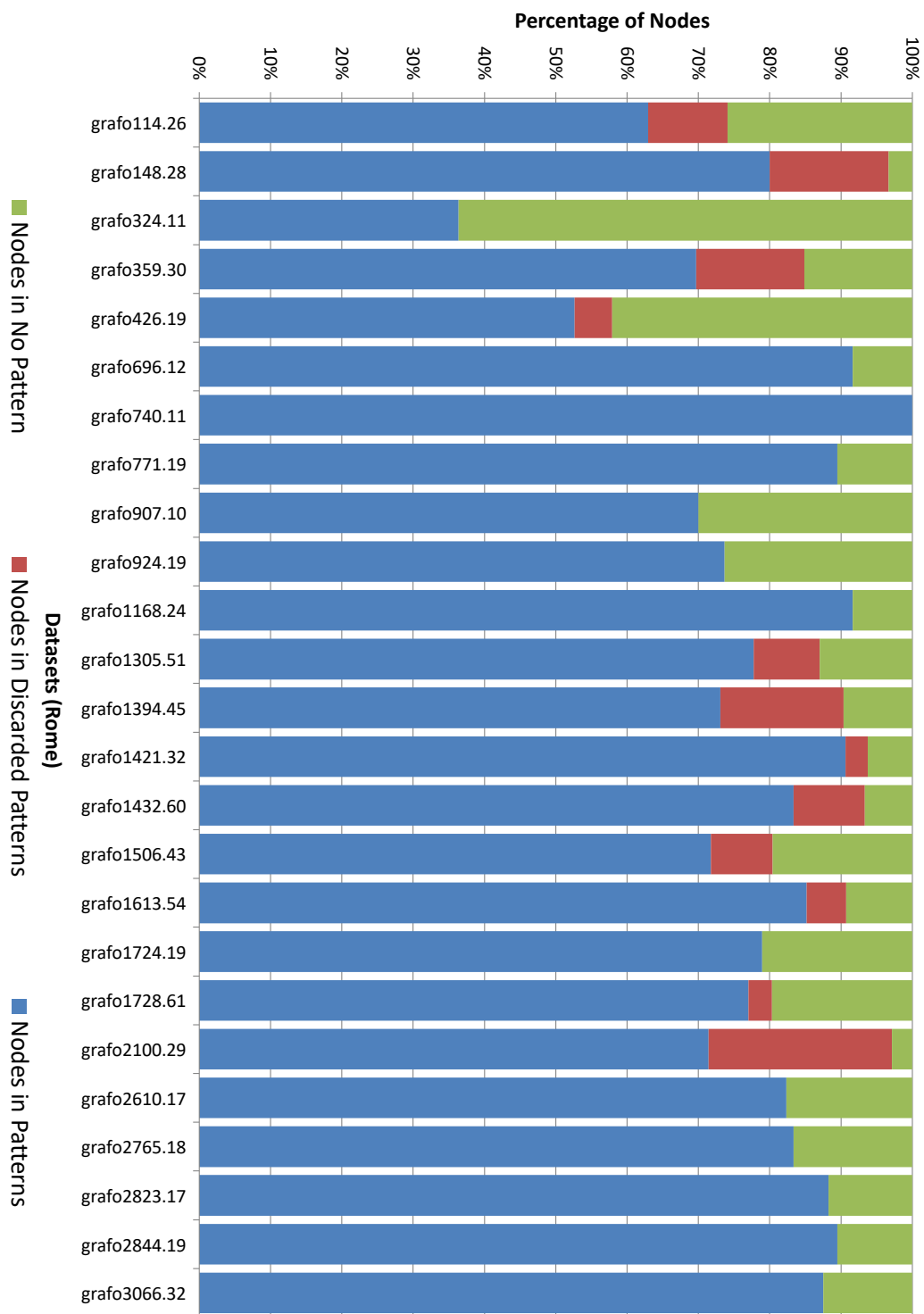


Figure A.40: Percentage of Nodes Drawn, in Discarded Patterns and in No Pattern (Rome) (1)

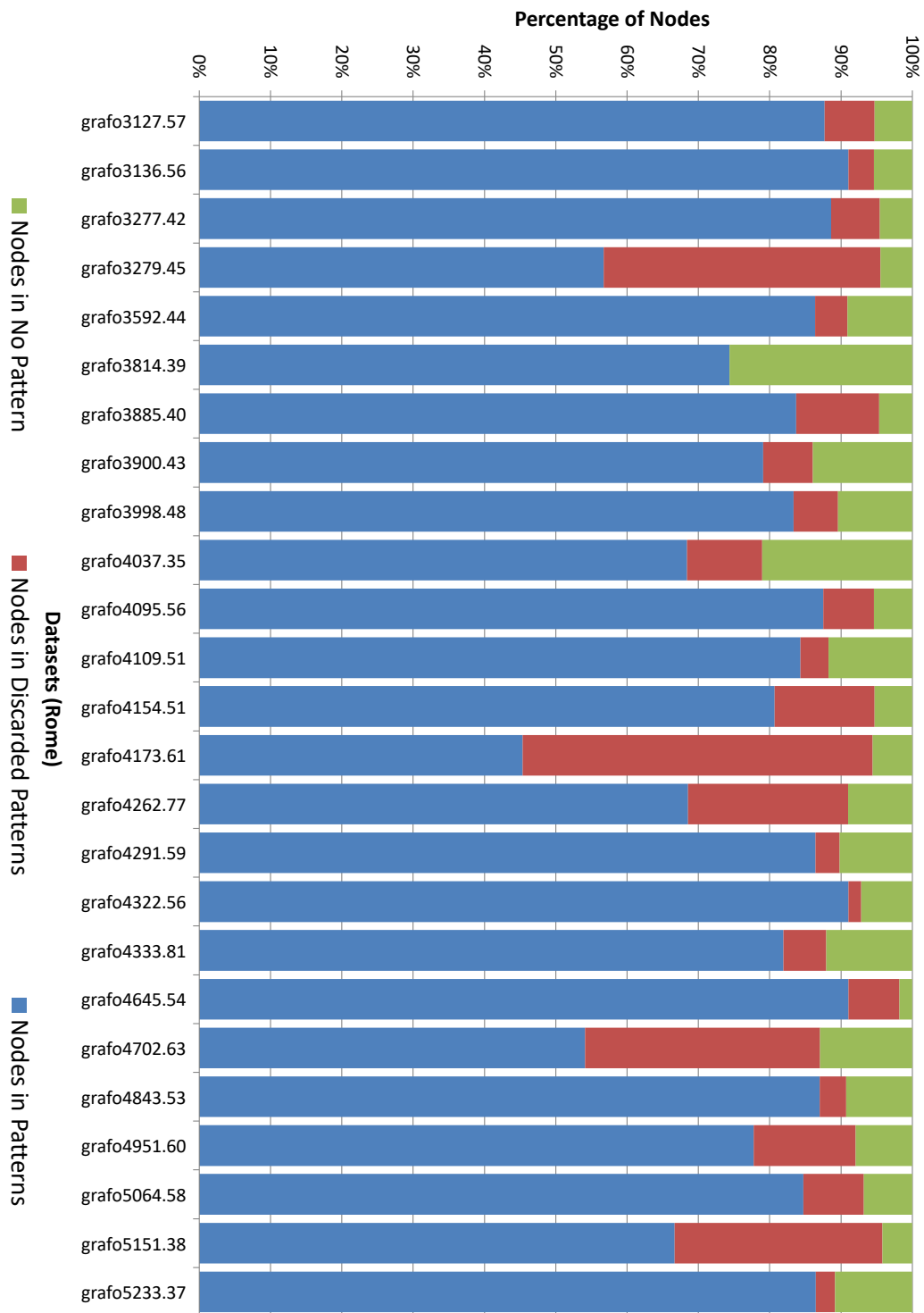


Figure A.41: Percentage of Nodes Drawn, in Discarded Patterns and in No Pattern (Rome) (2)

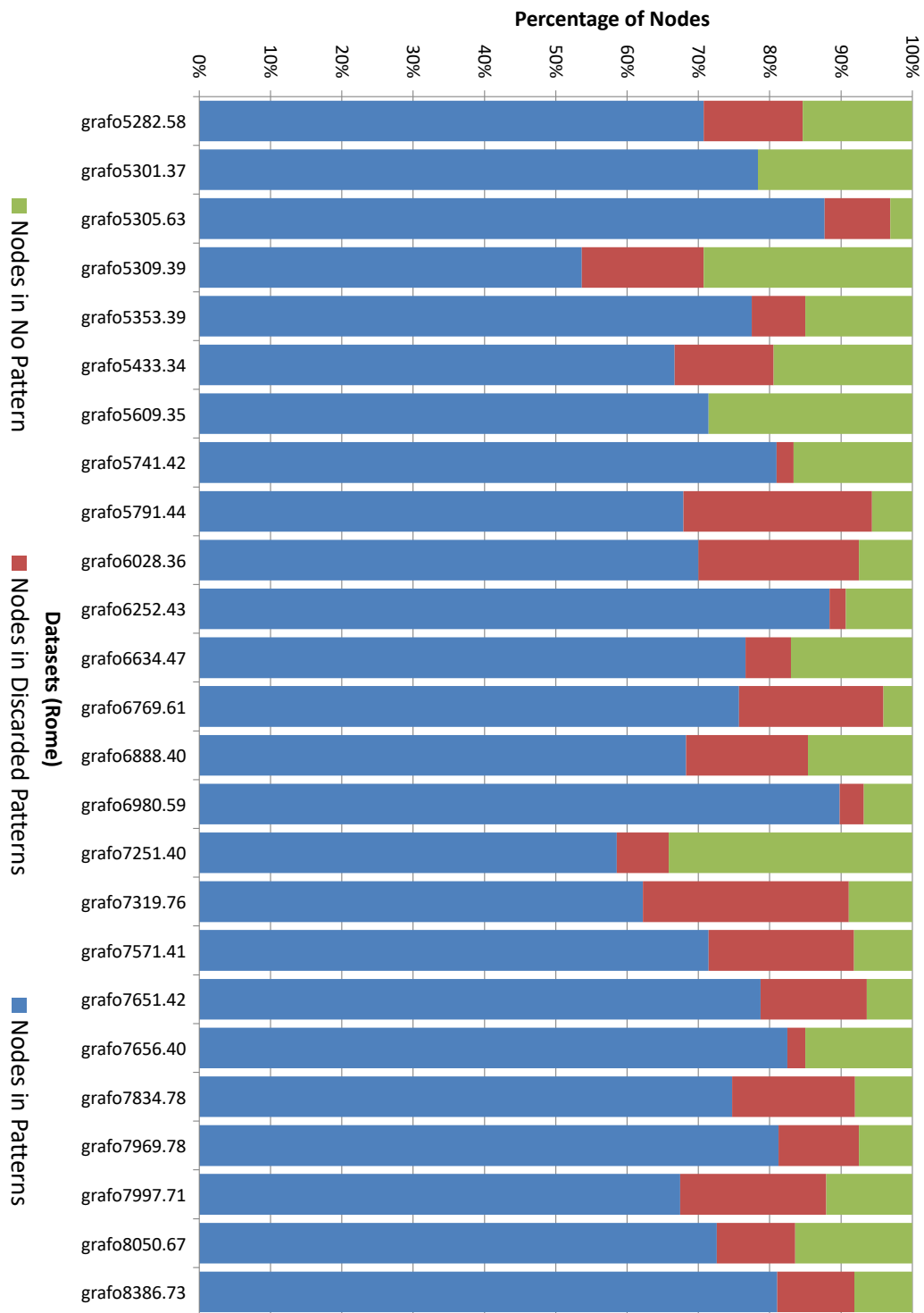


Figure A.42: Percentage of Nodes Drawn, in Discarded Patterns and in No Pattern (Rome) (3)

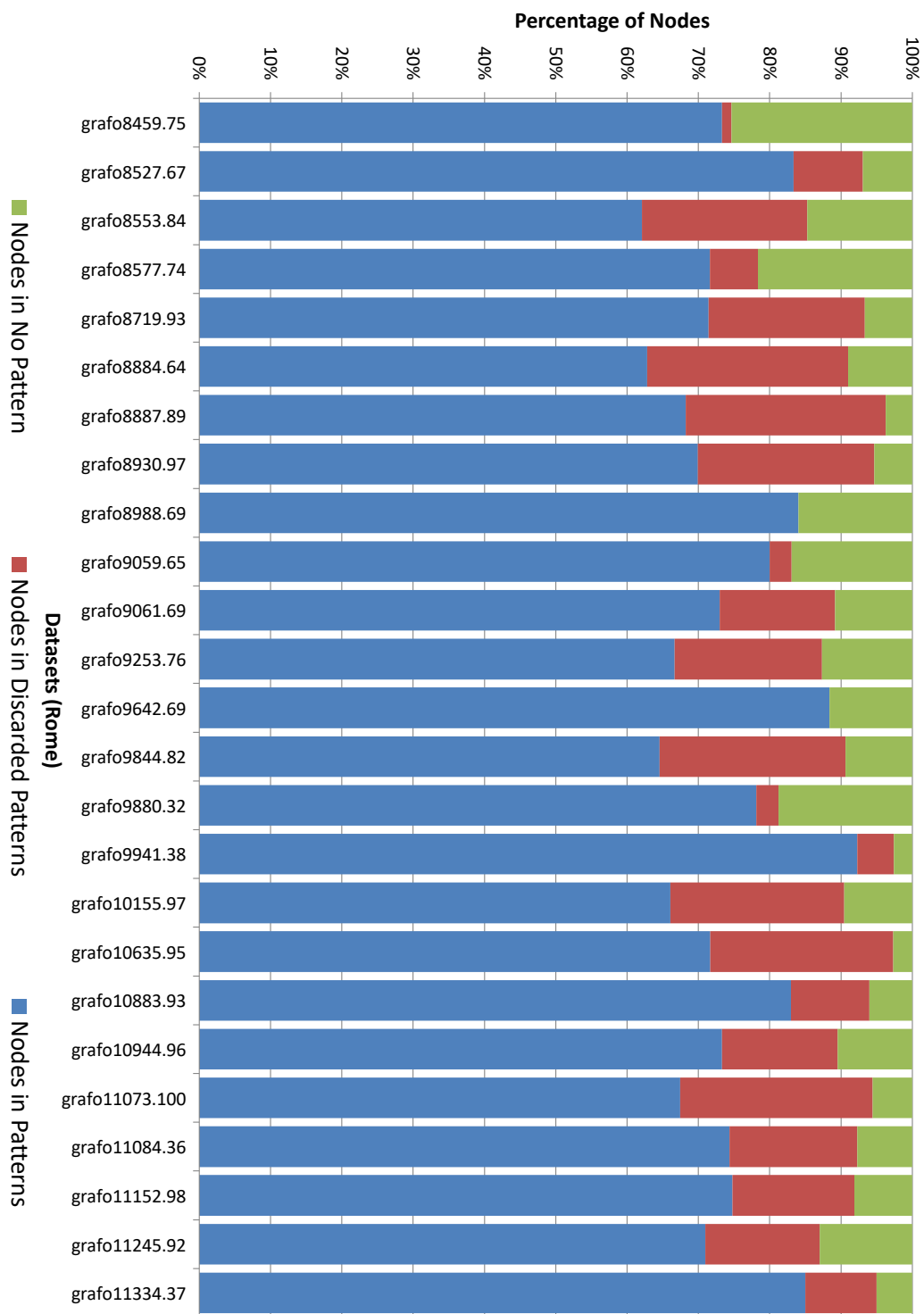


Figure A.43: Percentage of Nodes Drawn, in Discarded Patterns and in No Pattern (Rome) (4)

A.8 Node-node Occlusion

The following charts show the number of nodes who occlude other nodes in the graph, both for the pattern based system, and a force-directed layout. There were no instances of node-node occlusion in the datasets where only the Leading Actor and Actress nominees were selected when drawn by either the force directed method or the pattern based method.

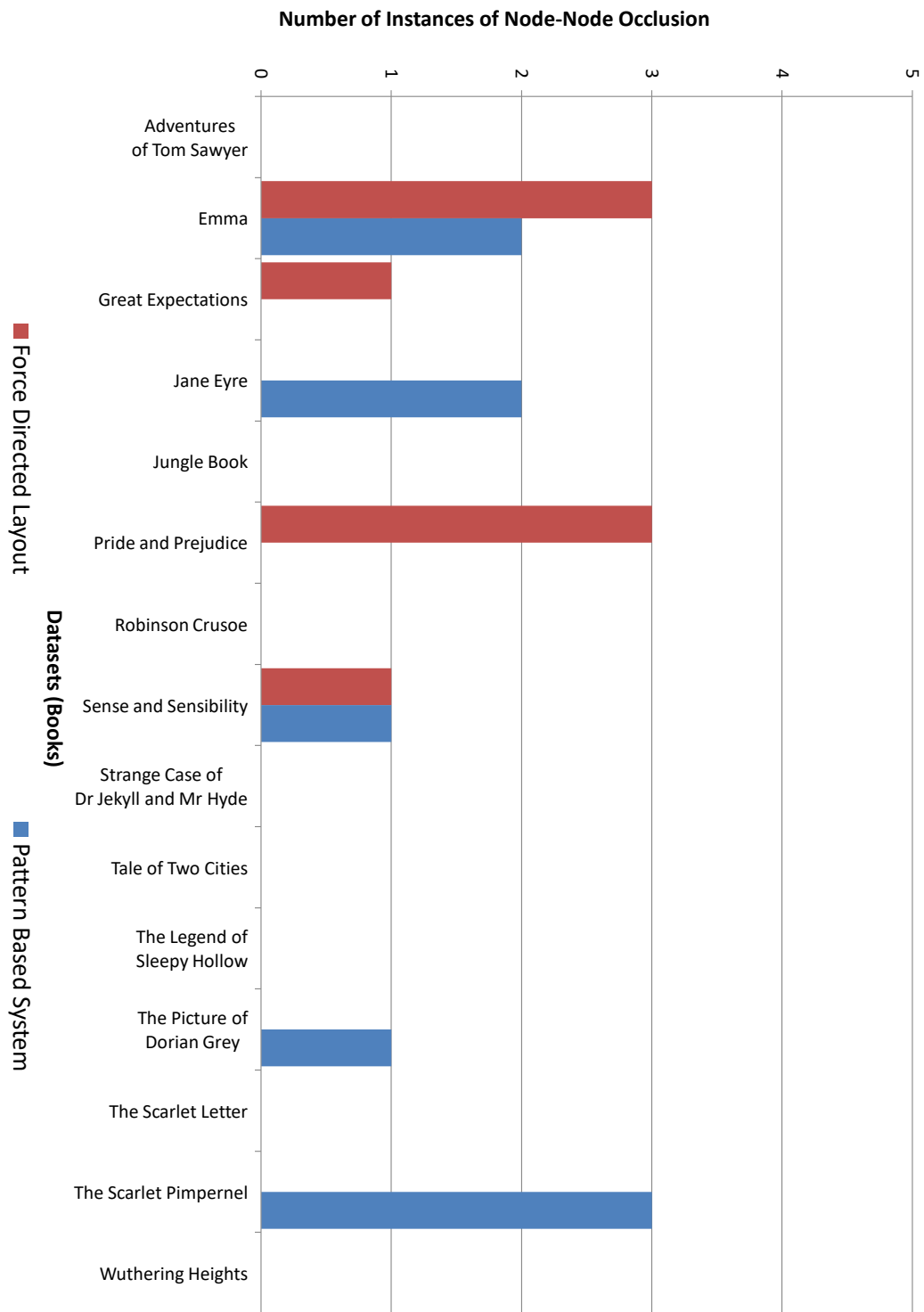


Figure A.44: Instances of Node-Node Occlusion (Books)

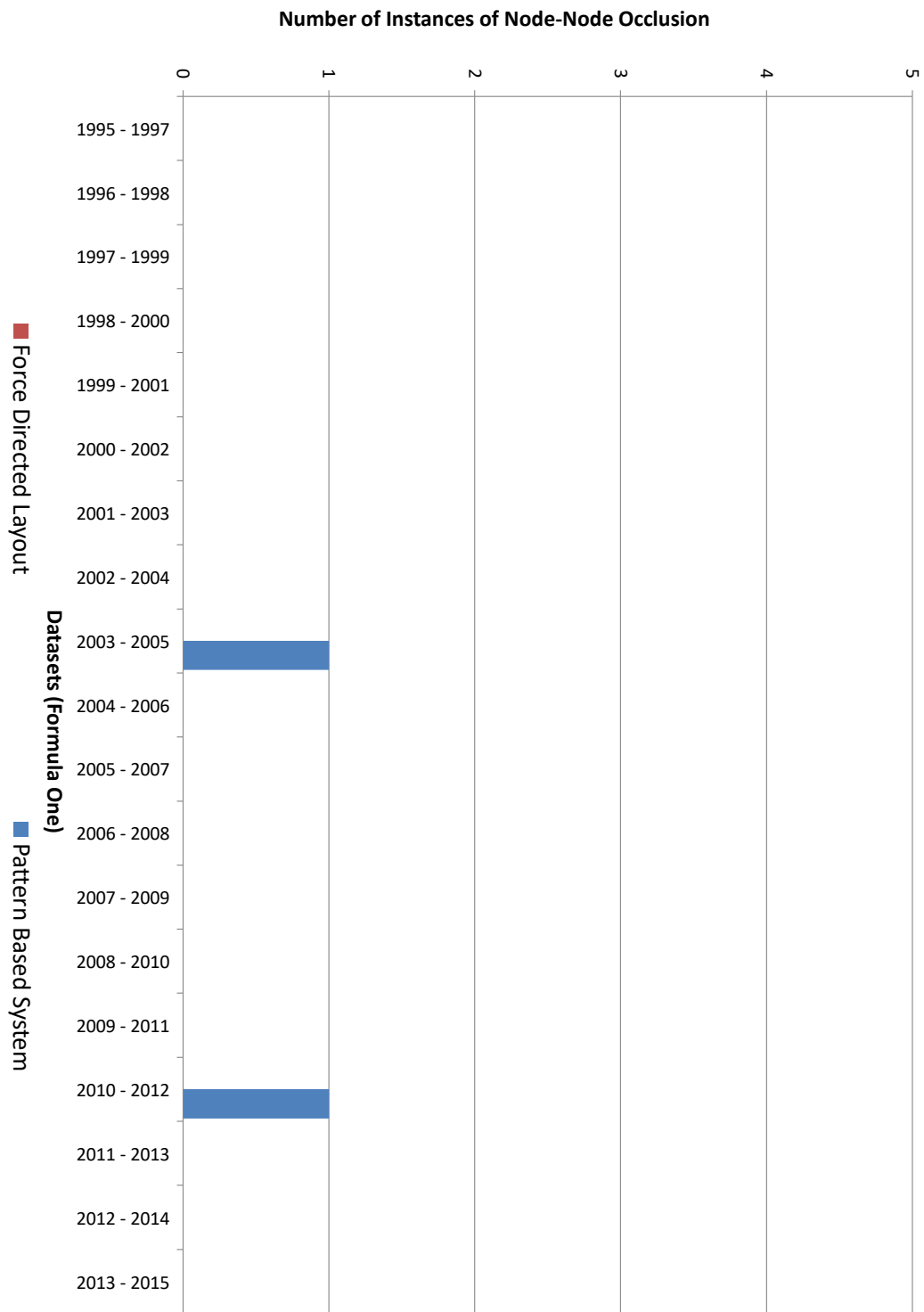


Figure A.45: Instances of Node-Node Occlusion (Formula One)

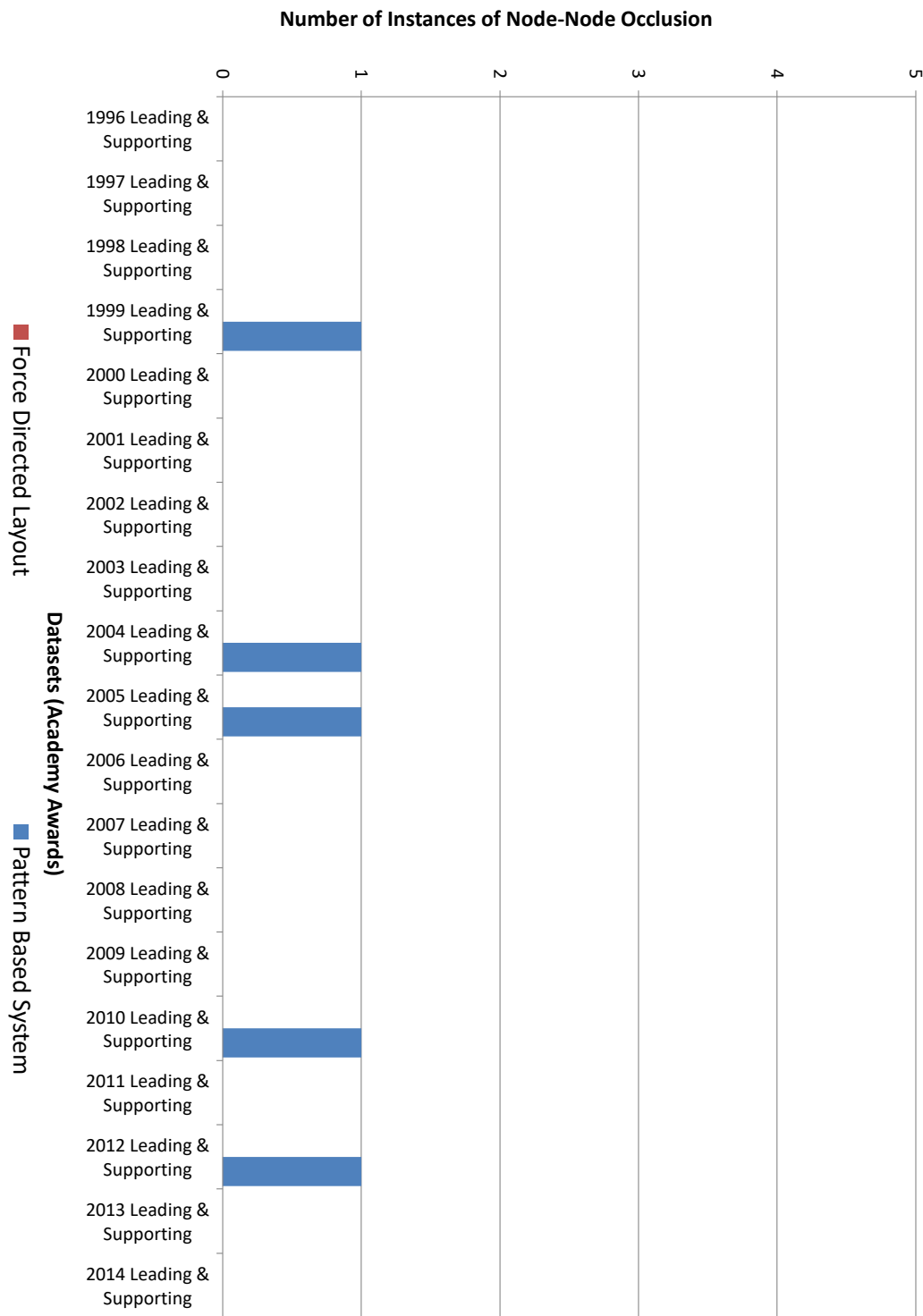


Figure A.46: Instances of Node-Node Occlusion (Academy Awards)

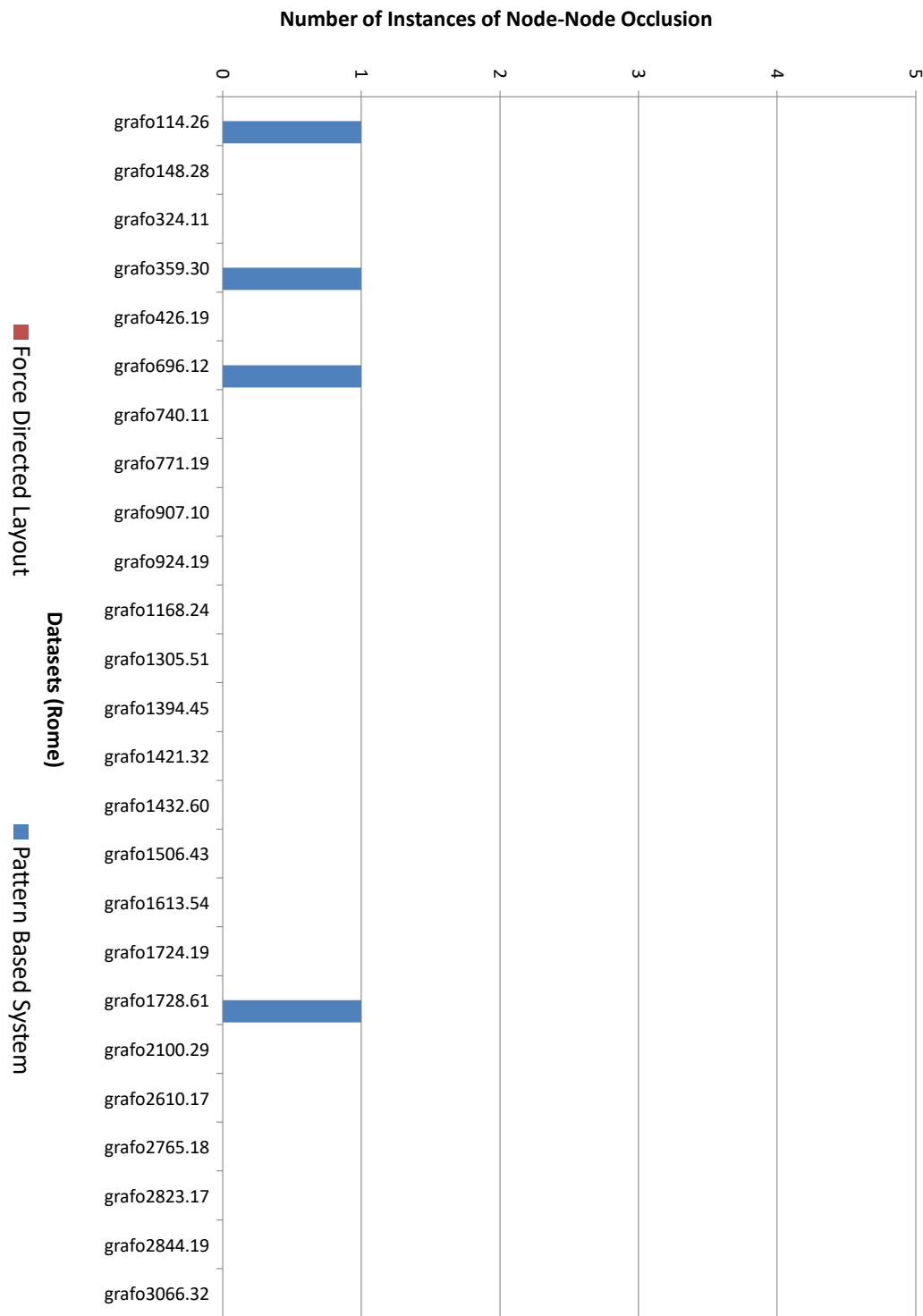


Figure A.47: Instances of Node-Node Occlusion (Rome) (1)

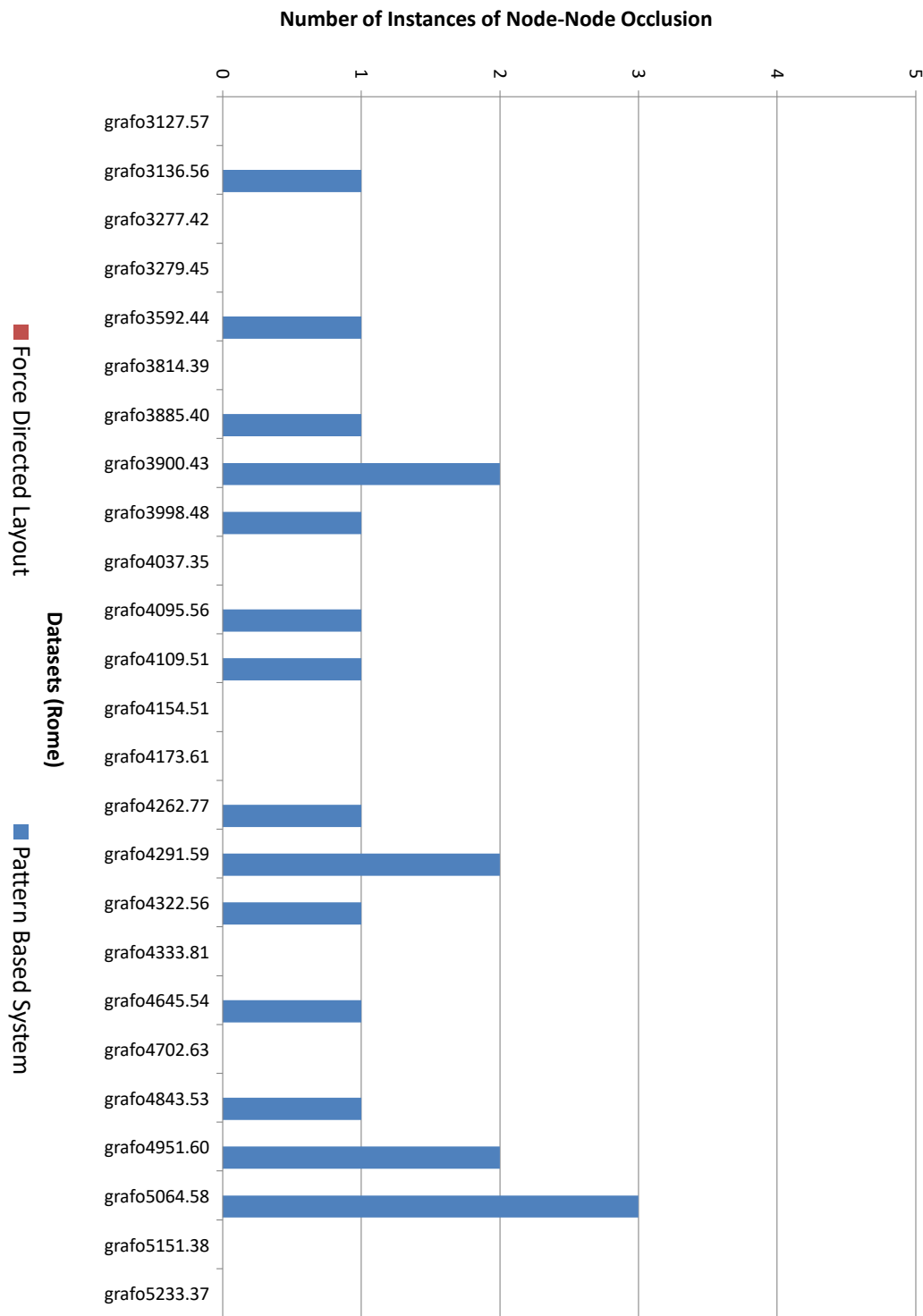


Figure A.48: Instances of Node-Node Occlusion (Rome) (2)

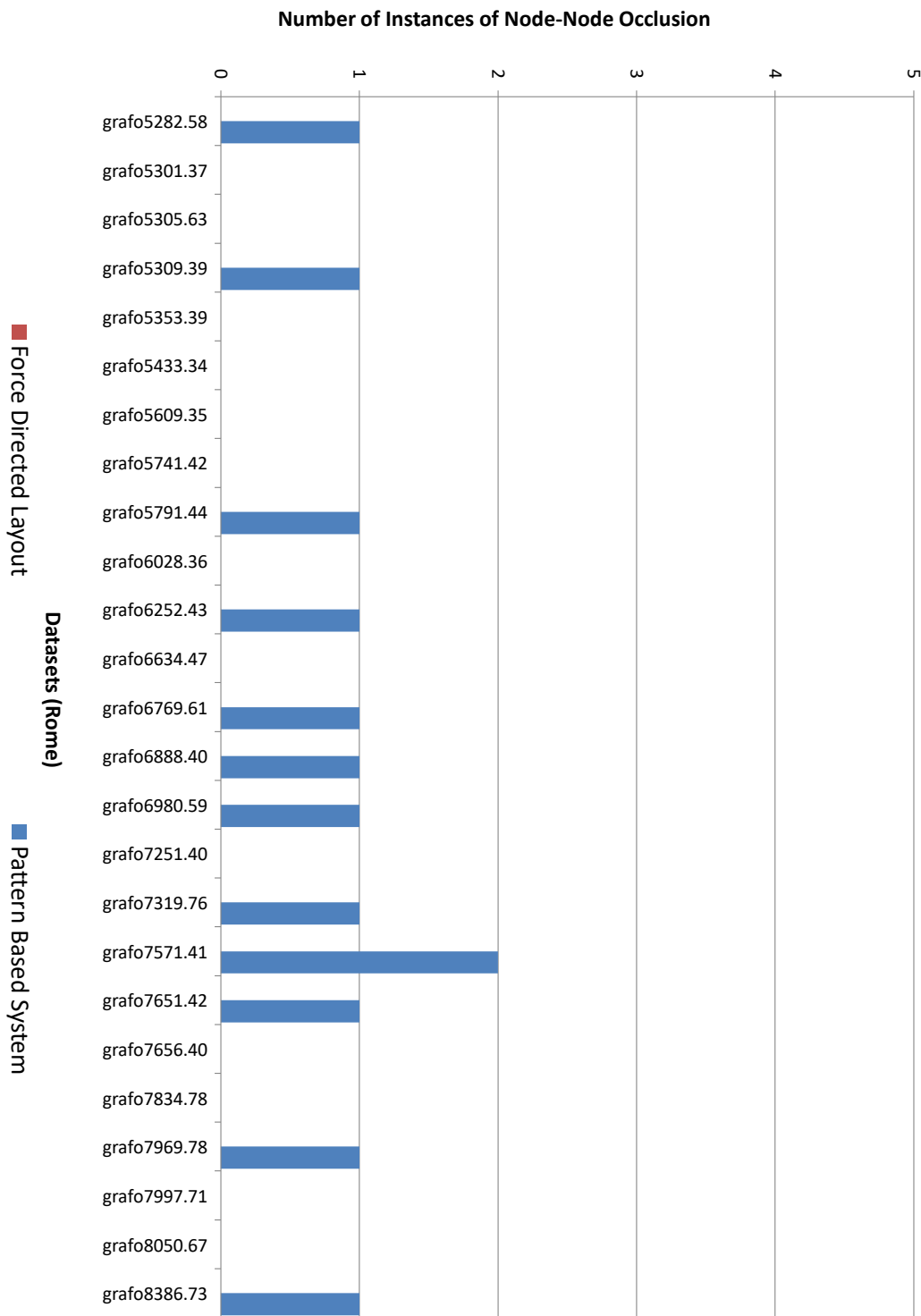


Figure A.49: Instances of Node-Node Occlusion (Rome) (3)

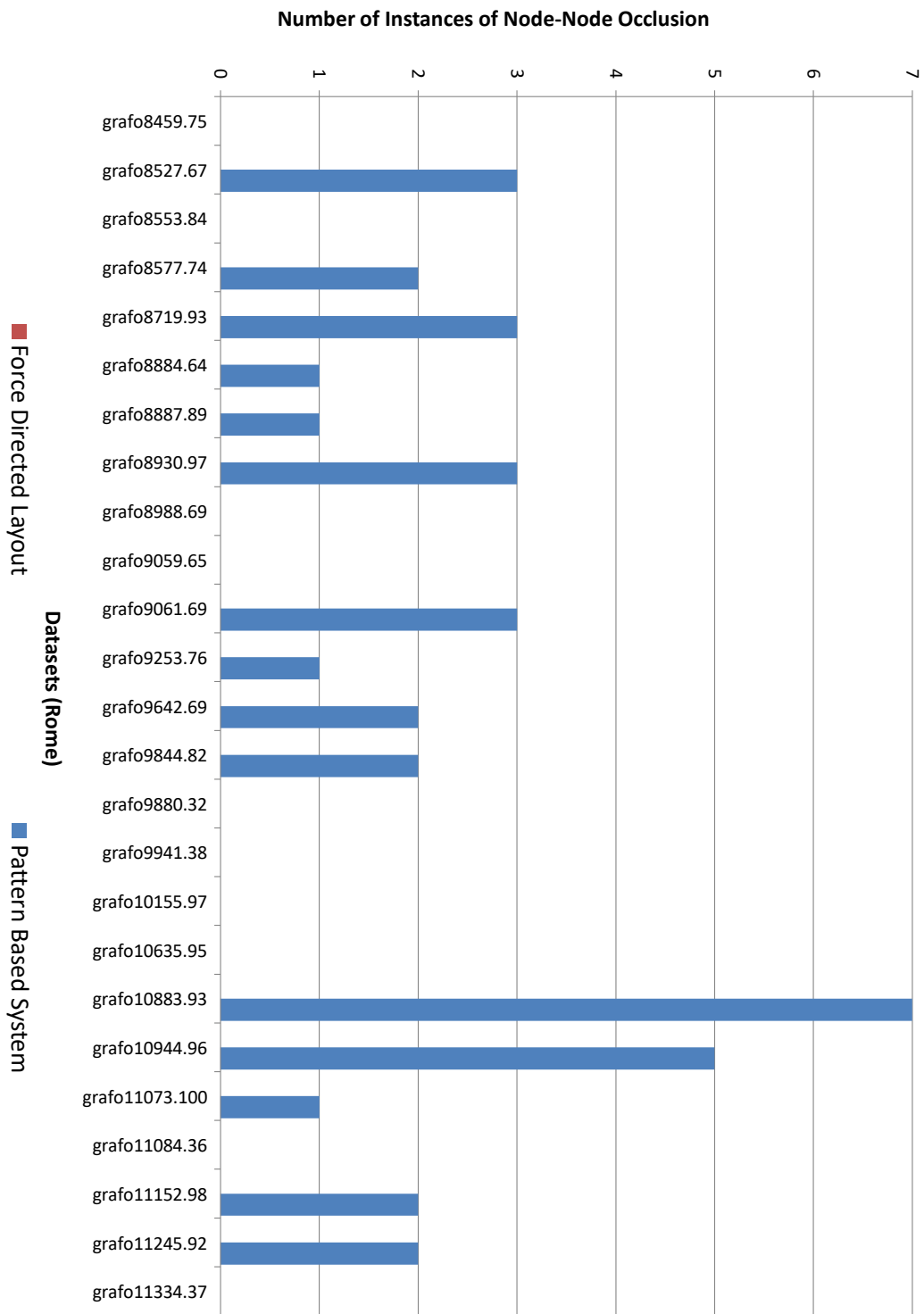


Figure A.50: Instances of Node-Node Occlusion (Rome) (4)

A.9 Node-edge Occlusion

The following charts show the number of nodes who occlude edges in the graph, both for the pattern based system, and a force-directed layout.

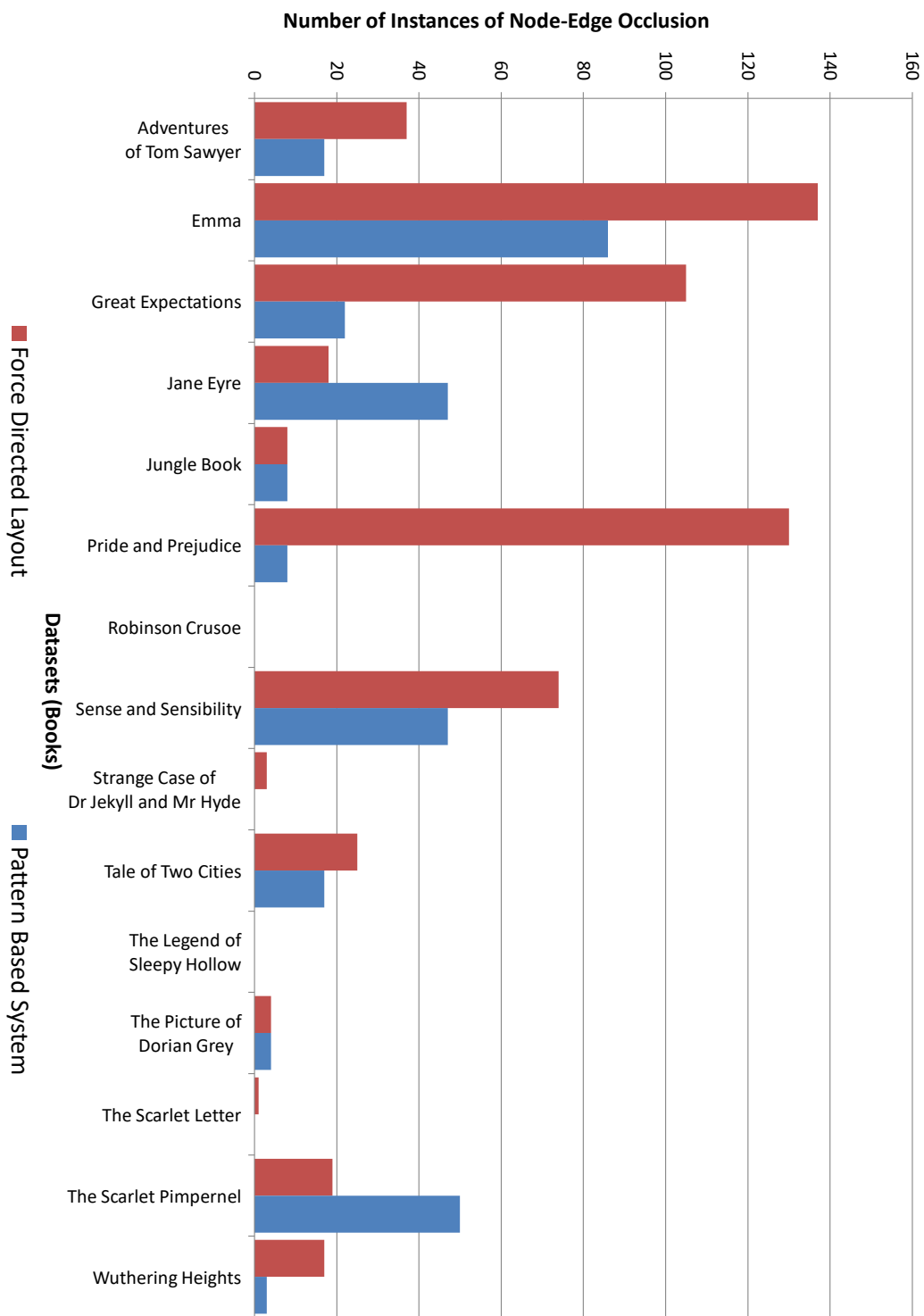


Figure A.51: Instances of Node-Edge Occlusion (Books)

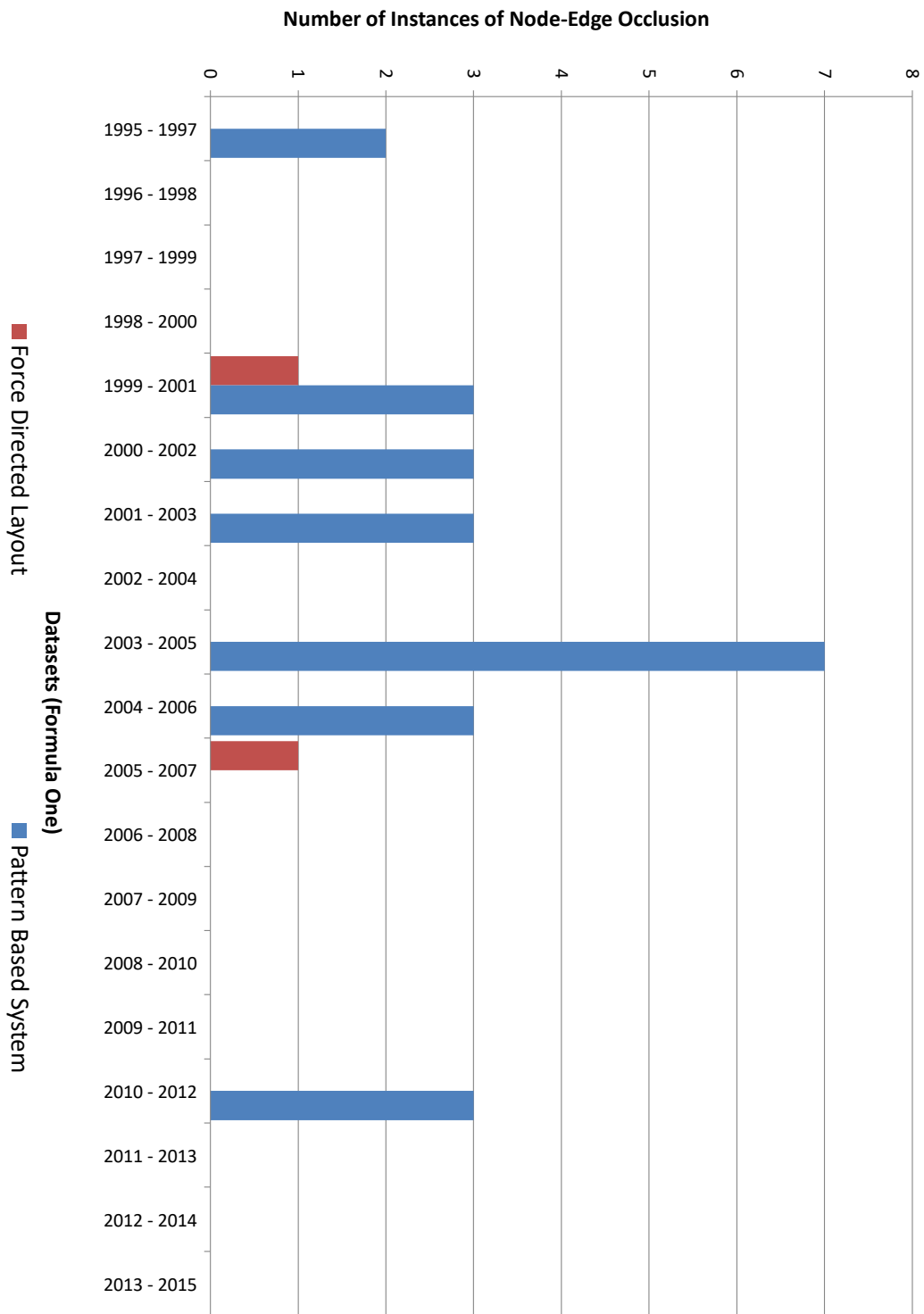


Figure A.52: Instances of Node-Edge Occlusion (Formula One)

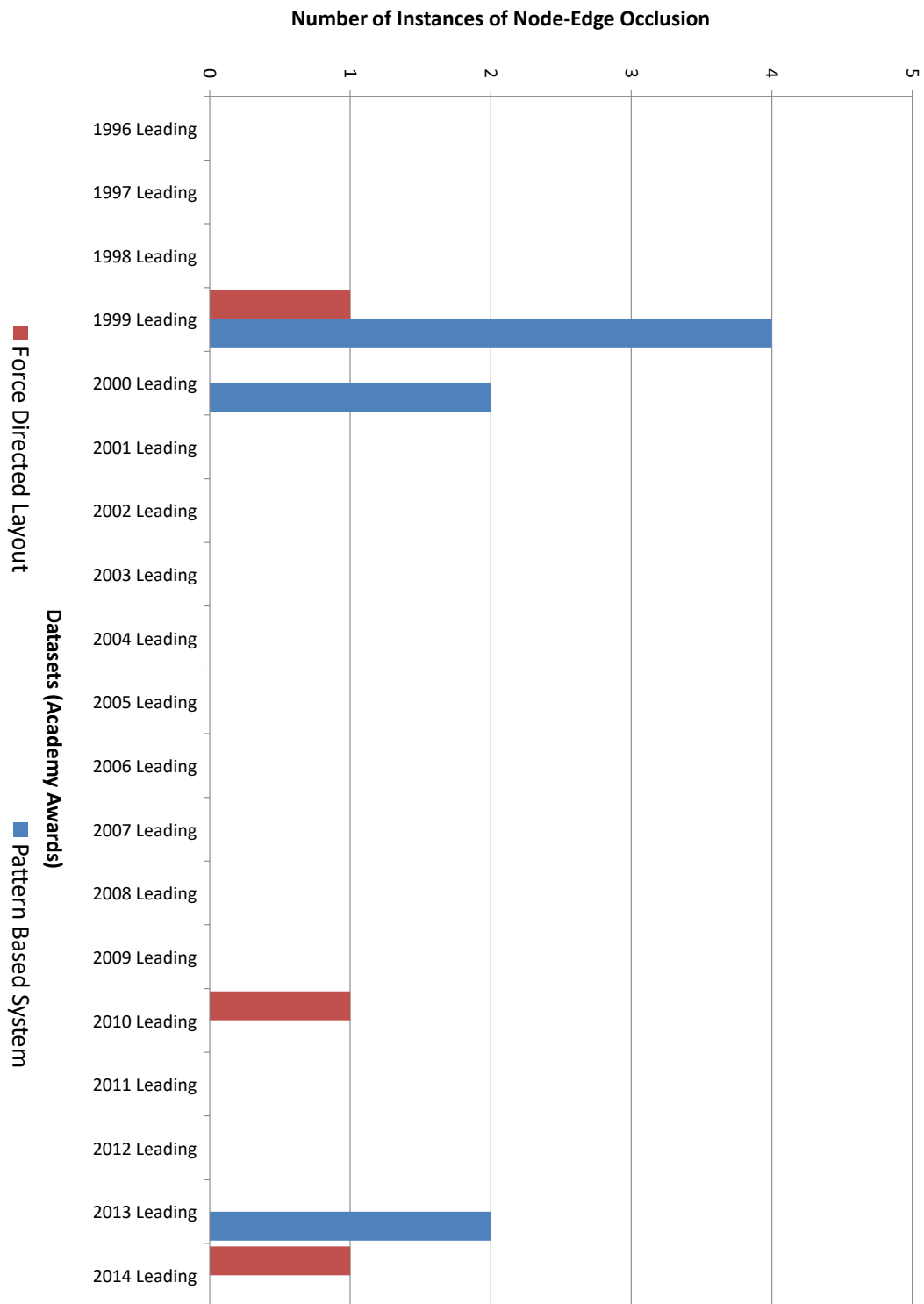


Figure A.53: Instances of Node-Edge Occlusion (Academy Awards) (1)

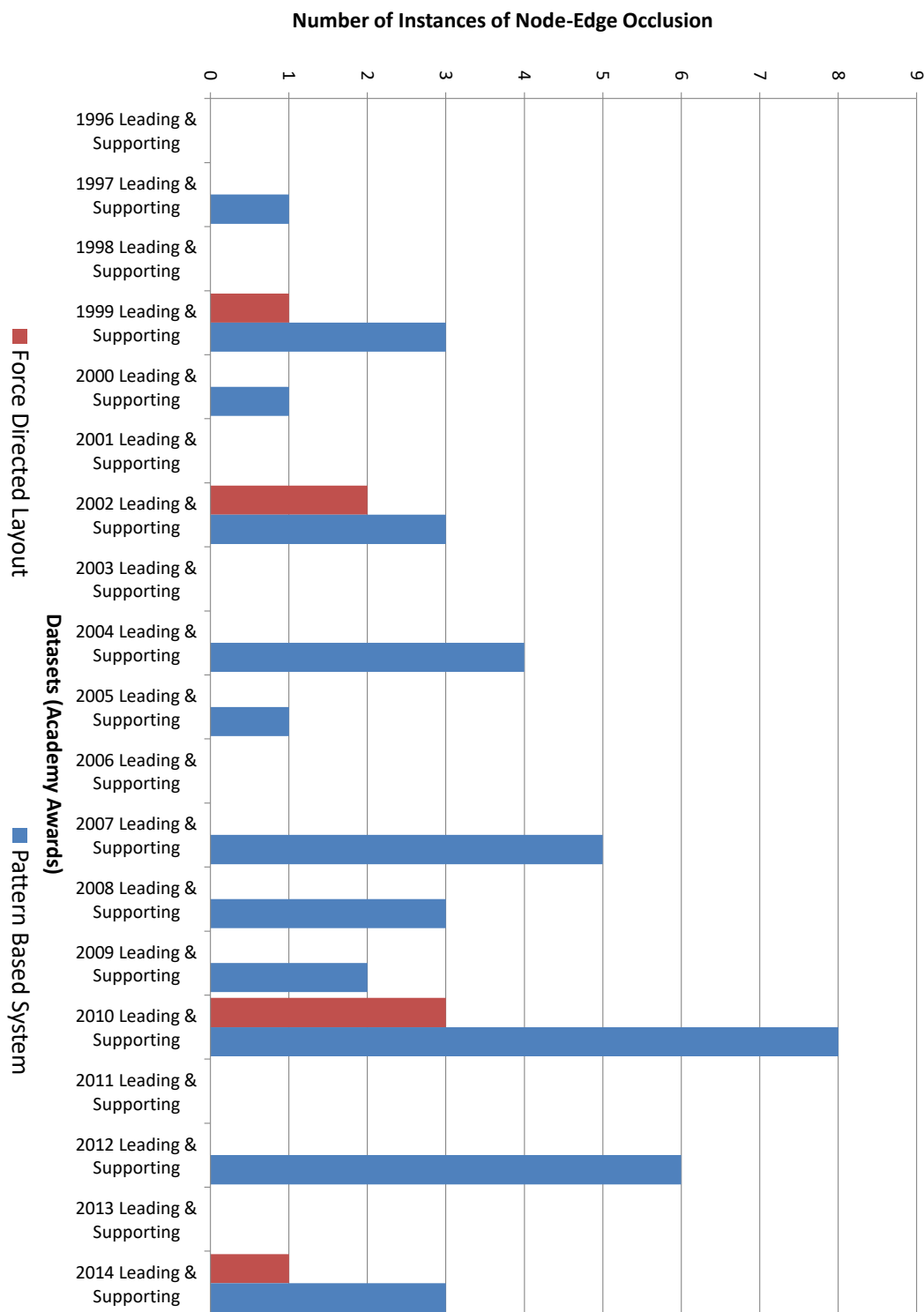


Figure A.54: Instances of Node-Edge Occlusion (Academy Awards) (2)

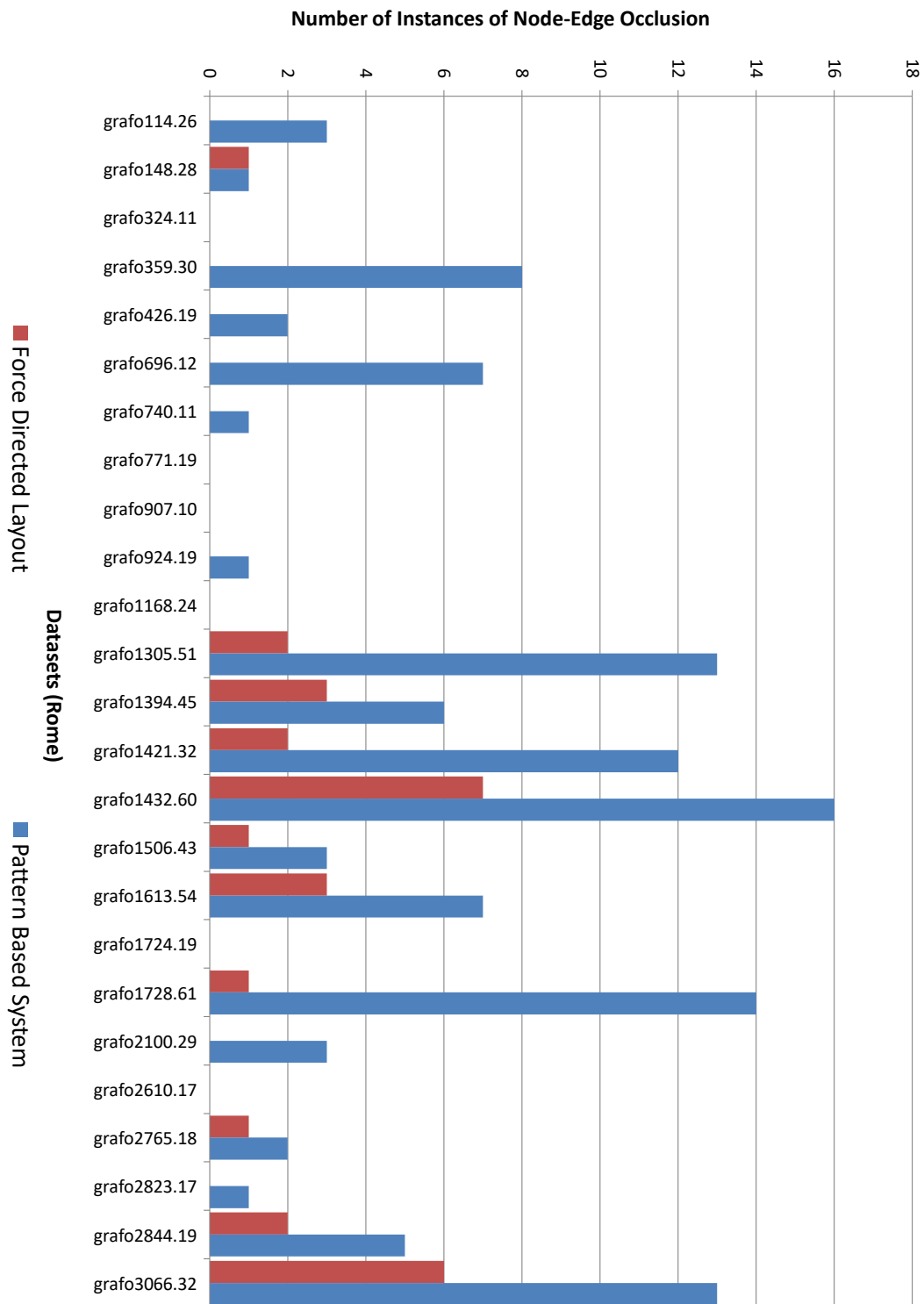


Figure A.55: Instances of Node-Edge Occlusion (Rome) (1)

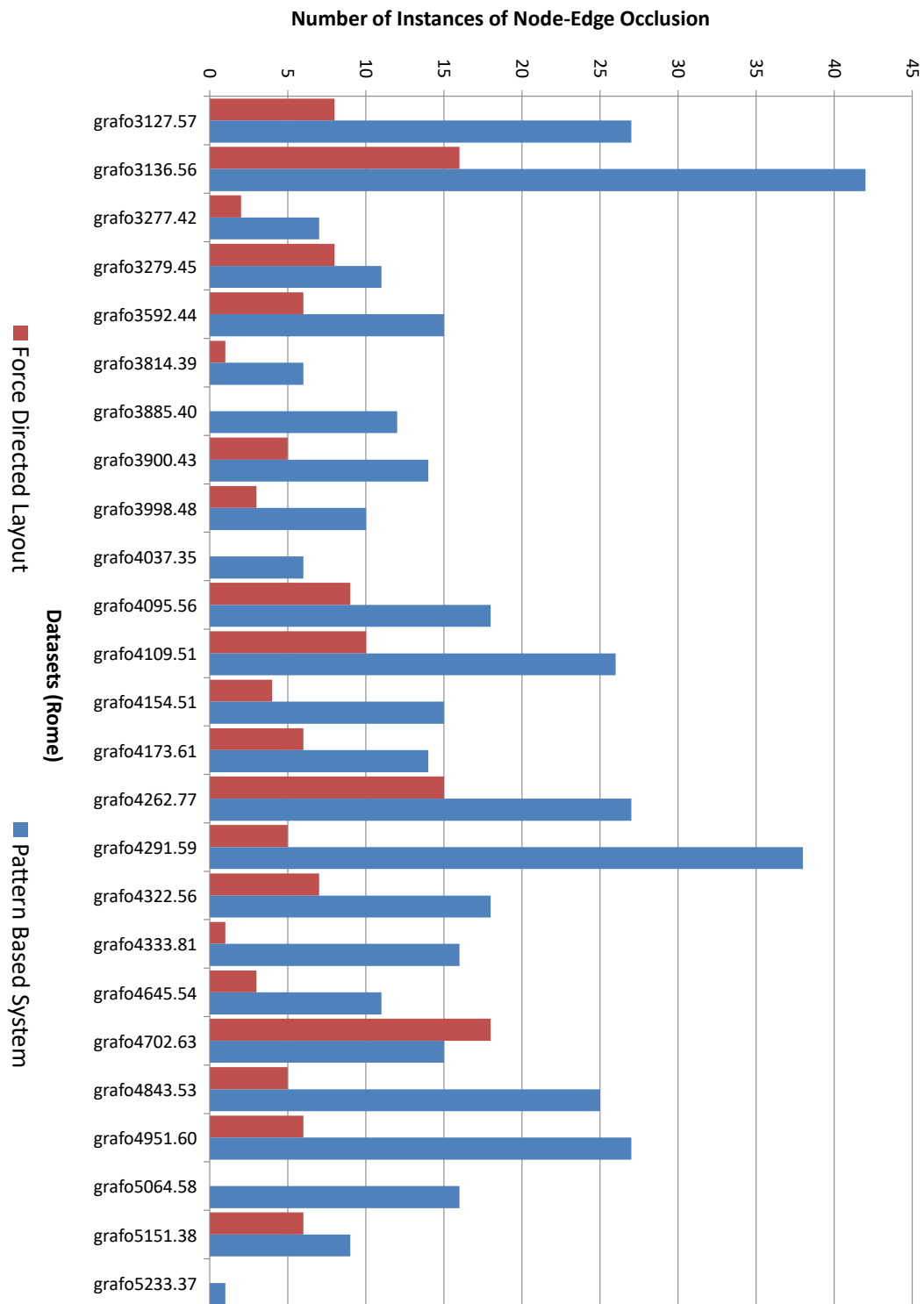


Figure A.56: Instances of Node-Edge Occlusion (Rome) (2)

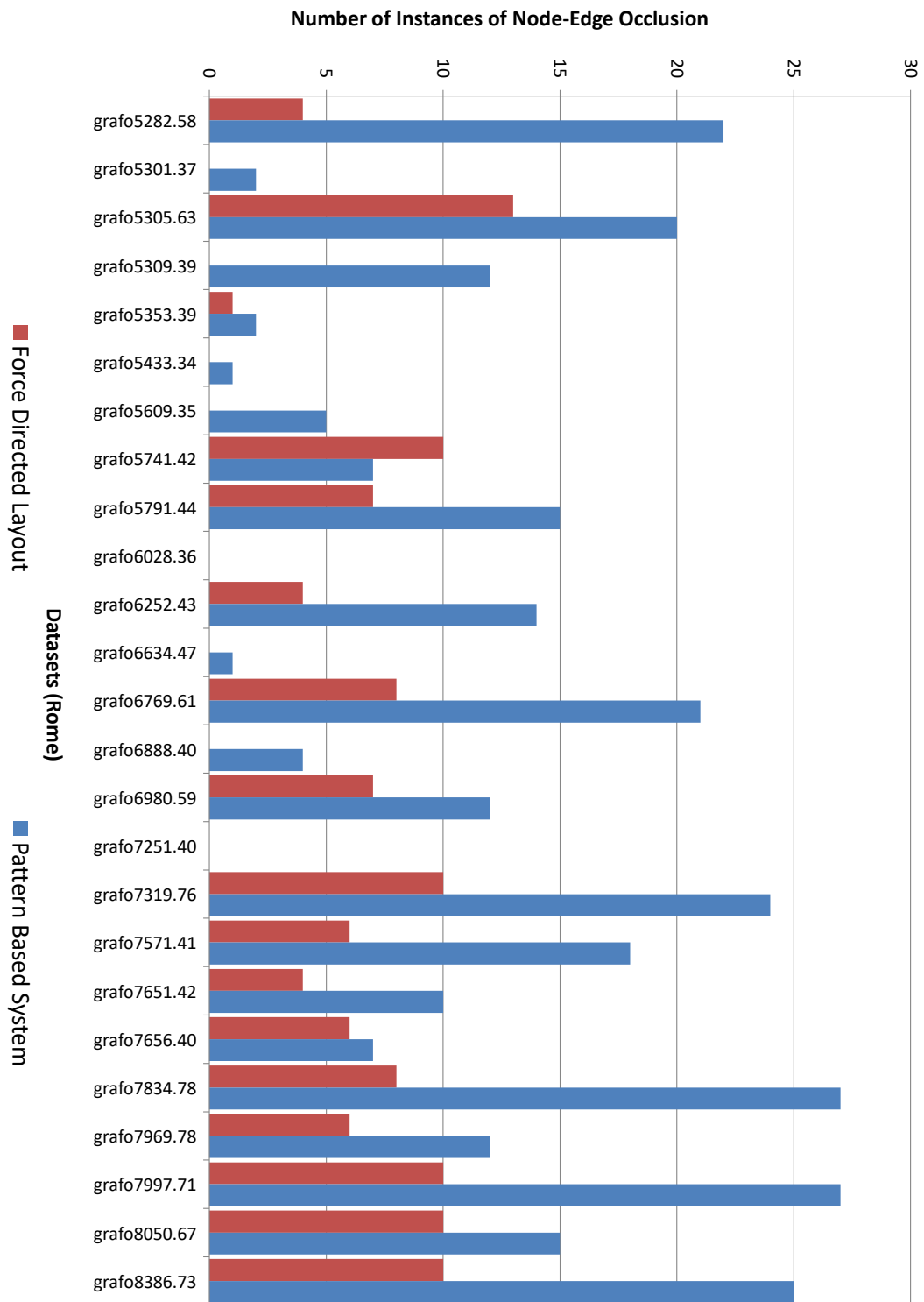


Figure A.57: Instances of Node-Edge Occlusion (Rome) (3)

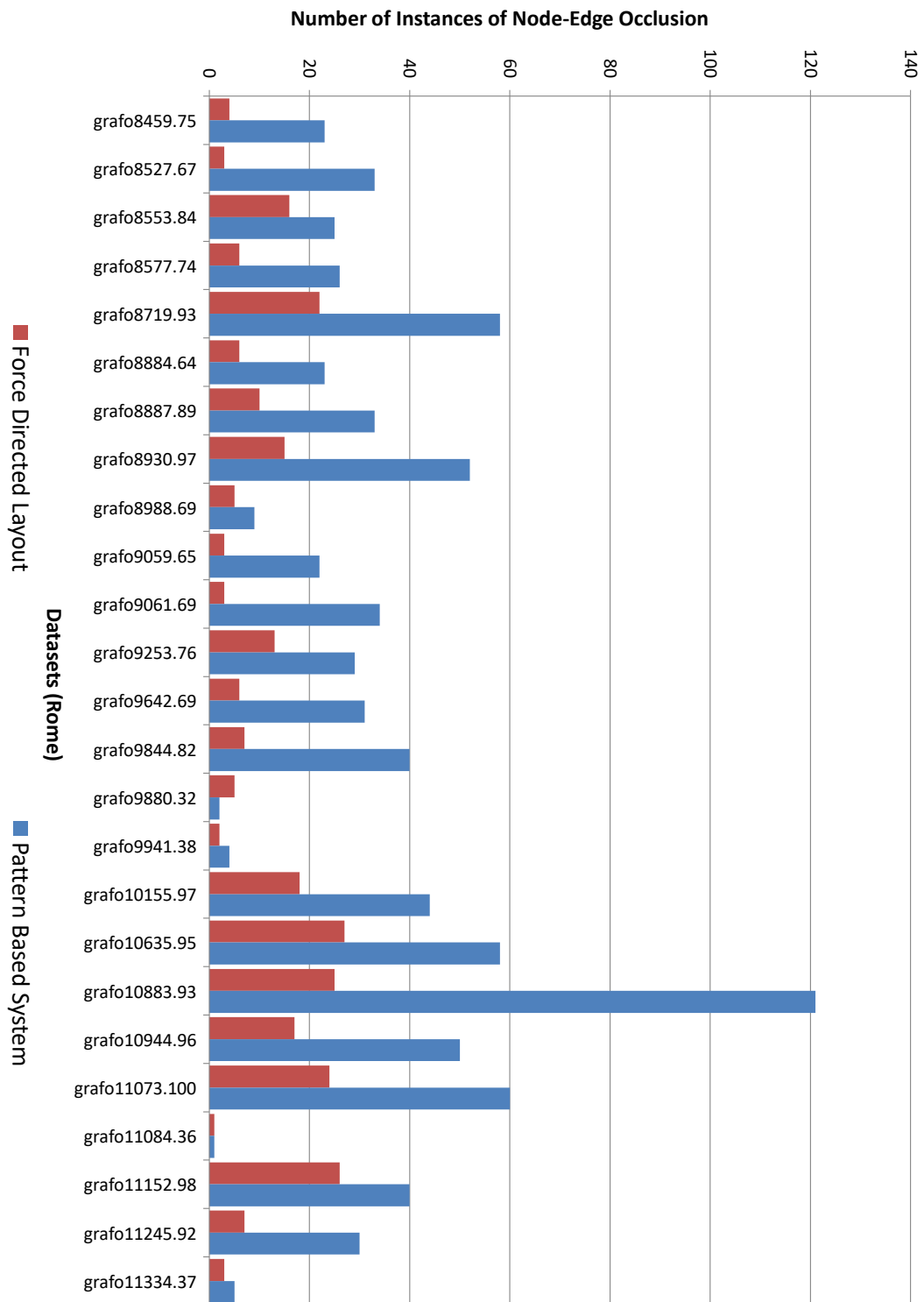


Figure A.58: Instances of Node-Edge Occlusion (Rome) (4)

A.10 Edge Crossings

The following charts show the number of edge crossings in the graph, both for the pattern based system, and a force-directed layout.

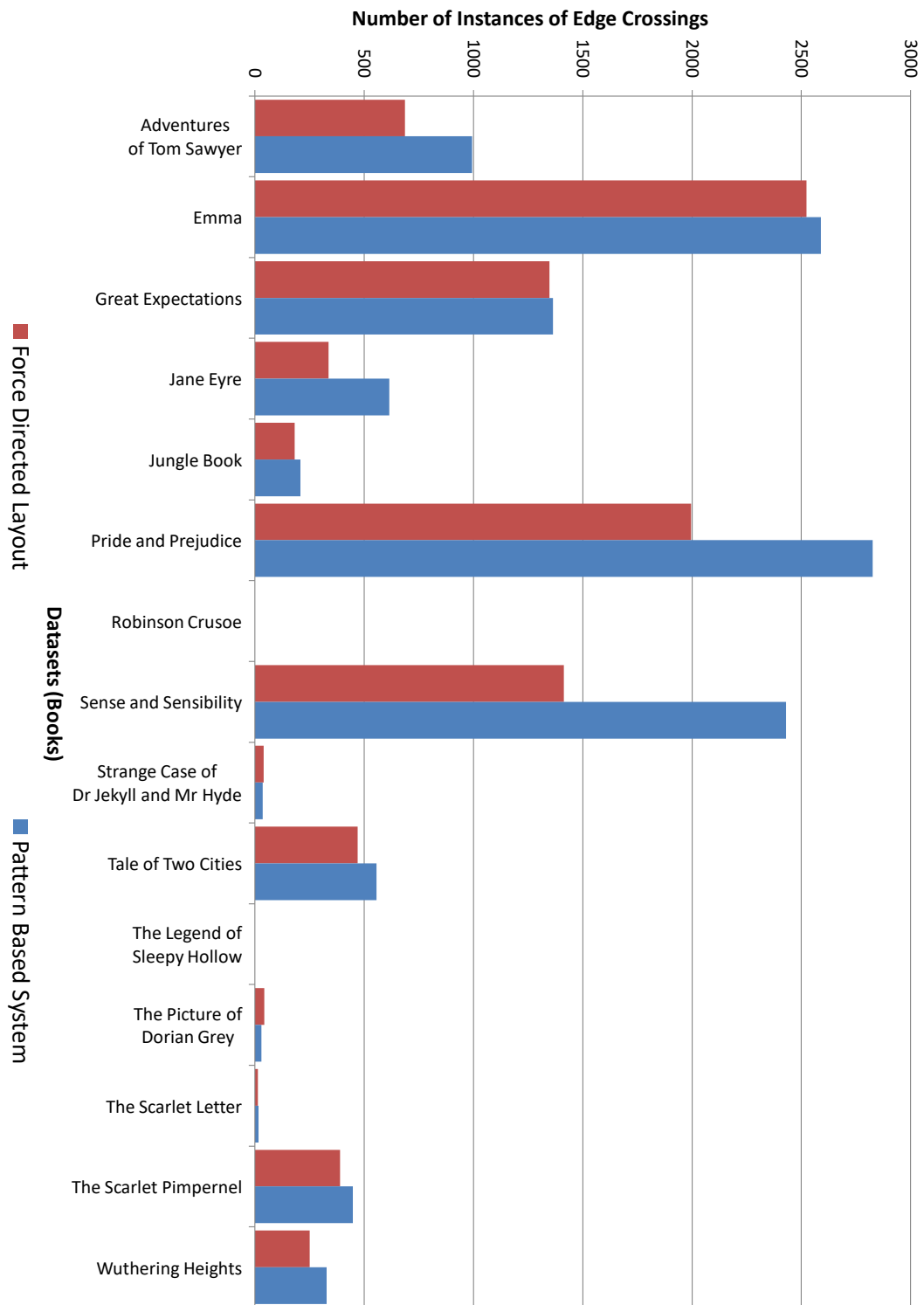


Figure A.59: Instances of Edge Crossings (Books)

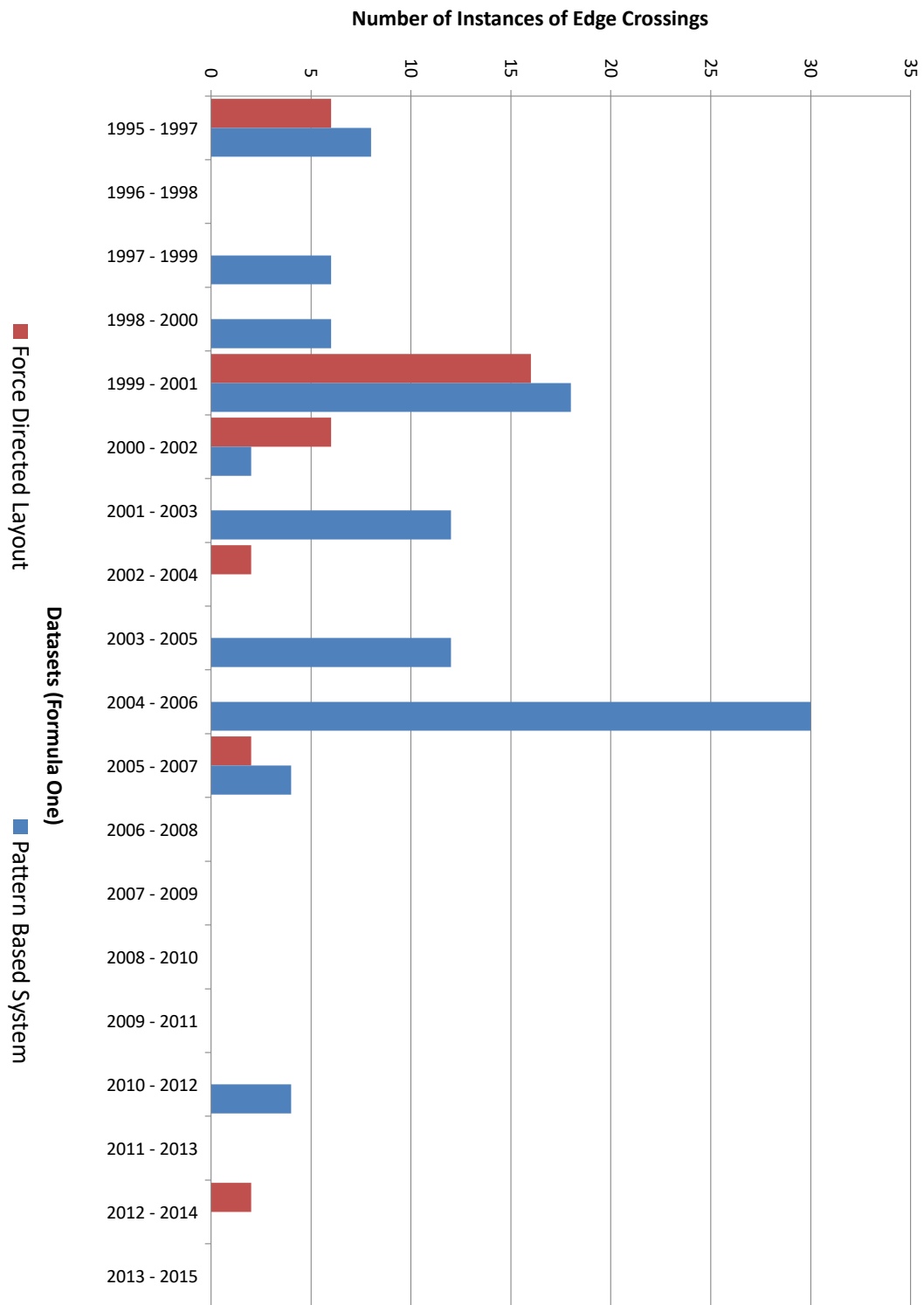


Figure A.60: Instances of Edge Crossings (Formula One)

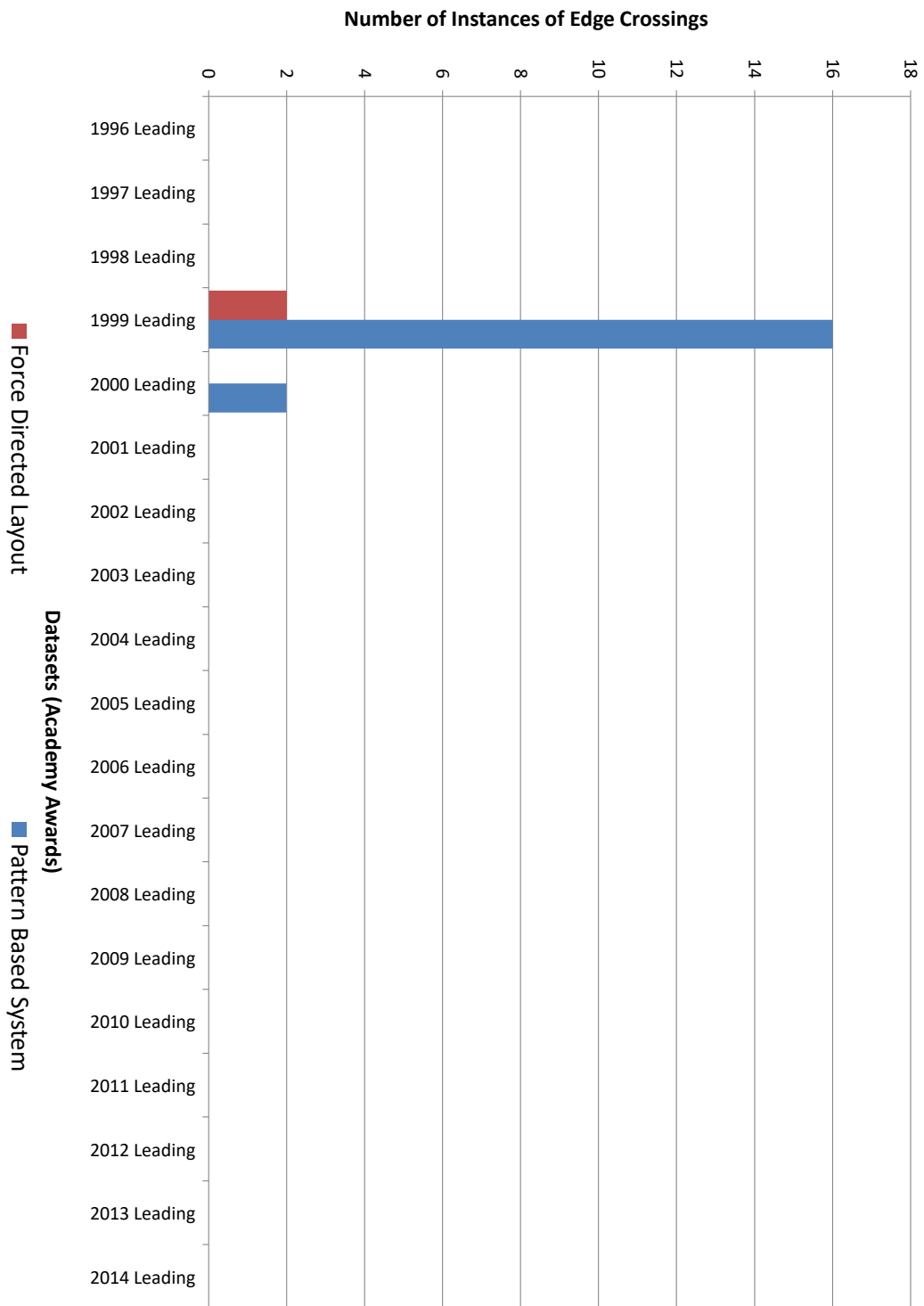


Figure A.61: Instances of Edge Crossings (Academy Awards) (1)

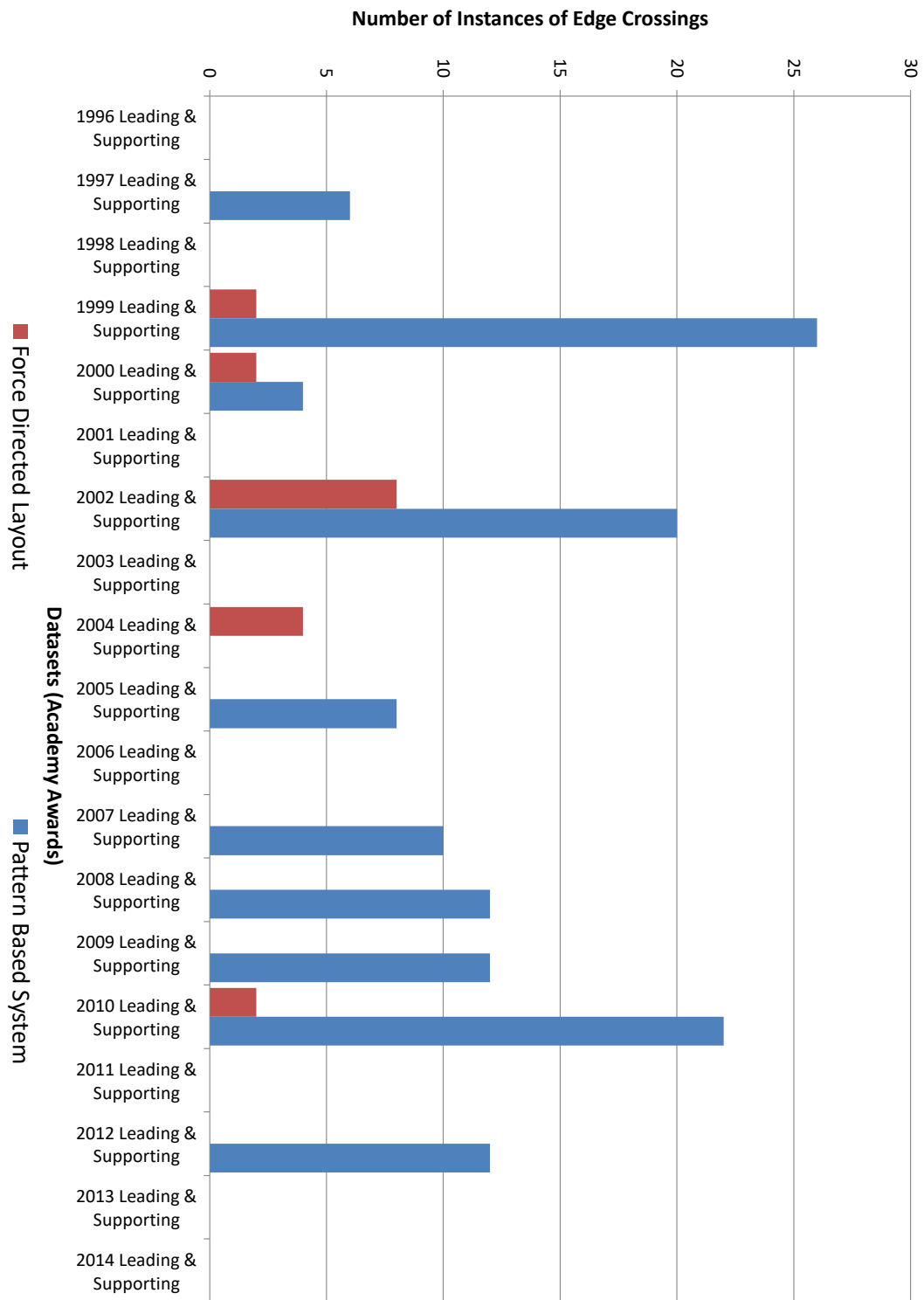


Figure A.62: Instances of Edge Crossings (Academy Awards) (2)

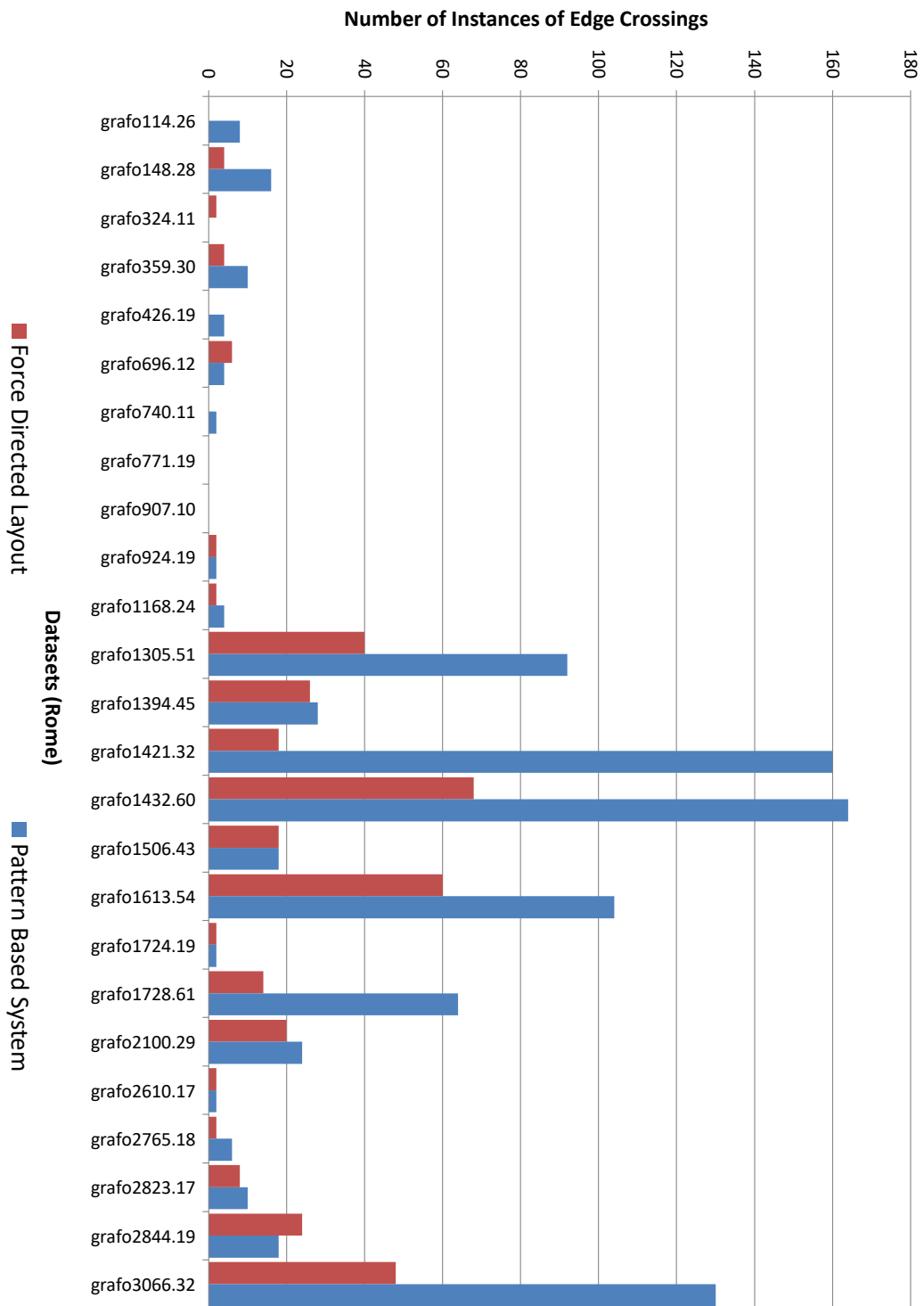


Figure A.63: Instances of Edge Crossings (Rome) (1)

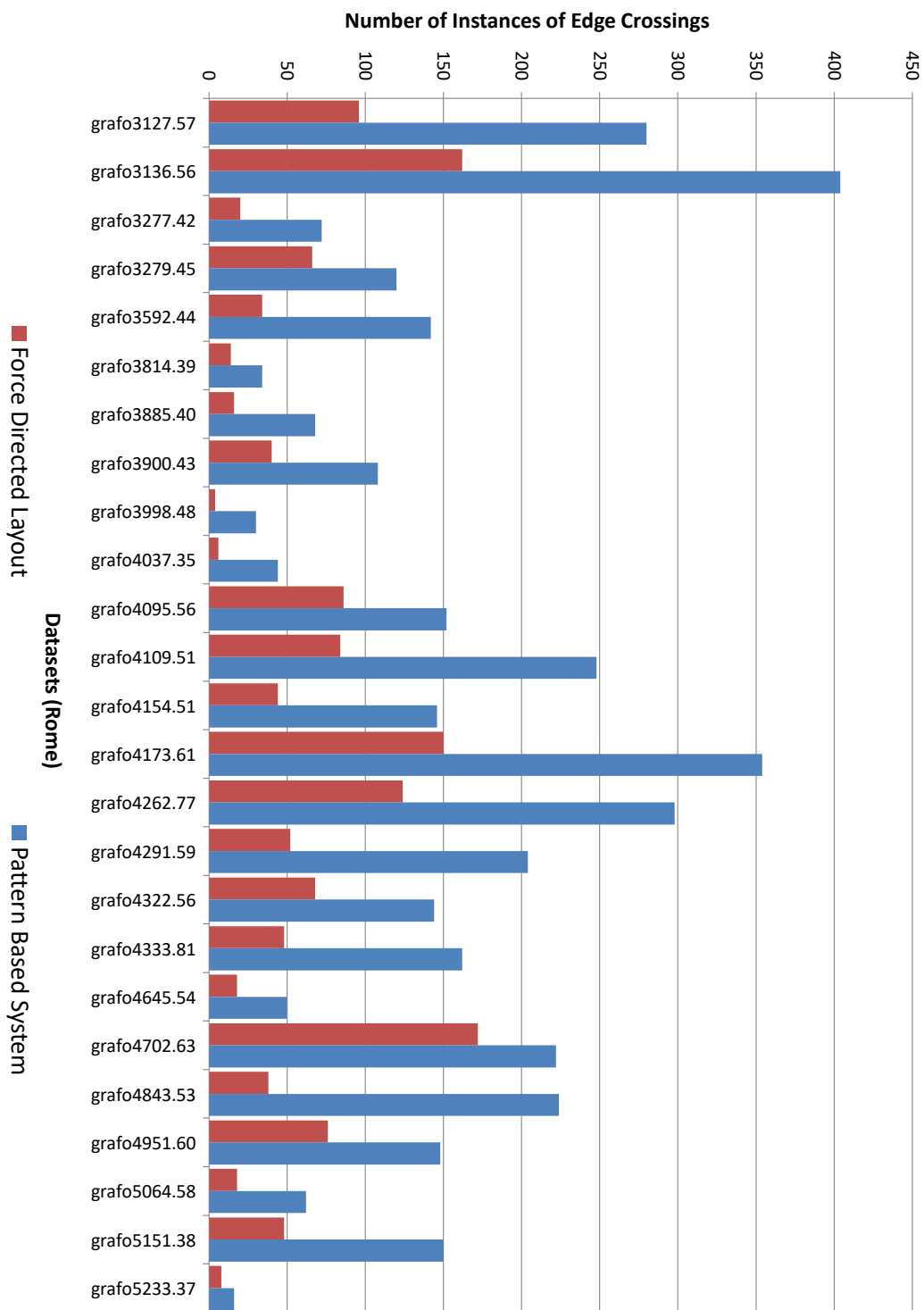


Figure A.64: Instances of Edge Crossings (Rome) (2)

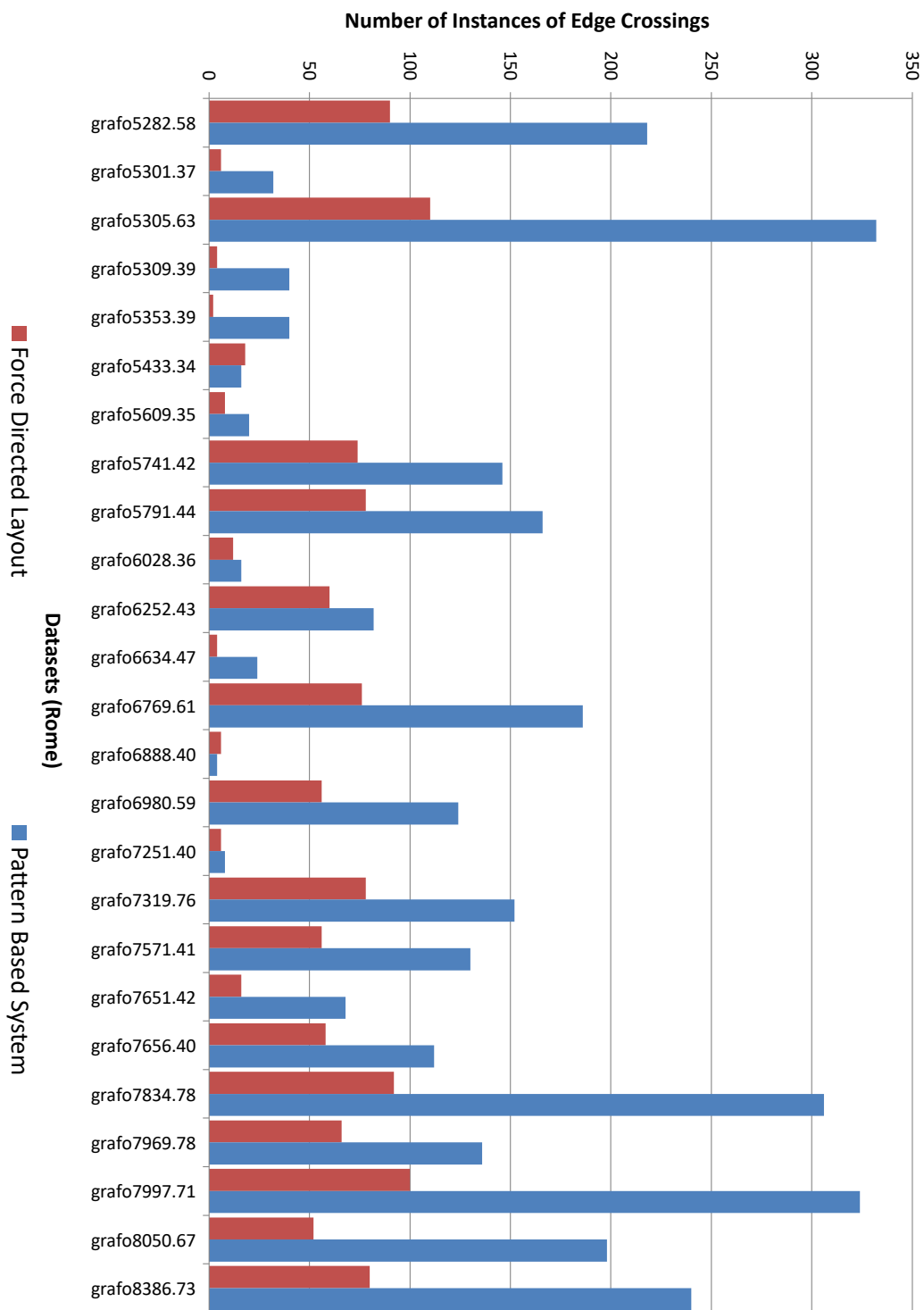


Figure A.65: Instances of Edge Crossings (Rome) (3)

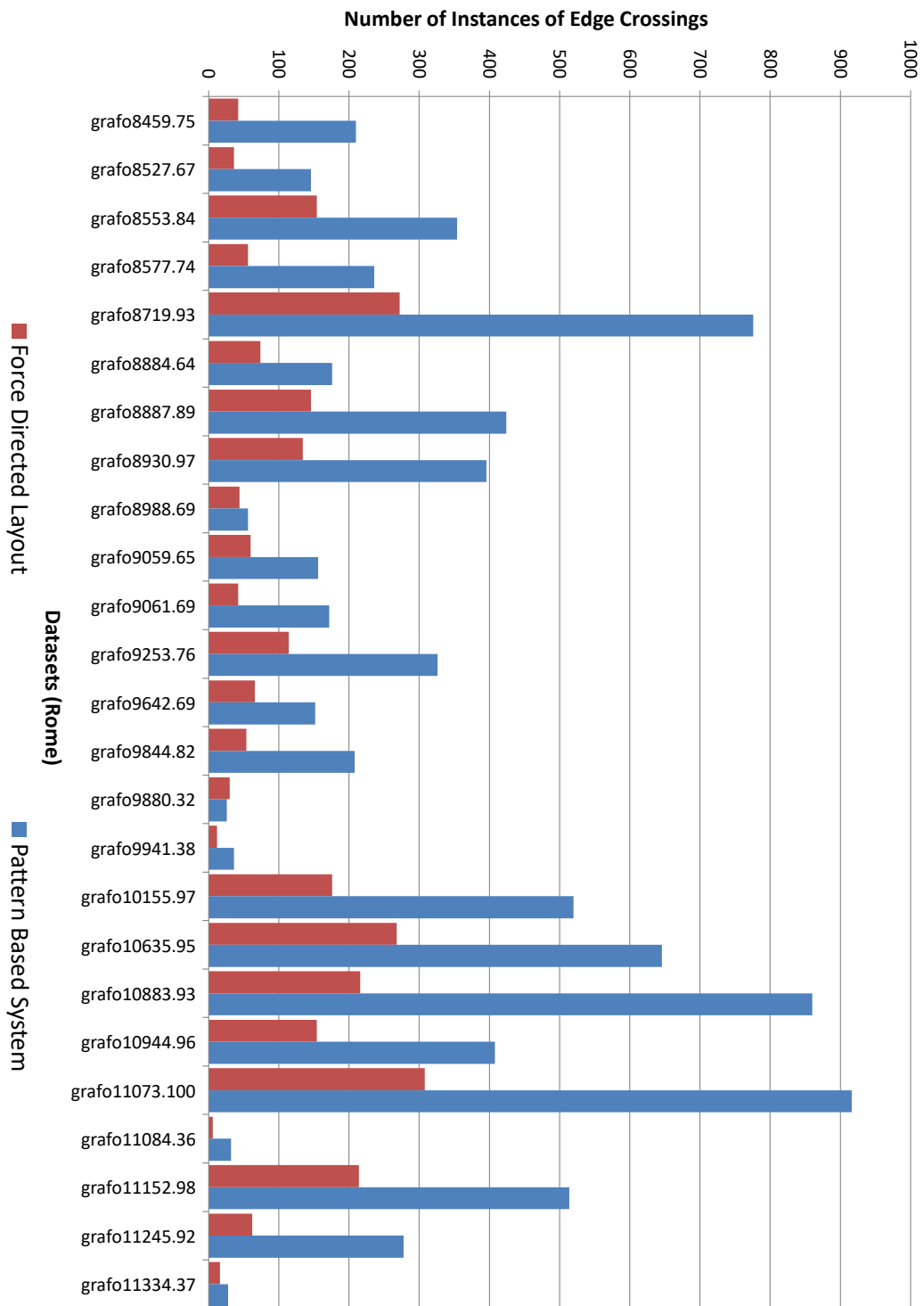


Figure A.66: Instances of Edge Crossings (Rome) (4)

A.11 Time taken

The following charts show the time taken for the pattern based system to complete a drawing.

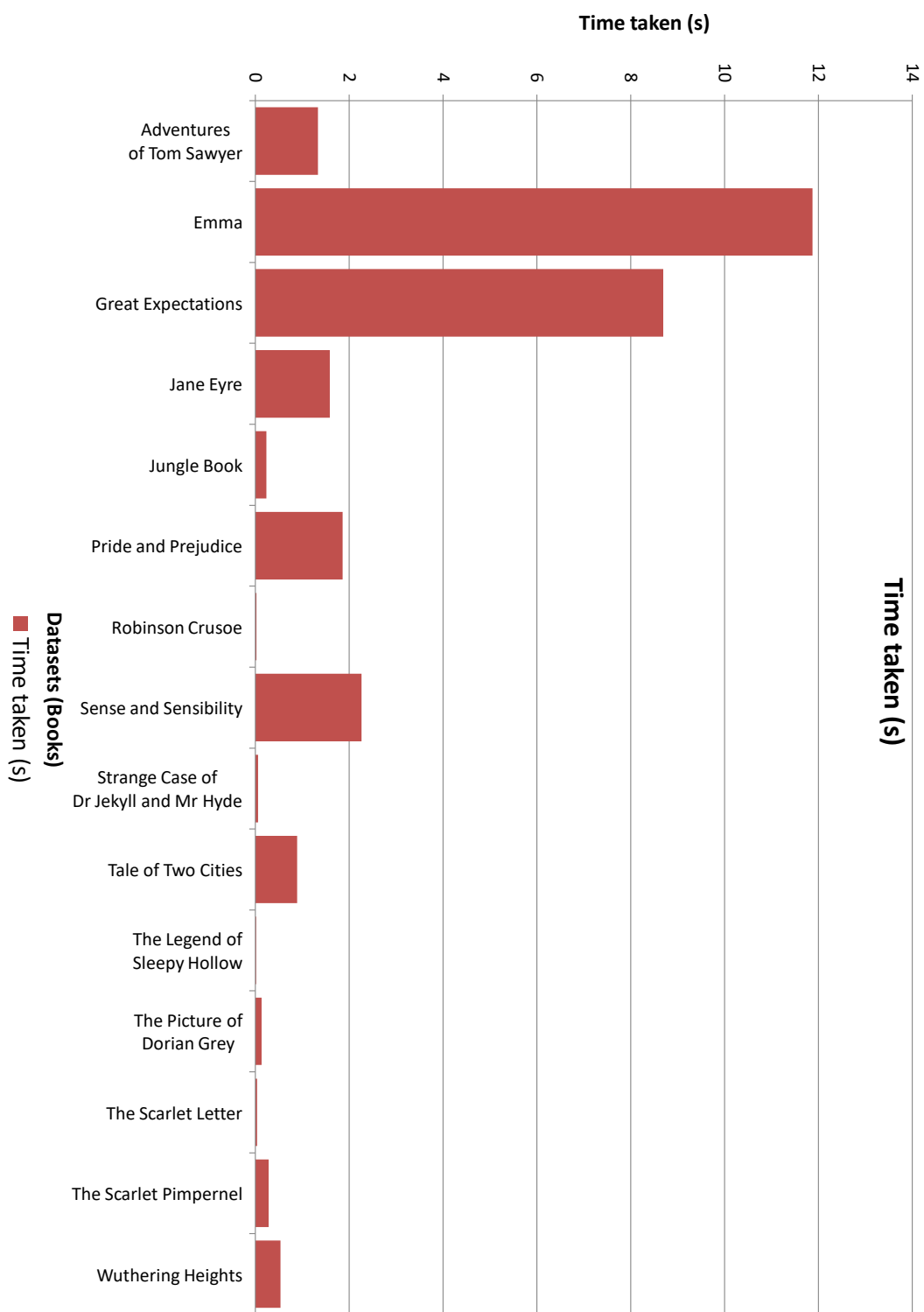


Figure A.67: Time taken (s) (Books)

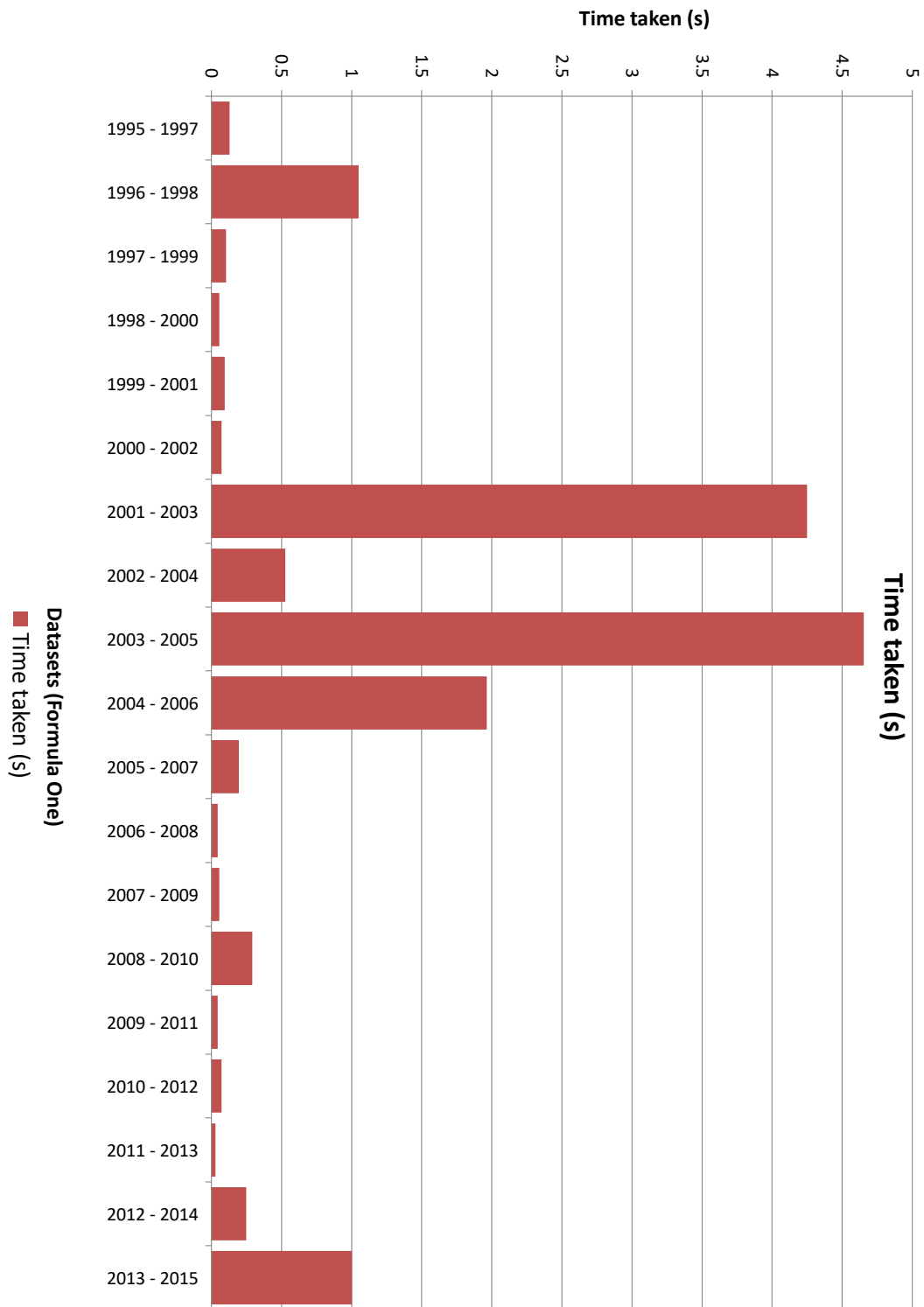


Figure A.68: Time taken (s) (Formula One)

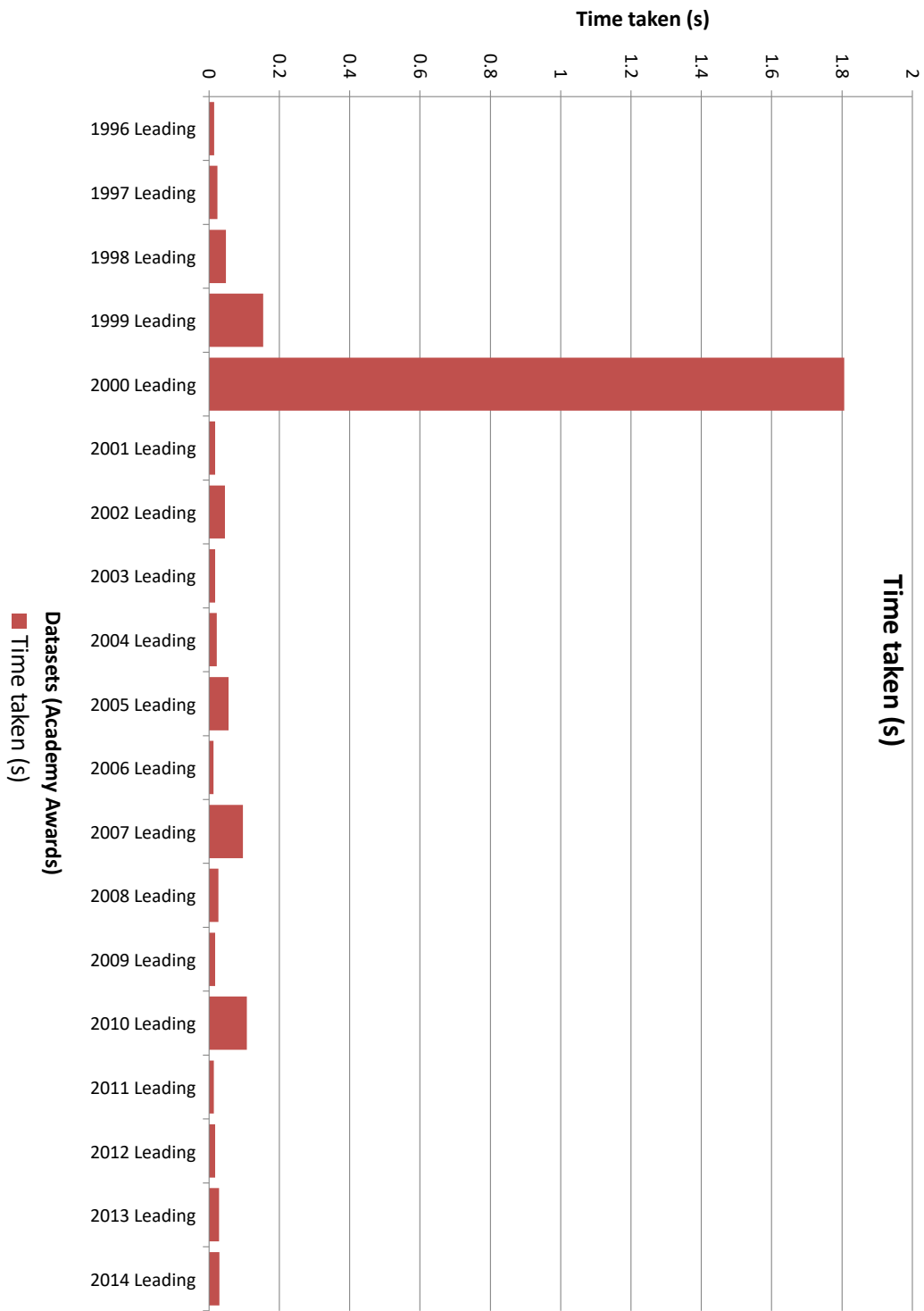


Figure A.69: Time taken (s) (Academy Awards) (1)

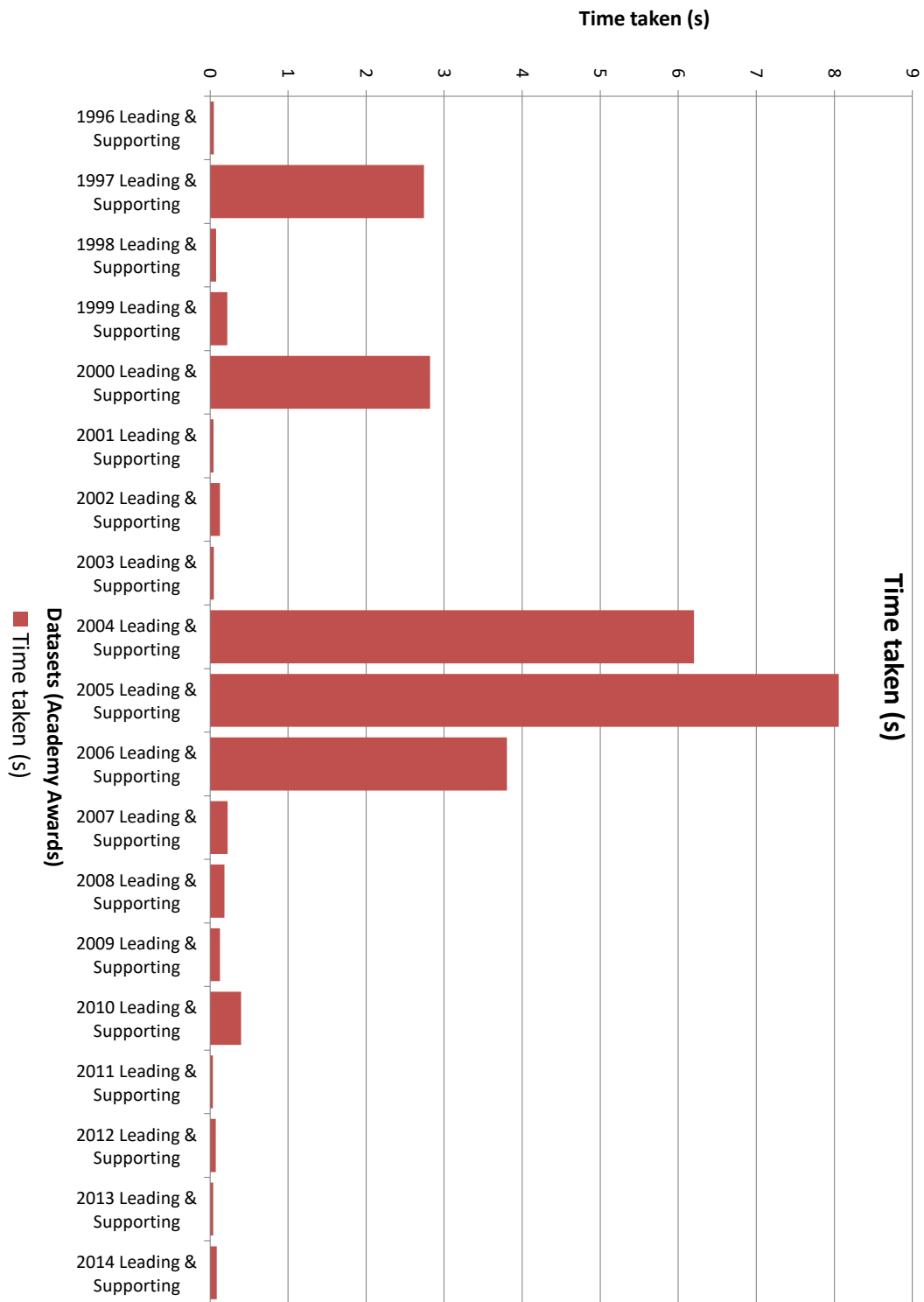


Figure A.70: Time taken (s) (Academy Awards) (2)

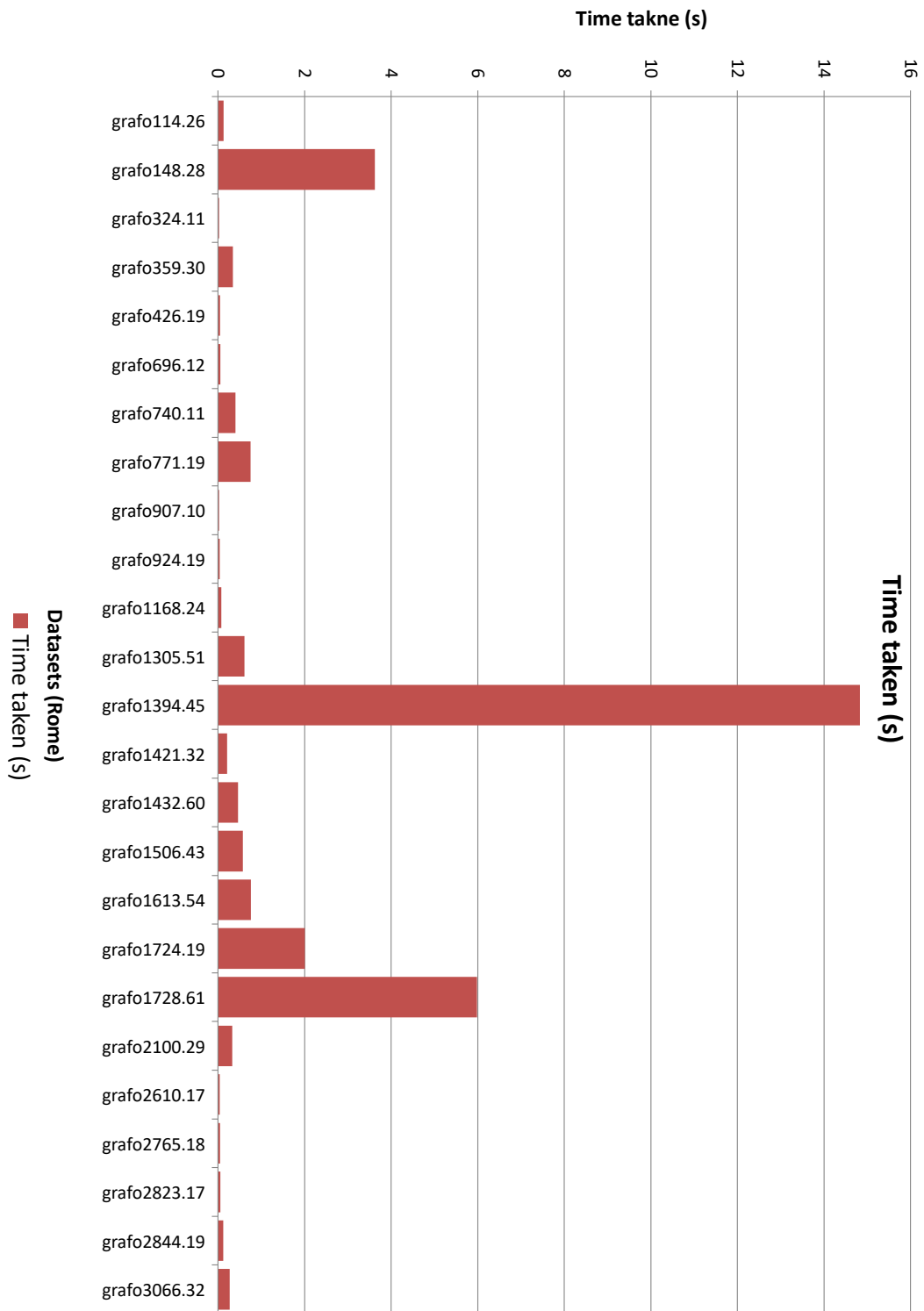


Figure A.71: Time taken (s) (Rome) (1)

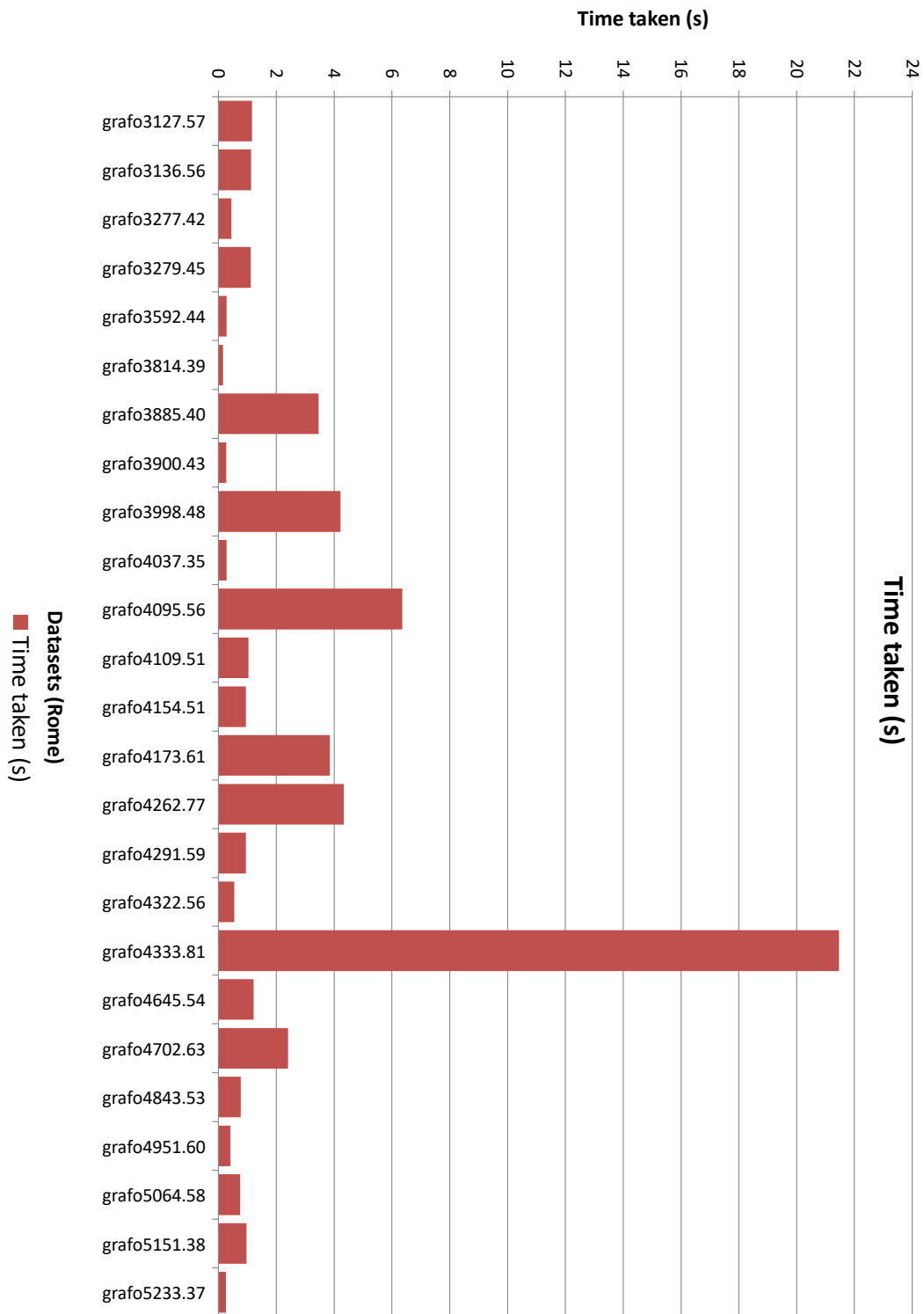


Figure A.72: Time taken (s) (Rome) (2)

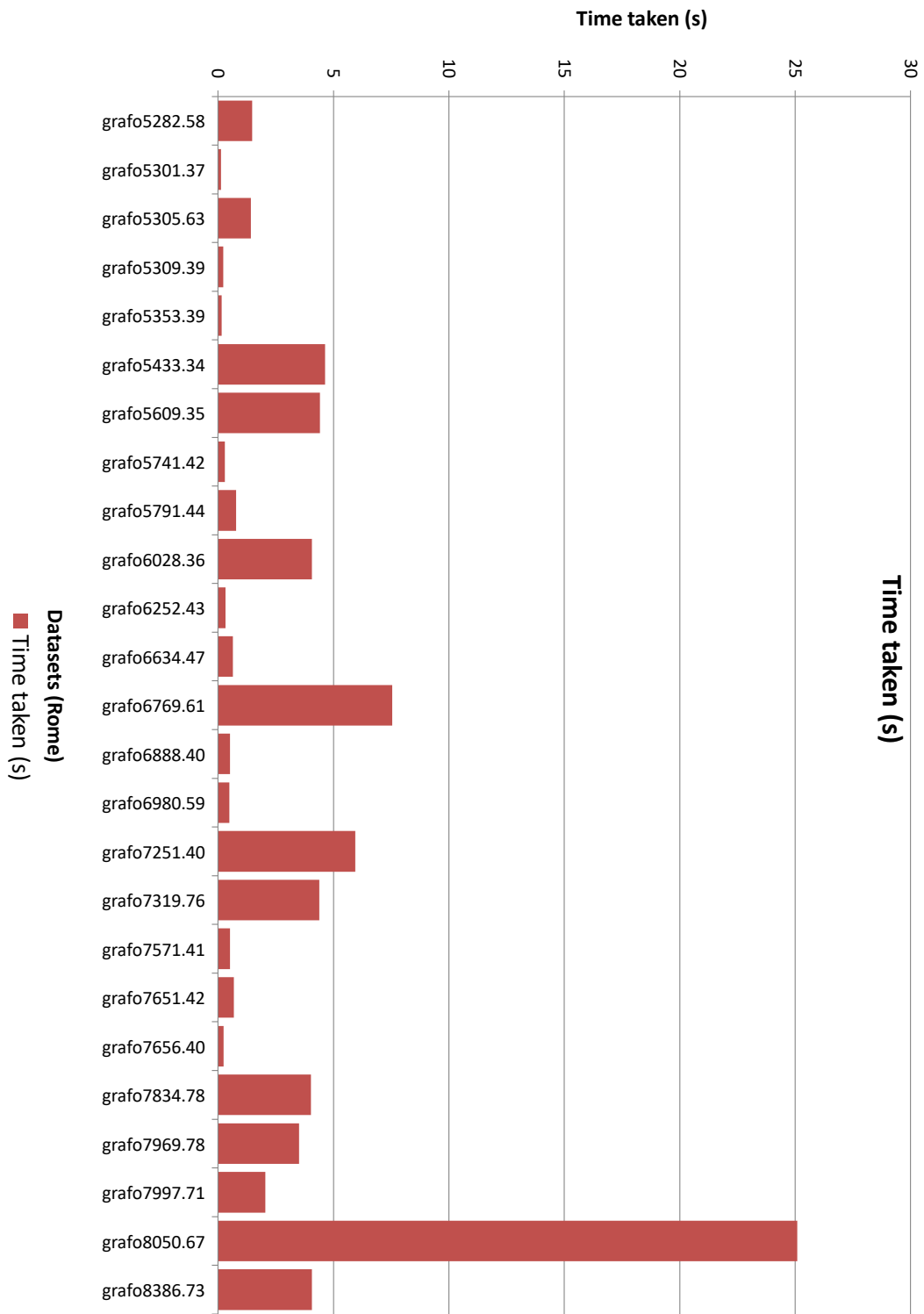


Figure A.73: Time taken (s) (Rome) (3)

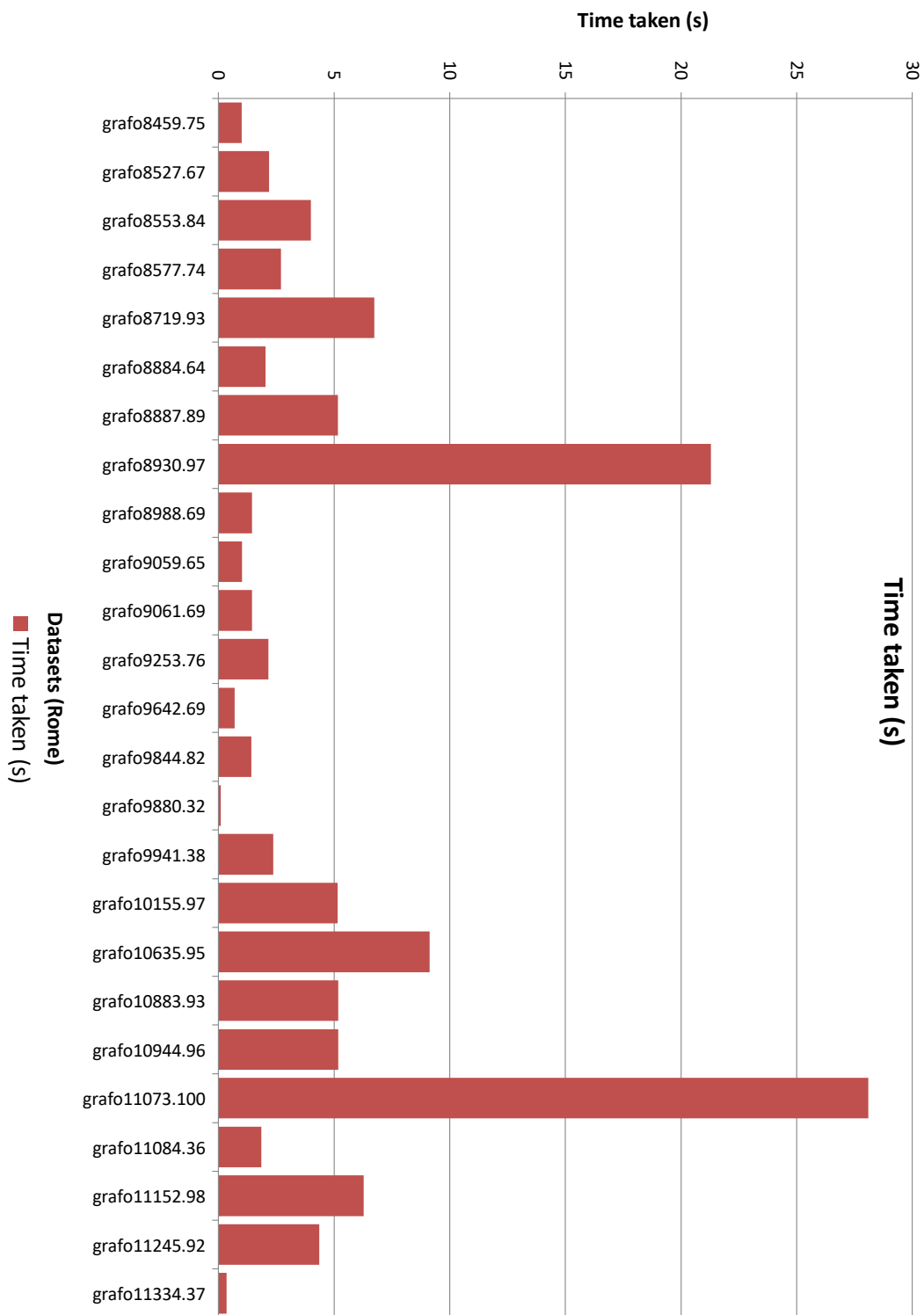


Figure A.74: Time taken (s) (Rome) (4)

Appendix B

Empirical Study Data

Detailed below are all the information provided to participants, along with the pictures and questions used. Also displayed are the raw and anonymised results for both the main study and the preference sheet. Demographic and individual comments have been removed. All of this information and the study software is available online ¹ for download in more useful formats.

In this Appendix, Question IDs are used to link pictures, questions and answers. These are in the form $t1mf$, where $t1$ represents Task Type 1, m represents the size (1 for large, m for medium, and s for small) and f represents the method (f for force-directed and p for the pattern-based technique).

B.1 Information Provided to Participants

Throughout the study, a number of pieces of information were given to the participants. These are detailed below.

¹<http://www.eulerdiagrams.com/pattern>

B.1.1 Introduction

This introduction was presented to the participant and read aloud by the person running the study.

B.1.1.1 Introduction Script

We are going to present you with a number of graphs that represent people and their connections. A person may be connected to many people. People are represented with circles. People are connected by lines.

You are now about to start the computer based test. Please feel free to adjust the chair, screen, mouse and keyboard so you are comfortable.

You can take as long as you like to answer each question, but we are recording the time taken to complete the tasks.

You will be given 8 practice questions, where the answer will be shown after each one. I will talk you through these, so please feel free to ask any questions. There will be a break once the practice is completed. You will then be given 24 questions.

After the test, you will be asked to complete a short questionnaire.

After this is completed, a sheet will be given with 3 pairs of graphs on to obtain your preferences.

Answers to all the questions asked will be treated anonymously.

You do not have to participate in this test if you do not wish to. You are free to leave now, or at any time during the test.

If you feel uncomfortable at any time, please let me know.

Are you happy to start the test now?

B.1.2 Practice Questions

The following was read to each participant during the practice questions. **Bold** text indicates instructions to the person running the study, some of which are based on the participant's response. *Italics* indicates a button on the software which must be clicked.

B.1.2.1 Practice Question Script

When you are ready, click *Begin*.

Question 1: How many people are directly connected to Billy (shown in blue)? We can see that **(pointing)** Carla, Edith and Doris are all connected to Billy, so the answer is 3.

Is that okay? **(Wait for response)**

Click on 3, then *Next*.

As it says, Billy is connected to Carla, Edith and Doris. Click *Next*.

Question 2: What is the minimum number of people that must be removed in order to disconnect Flynn & Emile (both shown in blue) so that there is no route between them?

There are two routes between Flynn and Emile. One is via Harvy **(pointing)**,

and the other via Drake & Isiah (**pointing**). If we removed Harvy and one of Drake or Isiah, there would be no connection between Flynn and Emile, so the answer is 2.

Is that okay? (**Wait for response**)

Click on 2, then *Next*

As it says, Harvy and one of Drake or Isiah need to be removed. Click *Next*.

Question 3: How many other people are there in the shortest route between Elgin & Marco (both shown in blue)?

There are two main routes from Elgin to Marco. One is on the left here (**pointing**) and the other on the right (**pointing**). The right hand side has 5 people, whereas the left hand side has only two people, who are Bambi and Harry. So the shortest route between the two has 2 other people.

Is that okay? (**Wait for response**)

Click on 2, then *Next*.

As it says, Bambi and Harry are the two other people in the shortest route. Elgin and Marco themselves are not counted. Click *Next*.

Question 4: How many triangles are there in the diagram? (A triangle is where 3 people are connected to each other)

There is a triangle made up from Carla, Freda and Ellen (**pointing**).

This is the only one in this example, so the answer is 1.

Is that okay? (**Wait for response**).

Click on 1, then *Next*.

As it says, Carla, Freda and Ellen are all in a triangle. Click *Next*.

Question 5: How many people are directly connected to Emily (shown in blue)? How many people do you think are connected to Emily? **(Wait for response)**

If 6 ▷ Good, there are indeed 6 people.

If not ▷ Okay, let's count. There are **(pointing)** 1 Philo, 2 Lexus, 3 Gregg, 4 Trent, 5 Vesla and 6 Sandy. So there are 6 people. Is that okay? **(Wait for response)**.

Click on 6, then *Next*.

As it says, there are 6 people connected to Emily. Click *Next*.

Question 6: What is the minimum number of people that must be removed in order to disconnect Ottis & Wyman (both shown in blue) so that there is no route between them?

How many people do you think need to be removed so there is no route between Ottis and Wyman? **(Wait for response)**

If 1 ▷ Who?

If Caryn or Brody ▷ Good, one of Caryn or Brody needs to be removed.

If not ▷ Okay, either Caryn or Brody **(pointing)** need to be removed. If one of them is removed, there's no route between Ottis and Wyman. Is that okay? **(Wait for response)**

If not ▷ Okay, let's work it out. If we removed Brody, there is no possible route from Ottis to Wyman. The same goes for if we removed Caryn. Either one can be removed. So the answer is 1. Is that okay? **(Wait for response)**

Click on 1, then *Next*

As it says, one of Caryn or Brody would need to be removed. Click *Next*.

Question 7: How many other people are there in the shortest route between Deana & Texas (both shown in blue)? How many people do you think there are in the shortest route between Deana and Texas?

If 4 ▷ Good. Who are they?

If Vance, Freda, Marge & Jayne ▷ Good, they are the people in the route.

If either Deana or Texas ▷ Remember, Deana and Texas aren't included in the route. Is that okay? **(Wait for response)**

If something else ▷ Okay, let's work it out. There's a route here **(pointing)** between Vance, Freda, Marge, and Jayne. Is that okay? **(Wait for response)**

If not ▷ Okay, let's work it out. There's a route here **(pointing)** between Vance, Freda, Marge and Jayne. Is that okay?

(Wait for response)

Click on 4, then *Next*.

As it says, Jayne, Freda, Marge and Vance are in the route, and that is 4 people. Click *Next*.

Question 8: How many triangles are there in the diagram? (A triangle is where 3 people are connected to each other) How many triangles do you think are in the diagram?

If 2 ▷ Good. Who are in those triangles?

If Irene, Galen & Nelle + Haley, Dante & Irene ▷ Good, they are in the two triangles.

If not ▷ Okay, let's work it out. A triangle is 3 people all connected together. There is a triangle here (**pointing**), formed of Irene, Galen & Nelle, and also one here (**pointing**) formed of Irene again, Haley and Dante. Is that okay?

(Wait for response)

If not ▷ Okay, let's work it out. A triangle is 3 people all connected together. There is a triangle here (**pointing**), formed of Irene, Galen & Nelle, and also one here (**pointing**) formed of Irene again, Haley and Dante. Is that okay? **(Wait for response)**

Click on *1*, then *Next*

As it says, Irene, Galen and Nelle are in a triangle, as are Irene again, Haley and Dante. Click *Next*.

I shall now move over the other side of the table (**move to other side of the table**) and allow you to complete the rest of the study. If you have any questions, please let me know now. If you have no questions then when you are ready, click *Begin*.

B.1.3 Conclusion

The following was read to each participant once they had completed their preference sheet, and they were told they could keep this debriefing document. It was also read aloud to the participants. Once completed, they signed a form as receipt of collecting their payment and were given with £7.

B.1.3.1 Conclusion Script

Thank you for participating in this research.

You were presented with a number of graphs which were drawn using two different drawing methods. One method was a forced based method, with the other being a pattern layout system. We wanted to find out how the drawing method alters people's understanding of these diagrams.

The purpose of this research is to evaluate the effectiveness of the pattern based system compared to another drawing mechanism.

We would appreciate it if you did not discuss this experiment with other students in the university. These experiments will be continuing through the next few weeks, and having subjects who have prior knowledge of what the tests are about makes the data less useful.

Thank you again for your contribution.

B.2 Examples Used

The Question IDs used in the following examples correlate with the Question IDs in Sections B.3 and B.5.

B.2.1 Practice

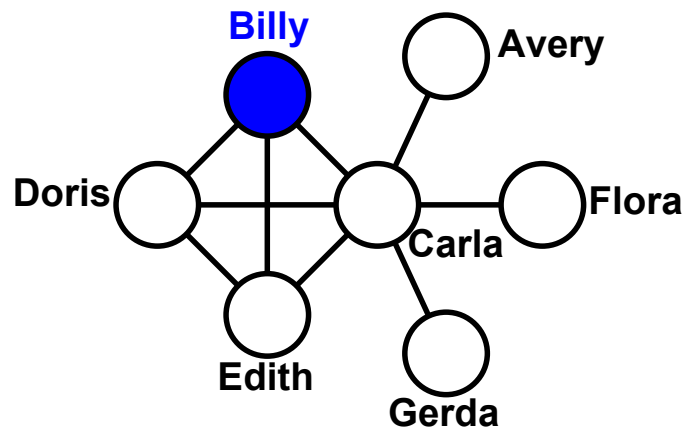


Figure B.1: p1 - Practice 1

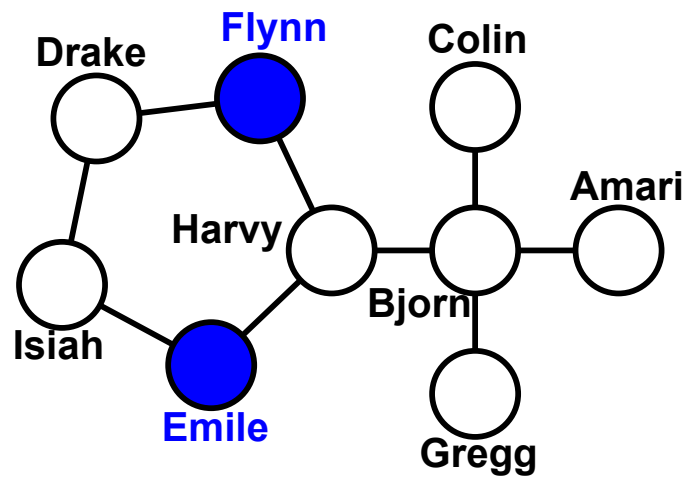


Figure B.2: p2 - Practice 2

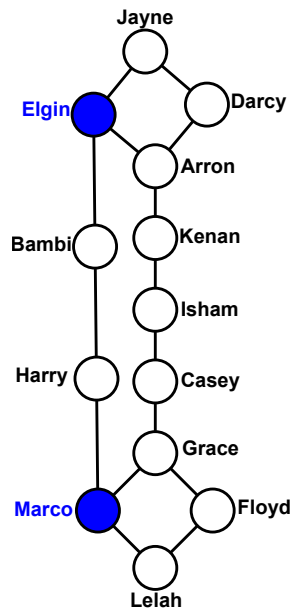


Figure B.3: p3 - Practice 3

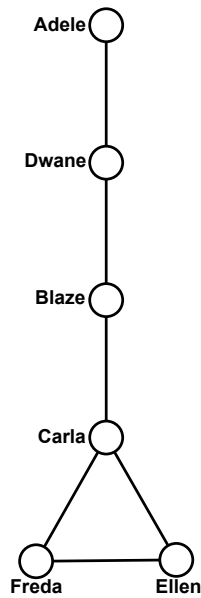


Figure B.4: p3a - Practice 4

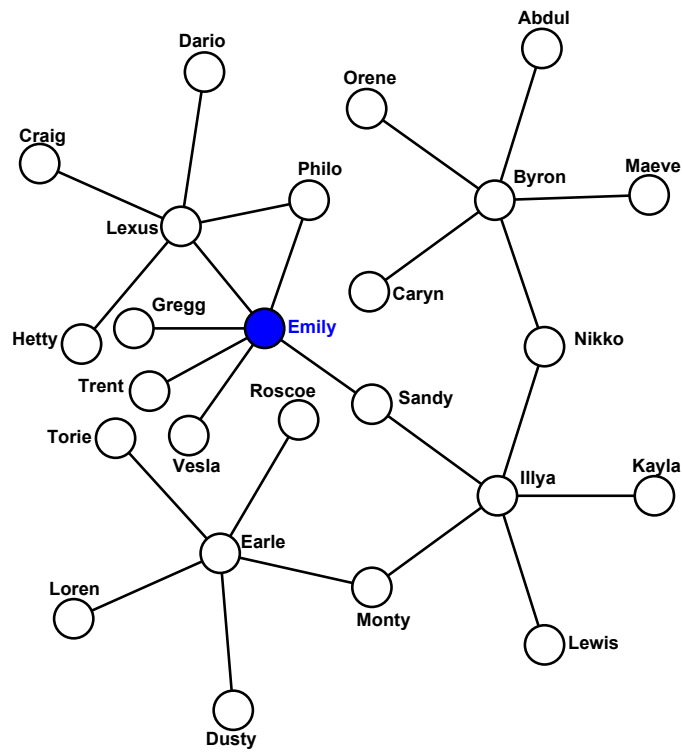


Figure B.5: p4 - Practice 5

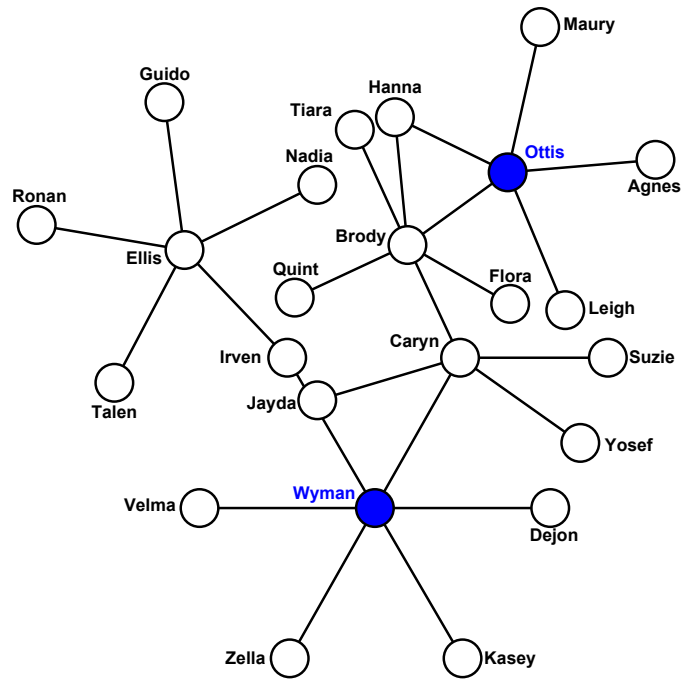


Figure B.6: p5 - Practice 6

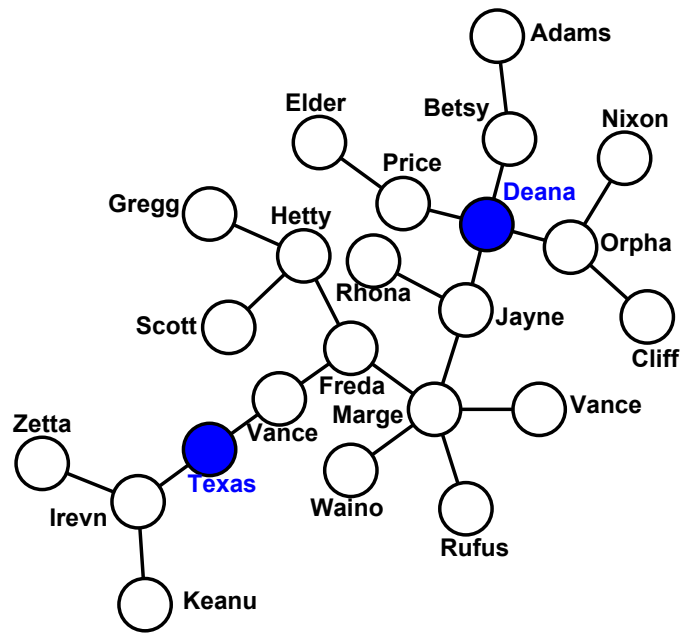


Figure B.7: p6 - Practice 7

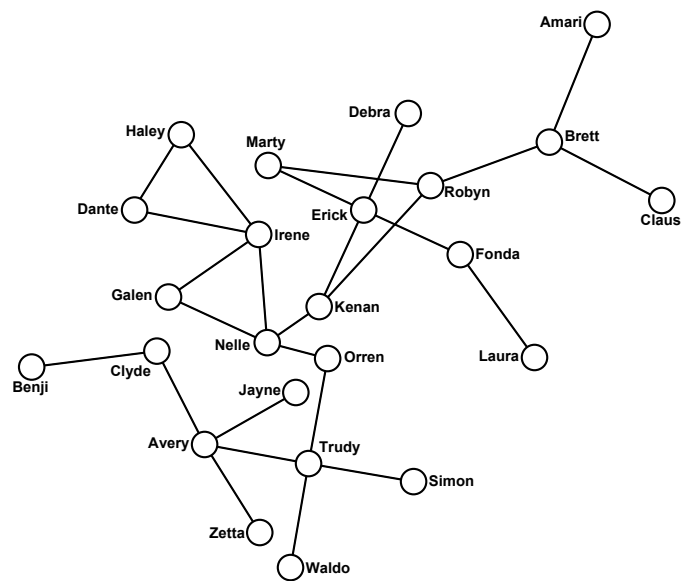


Figure B.8: p7 - Practice 8

B.2.2 Task Type 1

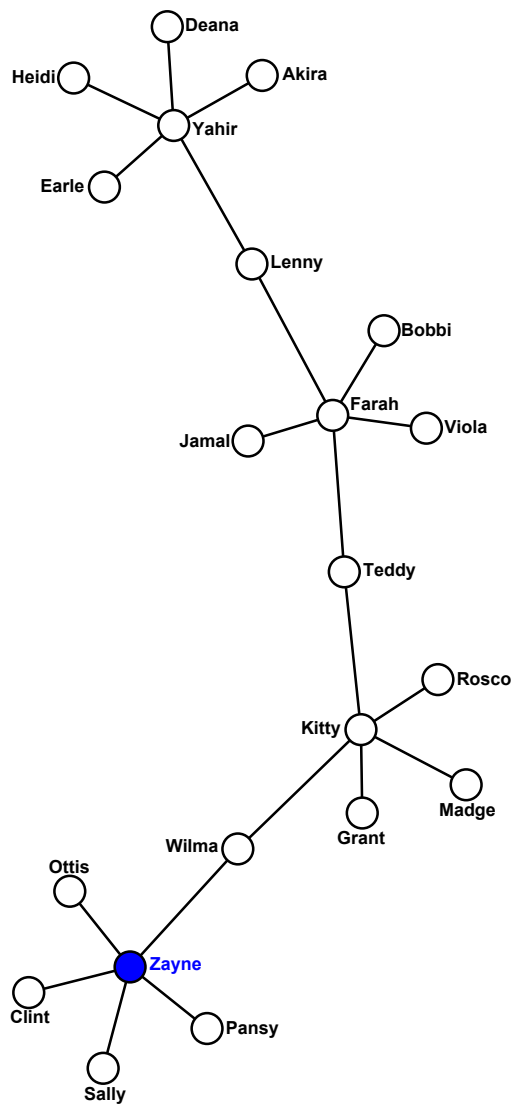


Figure B.9: t1sf - 1998 Best Leading and Supporting Actor and Actress Nominations

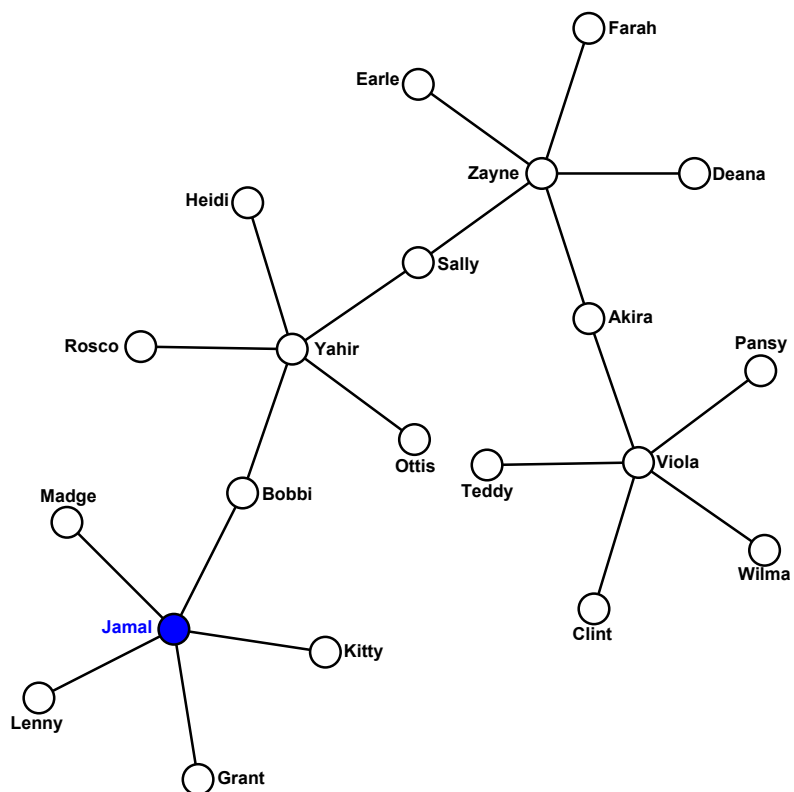


Figure B.10: t1sp - 1998 Best Leading and Supporting Actor and Actress Nominees

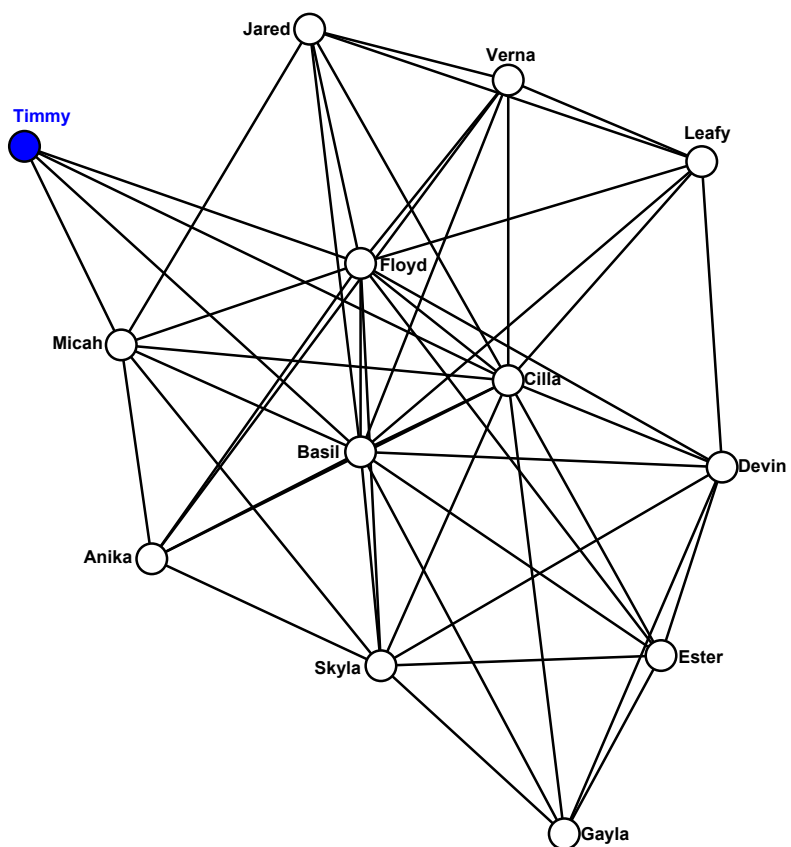


Figure B.11: t1mf - *The Jungle Book*, by Rudyard Kipling (1894)

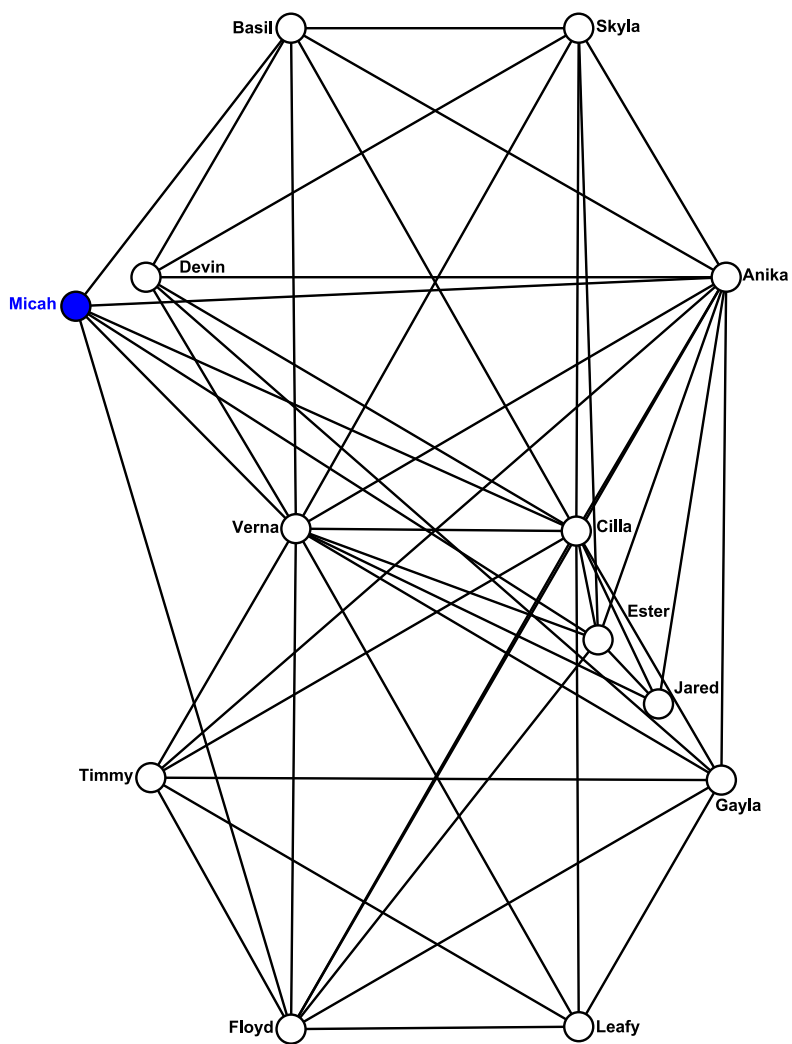


Figure B.12: t1mp - *The Jungle Book*, by Rudyard Kipling (1894)

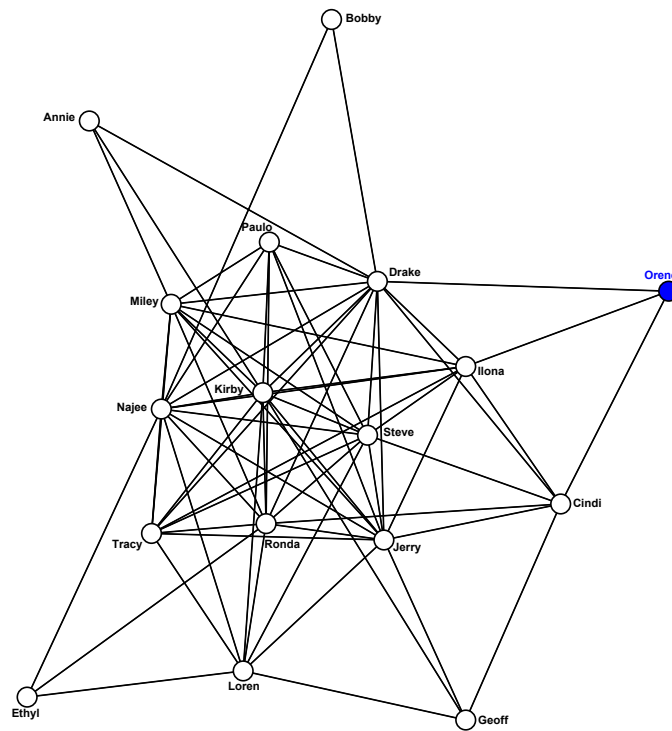


Figure B.13: t11f - *A Tale of Two Cities*, by Charles Dickens (1859)

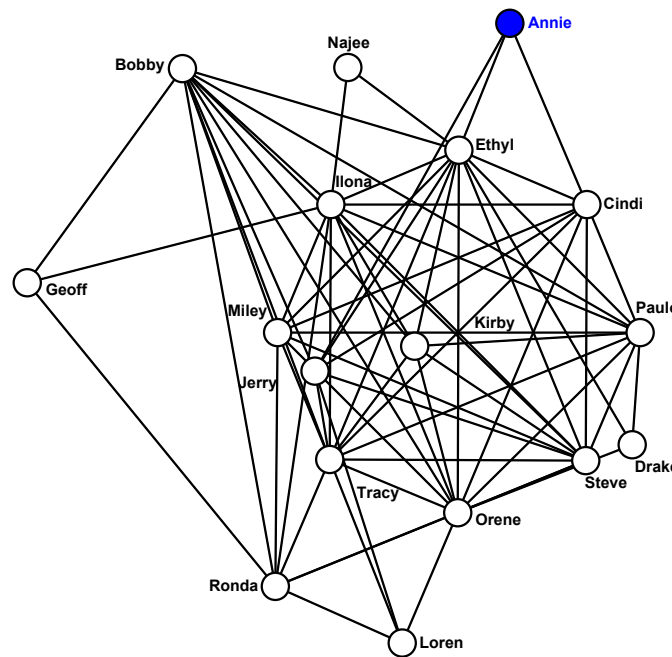


Figure B.14: t11p - *A Tale of Two Cities*, by Charles Dickens (1859)

B.2.3 Task Type 2

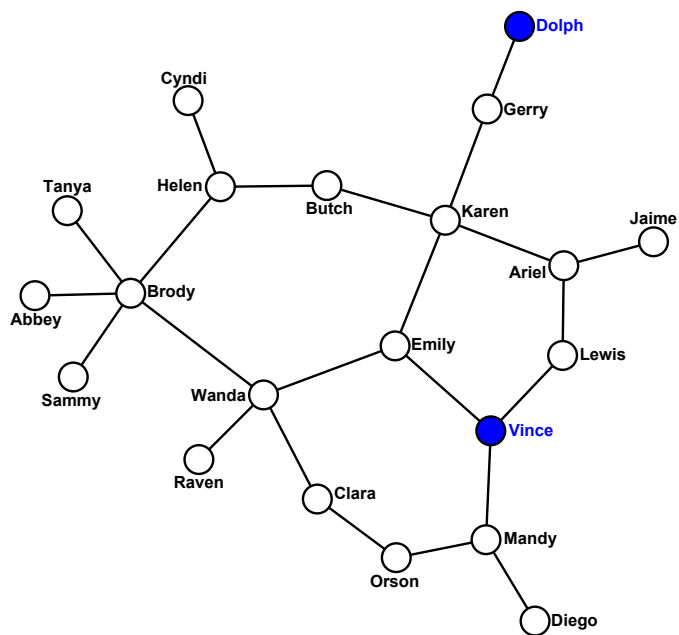


Figure B.15: t2sf - Formula One team-mates, 1997-1999

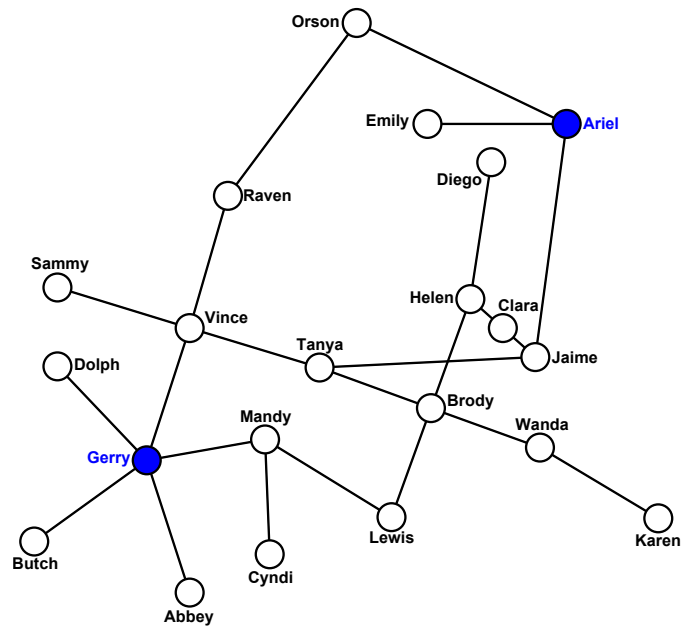


Figure B.16: t2sp - Formula One team-mates, 1997-1999

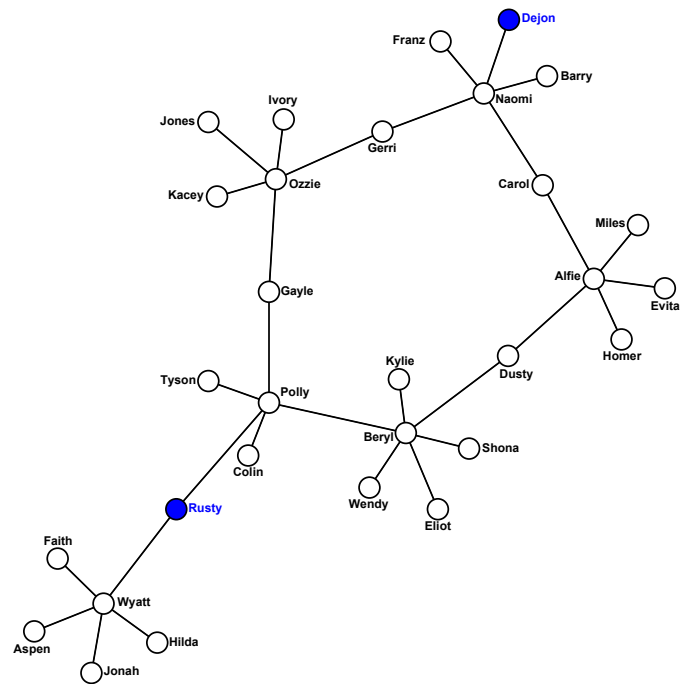


Figure B.17: t2mf - 2000 Best Leading Actor and Actress Nominees

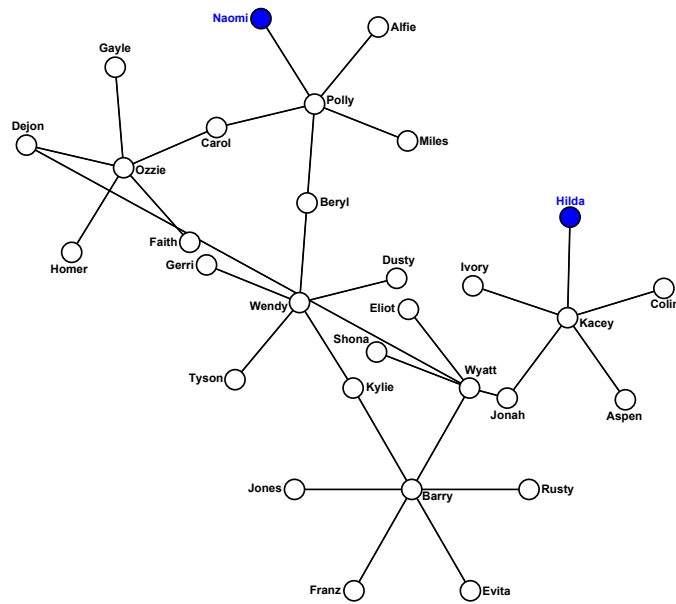


Figure B.18: t2mp - 2000 Best Leading Actor and Actress Nominees

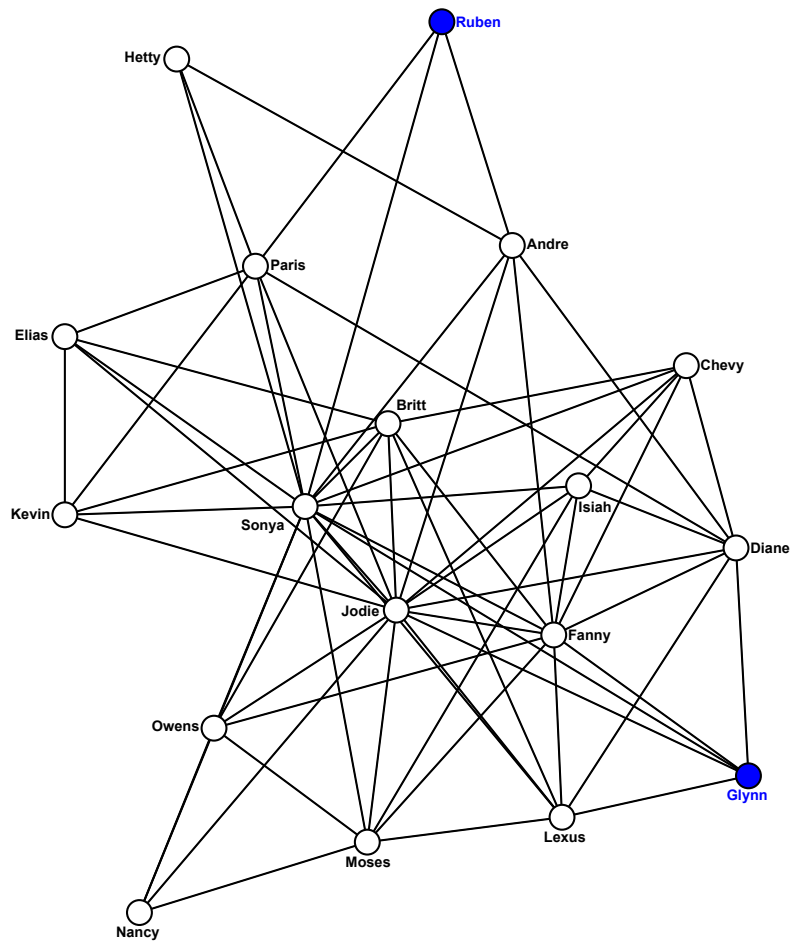


Figure B.19: t2lf - *Jane Eyre*, by Charlotte Brontë (1847)

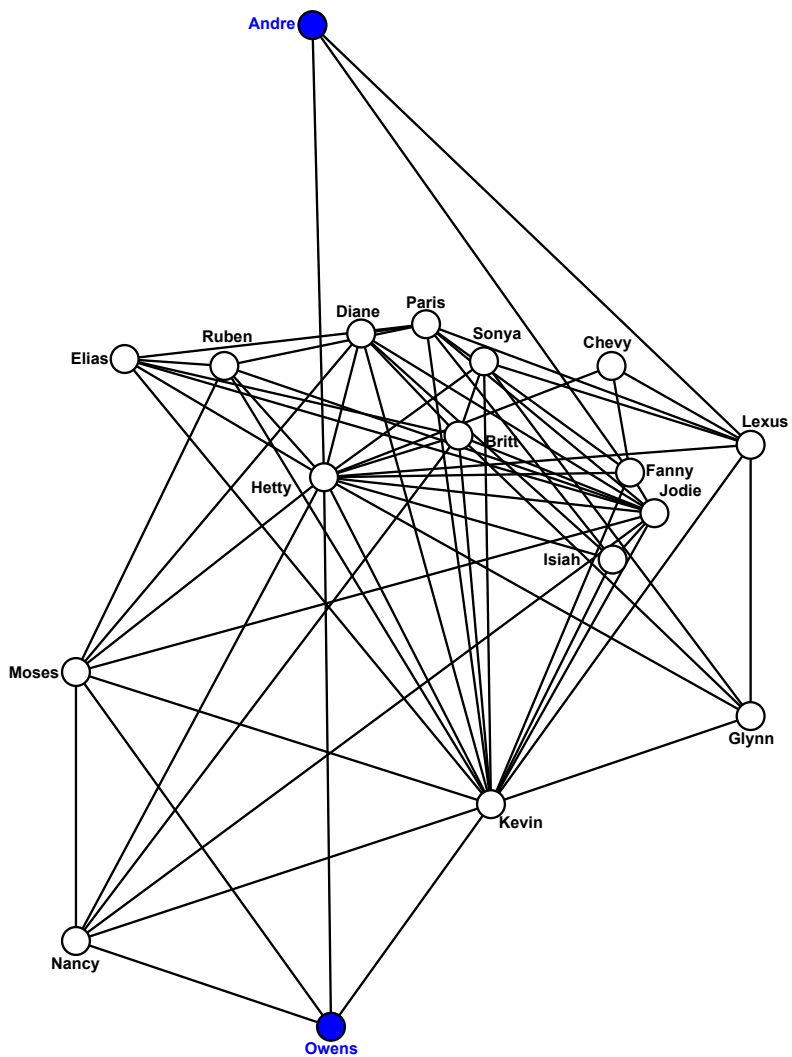


Figure B.20: t2lp - *Jane Eyre*, by Charlotte Brontë (1847)

B.2.4 Task Type 3

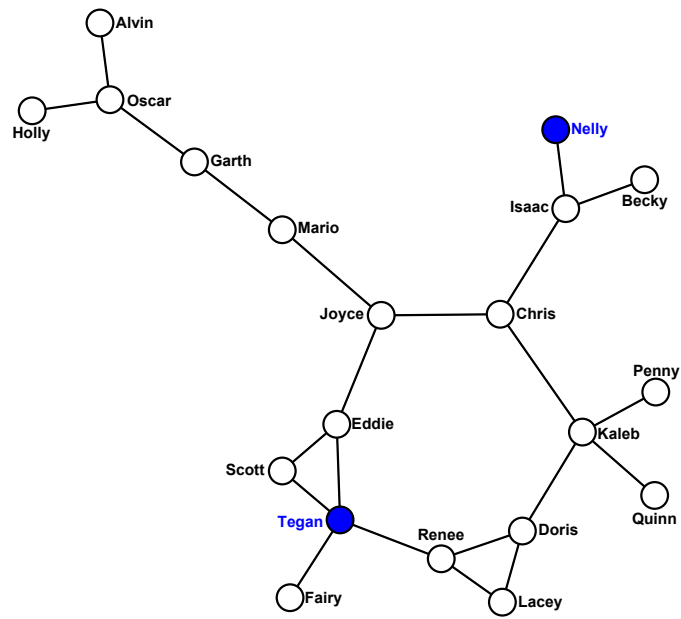


Figure B.21: t3sf - Formula One team-mates, 2009-2011

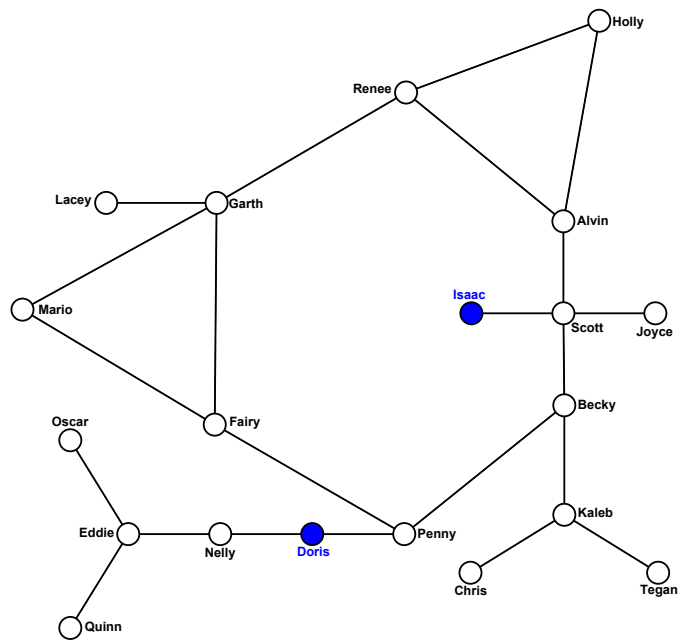


Figure B.22: t3sp - Formula One team-mates, 2009-2011

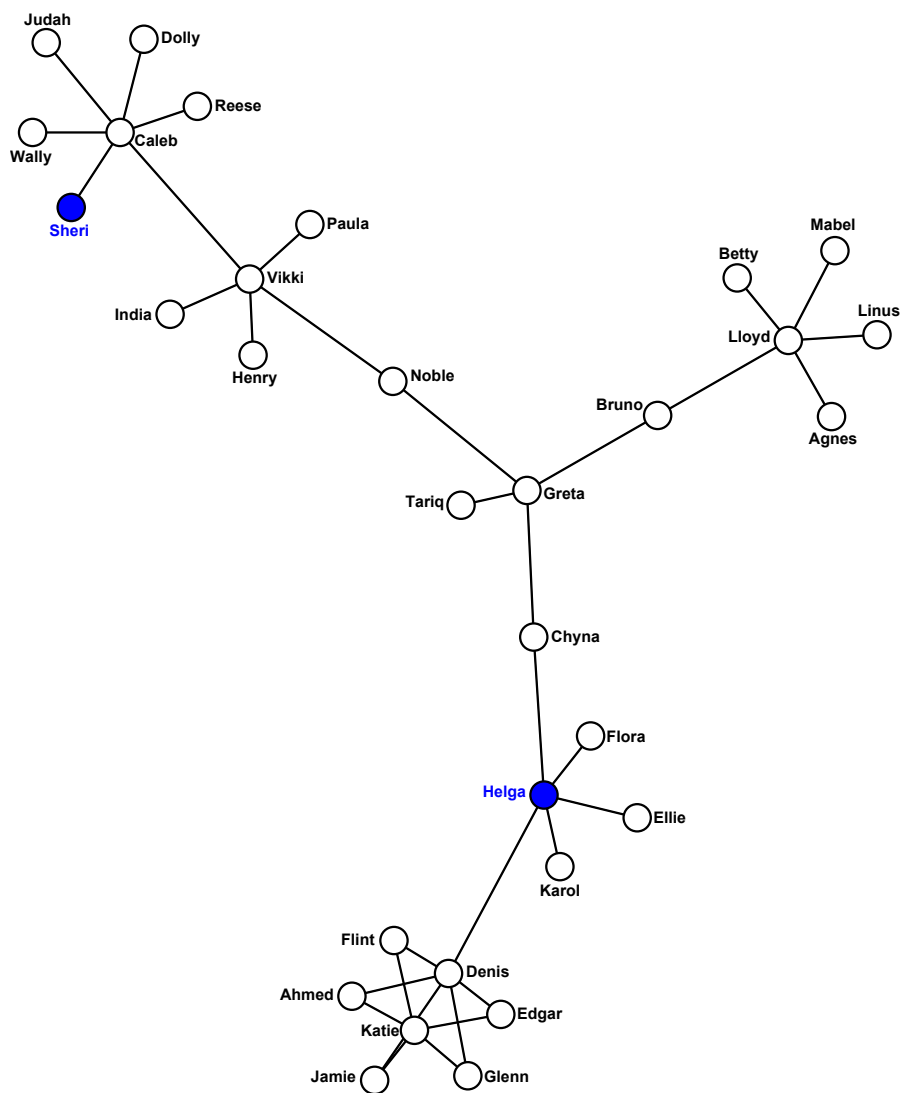


Figure B.23: t3mf - 2022 Best Leading and Supporting Actor and Actress Nominees

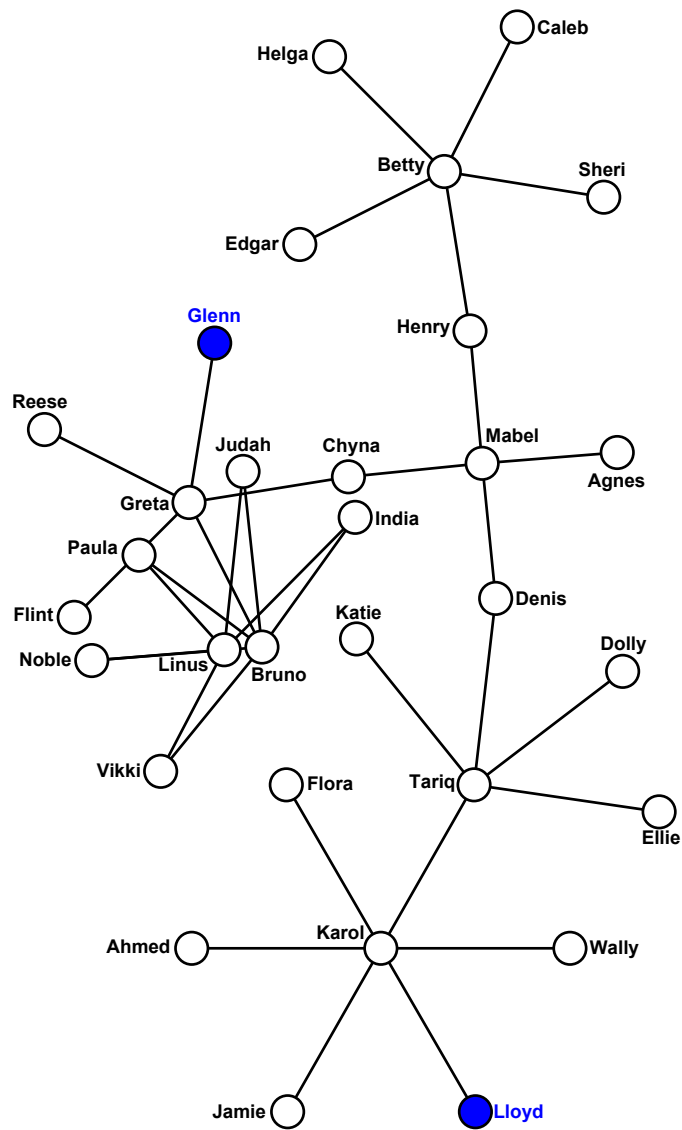


Figure B.24: t3mp - 2002 Best Leading and Supporting Actor and Actress Nominees

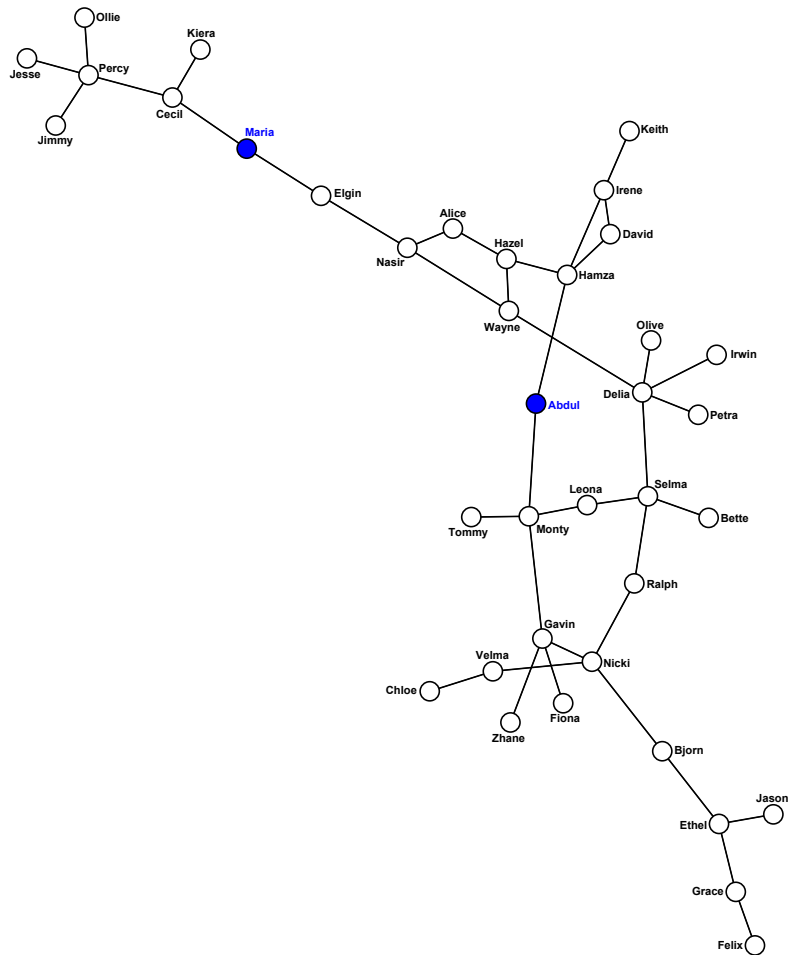


Figure B.25: t3lf - Formula One team-mates, 2004-2006

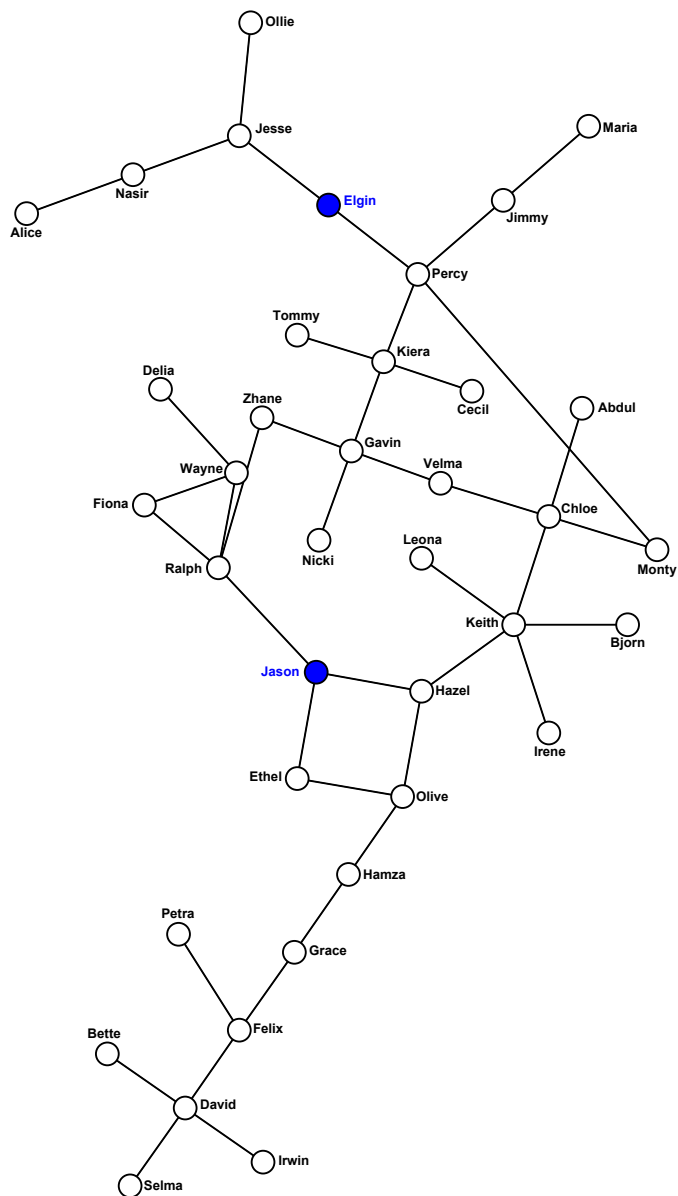


Figure B.26: t3lp - Formula One team-mates, 2004-2006

B.2.5 Task Type 4

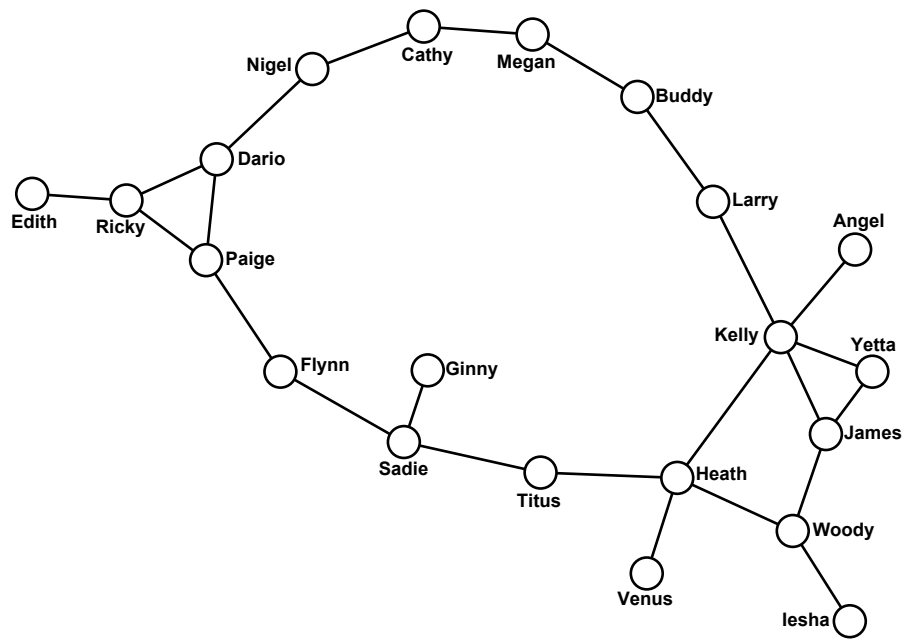


Figure B.27: t4sf - Formula One team-mates, 2010-2012

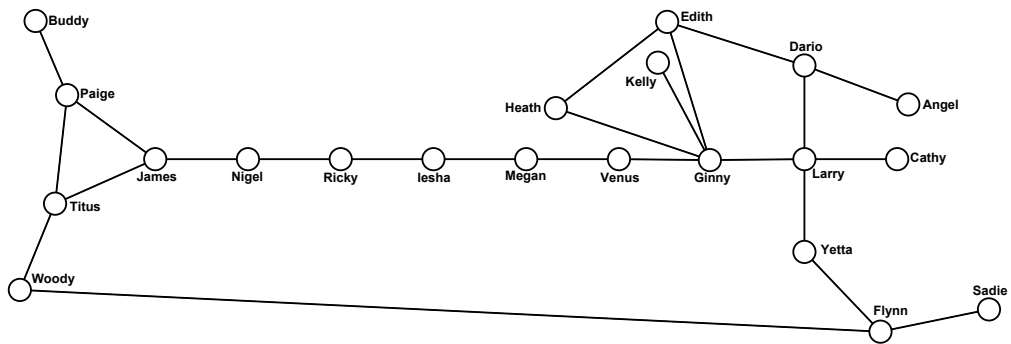


Figure B.28: t4sp - Formula One team-mates, 2010-2012

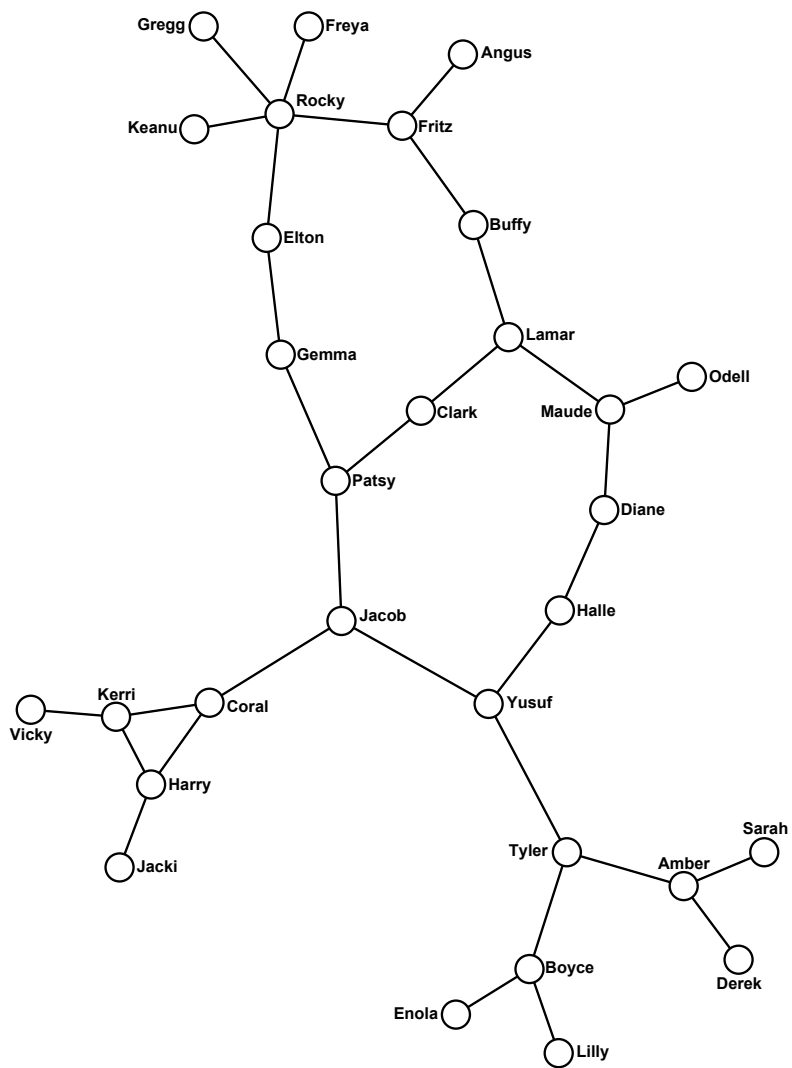


Figure B.29: t4mf - Formula One team-mates, 1996-1998

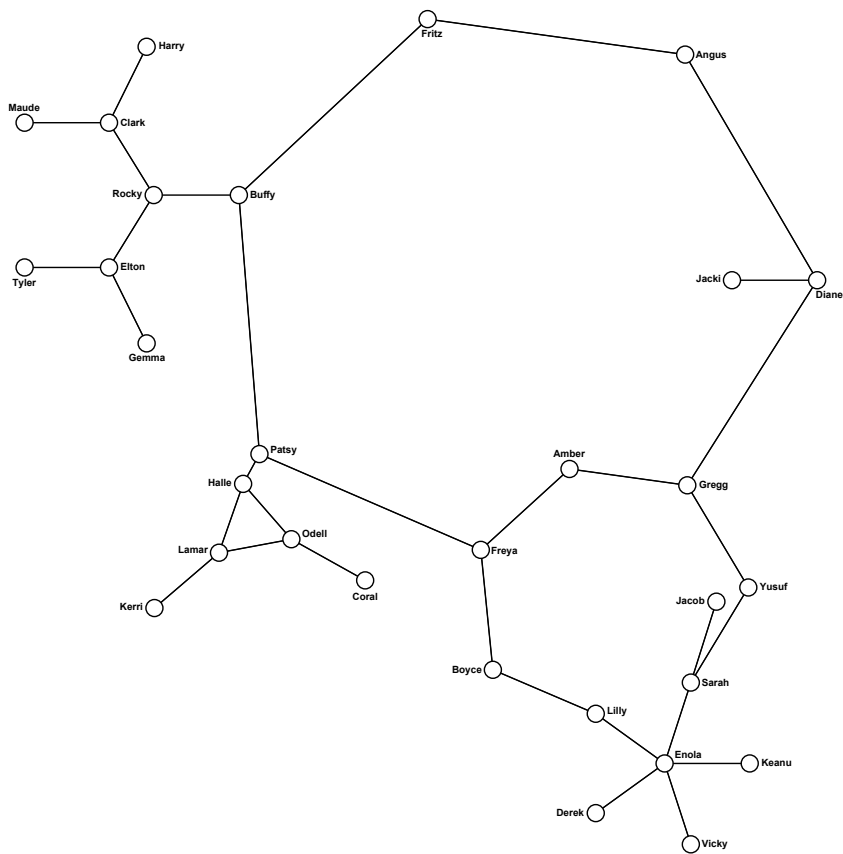


Figure B.30: t4mp - Formula One team-mates, 1996-1998

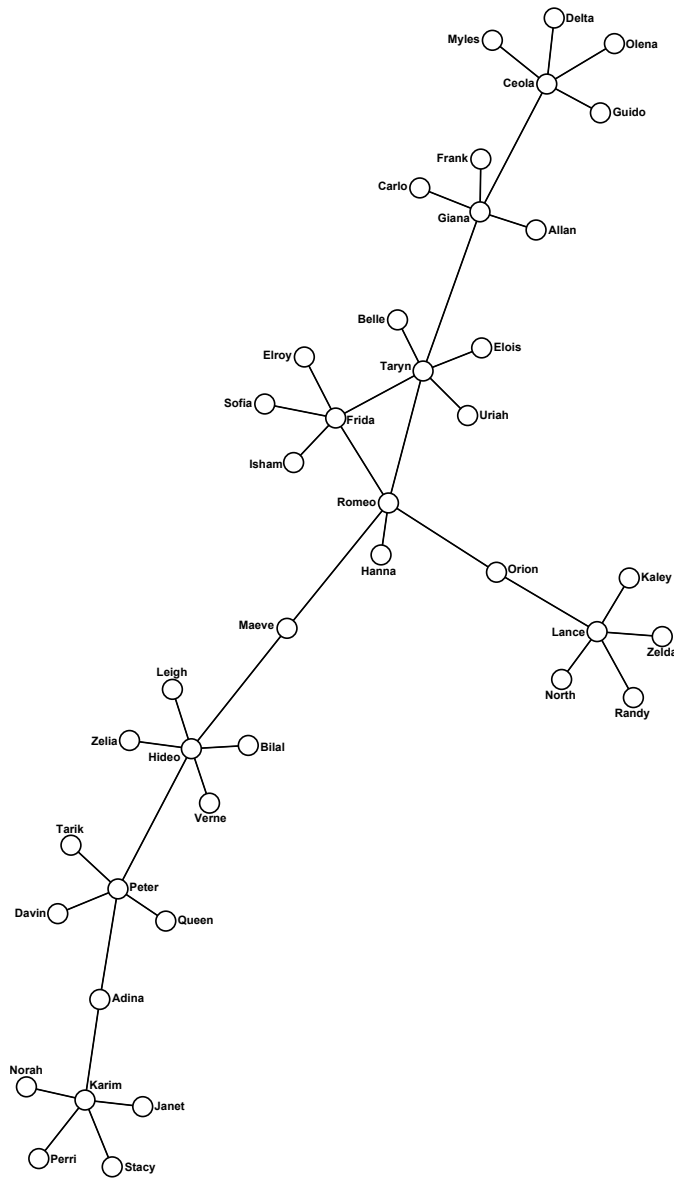


Figure B.31: t4lf - 2005 Best Leading and Supporting Actor and Actress Nominations

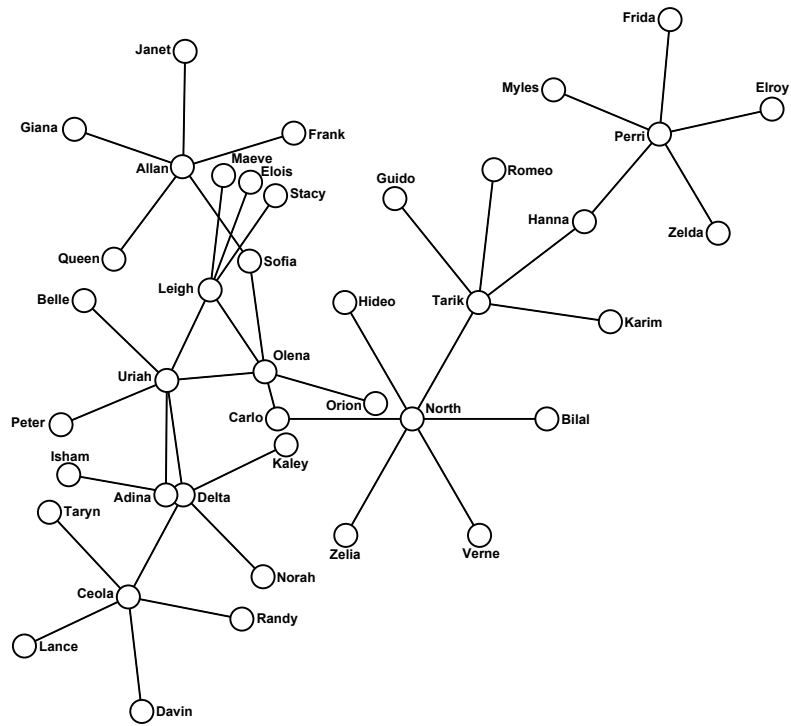


Figure B.32: t4lp - 2005 Best Leading and Supporting Actor and Actress Nominees

B.3 Questions and Answers

These are the questions asked for both the practice and main questions. All the answers presented to the participant, along with the correct answer. The Question IDs refer to the images used in Section B.2.

B.3.1 Practice Questions

Unlike in the main study, the answer and an explanation is shown to the user once they have confirmed an answer in the practice section.

Table B.1: Practice Questions Asked

Question ID	Text	A1	A2	A3	A4	Correct Answer
p1 (Fig B.1)	How many people are directly connected to Billy (shown in blue)?	3	4	5	6	The correct answer is 3. Billy is directly connected to Carla, Doris & Edith.
Continued on next page						

Table B.1 – continued from previous page

Question ID	Text	A1	A2	A3	A4	Correct Answer
p2 (Fig B.2)	What is the minimum number of people that must be removed in order to disconnect Flynn & Emile (both shown in blue) so that there is no route between them?	1	2	3	4	The correct answer is 2. Harvey would need to be removed, plus one of Drake or Isiah.
p3 (Fig B.3)	How many other people are there in the shortest route between Elgin & Marco (both shown in blue)?	2	3	4	5	The correct answer is 2 (Bambi & Harry). You do not include Elgin & Marco in the count.
Continued on next page						

Table B.1 – continued from previous page

Question ID	Text	A1	A2	A3	A4	Correct Answer
p3a (Fig B.4)	How many triangles are there in the diagram? (A triangle is where 3 people are connected to each other)	1	2	3	4	The correct answer is 1. Carla, Freda & Ellen are in a triangle.
p4 (Fig B.5)	How many people are directly connected to Emily (shown in blue)?	3	4	5	6	The correct answer is 6. Emily is connected to Philo, Lexus, Gregg, Trent, Vesla & Sandy
p5 (Fig B.6)	What is the minimum number of people that must be removed in order to disconnect Ottis & Wyman (both shown in blue) so that there is no route between them?	1	2	3	4	The correct answer is 1. One of Caryn or Brody would need to be removed.

Continued on next page

Table B.1 – continued from previous page

Question ID	Text	A1	A2	A3	A4	Correct Answer
p6 (Fig B.7)	How many other people are there in the shortest route between Deana & Texas (both shown in blue)?	2	3	4	5	The correct answer is 4. (Jayne -; Marge -; Freda -; Vance). You do not include Deana & Texas in the count.
p7 (Fig B.8)	How many triangles are there in the diagram? (A triangle is where 3 people are connected to each other)	1	2	3	4	The correct answer is 2. Irene, Galen & Nelle are in a triangle, as are Haley, Dante & Irene.

B.3.2 Main Study

The correct answer is not shown to the user, and refers to the correct option rather than the actual value.

Table B.2: Questions Asked

Question ID	Text	A1	A2	A3	A4	Correct Answer
t1sf (Fig B.9)	How many people are directly connected to Zayne (shown in blue)?	3	4	5	6	A3
t1sp (Fig B.10)	How many people are directly connected to Jamal (shown in blue)?	3	4	5	6	A3
t1mf (Fig B.11)	How many people are directly connected to Timmy (shown in blue)?	3	4	5	6	A2
t1mp (Fig B.12)	How many people are directly connected to Micah (shown in blue)?	3	4	5	6	A4
t1lf (Fig B.13)	How many people are directly connected to Orene (shown in blue)?	3	4	5	6	A1
Continued on next page						

Table B.2 – continued from previous page

Question ID	Text	A1	A2	A3	A4	Correct Answer
t1lp (Fig B.14)	How many people are directly connected to Annie (shown in blue)?	3	4	5	6	A1
t2sf (Fig B.15)	What is the minimum number of people that must be removed in order to disconnect Dolph & Vince (both shown in blue) so that there is no route between them?	1	2	3	4	A1
t2sp (Fig B.16)	What is the minimum number of people that must be removed in order to disconnect Ariel & Gerry (both shown in blue) so that there is no route between them?	1	2	3	4	A2
Continued on next page						

Table B.2 – continued from previous page

Question ID	Text	A1	A2	A3	A4	Correct Answer
t2mf (Fig B.17)	What is the minimum number of people that must be removed in order to disconnect Dejon & Rusty (both shown in blue) so that there is no route between them?	1	2	3	4	A1
t2mp (Fig B.18)	What is the minimum number of people that must be removed in order to disconnect Naomi & Hilda (both shown in blue) so that there is no route between them?	1	2	3	4	A1
t2lf (Fig B.19)	What is the minimum number of people that must be removed in order to disconnect Ruben & Glynn (both shown in blue) so that there is no route between them?	1	2	3	4	A3
Continued on next page						

Table B.2 – continued from previous page

Question ID	Text	A1	A2	A3	A4	Correct Answer
t2lp (Fig B.20)	What is the minimum number of people that must be removed in order to disconnect Andre & Owens (both shown in blue) so that there is no route between them?	1	2	3	4	A3
t3sf (Fig B.21)	How many other people are there in the shortest route between Tegan & Nelly (both shown in blue)?	3	4	5	6	A2
t3sp (Fig B.22)	How many other people are there in the shortest route between Isaac & Doris (both shown in blue)?	3	4	5	6	A1
t3mf (Fig B.23)	How many other people are there in the shortest route between Helga & Sherri (both shown in blue)?	3	4	5	6	A3
Continued on next page						

Table B.2 – continued from previous page

Question ID	Text	A1	A2	A3	A4	Correct Answer
t3mp (Fig B.24)	How many other people are there in the shortest route between Glenn & Lloyd (both shown in blue)?	3	4	5	6	A4
t3lf (Fig B.25)	How many other people are there in the shortest route between Abdul & Maria (both shown in blue)?	4	5	6	7	A2
t3lp (Fig B.26)	How many other people are there in the shortest route between Elgin & Jason (both shown in blue)?	4	5	6	7	A2
t4sf (Fig B.27)	How many triangles are there in the diagram? (A triangle is where 3 people are connected to each other)	1	2	3	4	A2
Continued on next page						

Table B.2 – continued from previous page

Question ID	Text	A1	A2	A3	A4	Correct Answer
t4sp (Fig B.28)	How many triangles are there in the diagram? (A triangle is where 3 people are connected to each other)	1	2	3	4	A2
t4mf (Fig B.29)	How many triangles are there in the diagram? (A triangle is where 3 people are connected to each other)	1	2	3	4	A1
t4mp (Fig B.30)	How many triangles are there in the diagram? (A triangle is where 3 people are connected to each other)	1	2	3	4	A1
t4lf (Fig B.31)	How many triangles are there in the diagram? (A triangle is where 3 people are connected to each other)	1	2	3	4	A1
Continued on next page						

Table B.2 – continued from previous page

Question ID	Text	A1	A2	A3	A4	Correct Answer
t4lp (Fig B.32)	How many triangles are there in the diagram? (A triangle is where 3 people are connected to each other)	1	2	3	4	A1

B.4 Preference Sheet

Shown below are the three pairings shown to participants to indicate their preference.

B.4.1 Pair 1

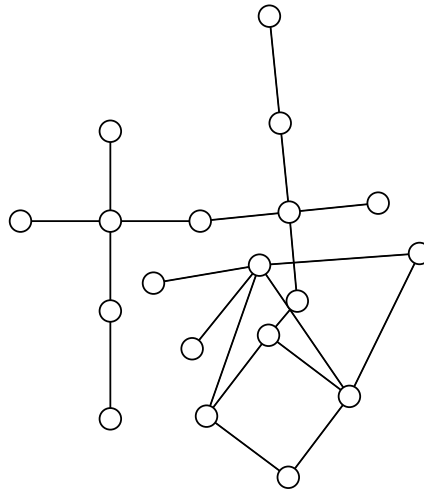


Figure B.33: Preference Pair 1, Pattern - Formula One team-mates, 2002-2004

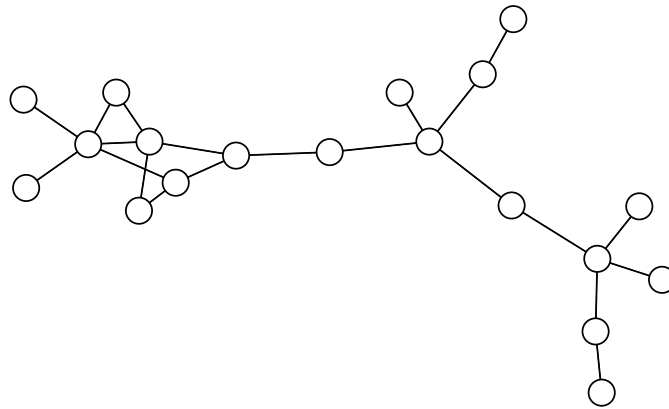


Figure B.34: Preference Pair 1, Force - Formula One team-mates, 2002-2004

B.4.2 Pair 2

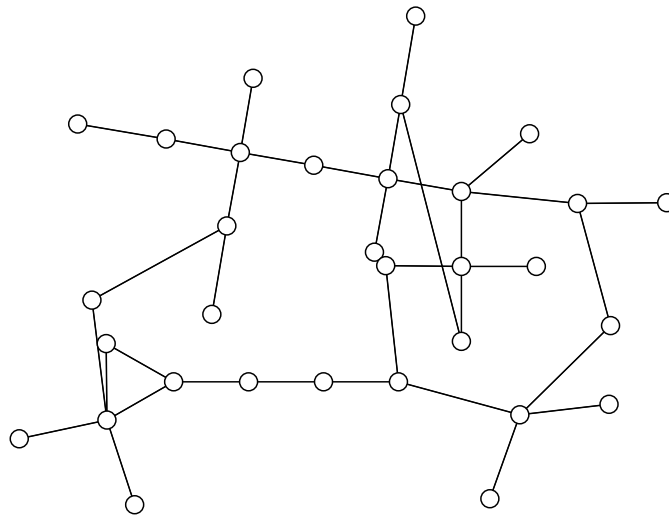


Figure B.35: Preference Pair 2, Pattern - Formula One team-mates, 2005-2007

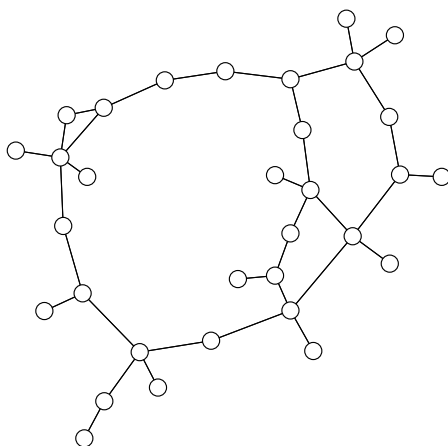


Figure B.36: Preference Pair 2, Force - Formula One team-mates, 2005-2007

B.4.3 Pair 3

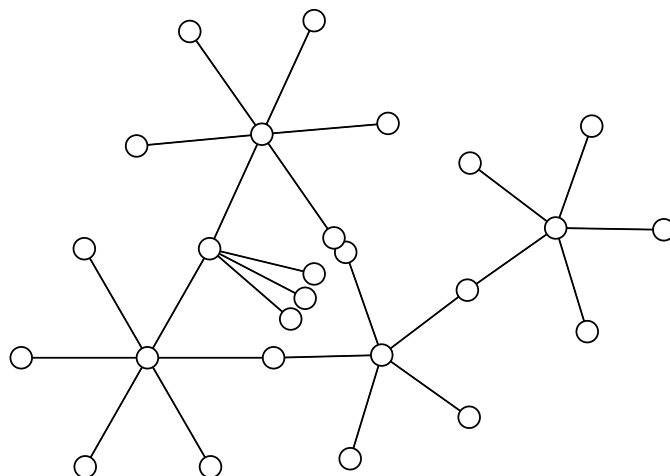


Figure B.37: Preference Pair 3, Pattern - 2007 Best Leading Actor and Actress Nominees

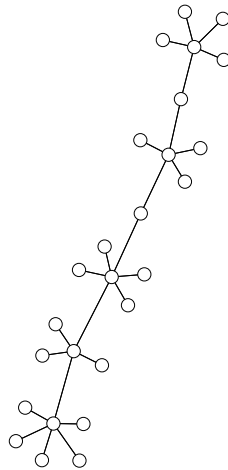


Figure B.38: Preference Pair 3, Force - 2007 Best Leading Actor and Actress Nominees

B.5 Results

Listed below are all the results for the main and preference study. Personal information has been removed. The user IDs in the main study results relate to the user IDs in the preference study. The question IDs and answers in the main study results correlate to the examples and questions given in Sections B.2 and B.3.

B.5.1 Main Study

Table B.3: Main study results

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t2lf	a3	4	27370	1	2	l	f
t3lf	a2	2	34921	1	3	l	f
t2sf	a1	1	24610	1	2	s	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t1lf	a1	1	14983	1	1	l	f
t1sf	a3	3	7168	1	1	s	f
t3sf	a2	2	12472	1	3	s	f
t3mp	a4	4	14322	1	3	m	p
t4mf	a1	1	14984	1	4	m	f
t1mp	a4	4	14095	1	1	m	p
t2sp	a2	2	19496	1	2	s	p
t4lf	a1	1	15244	1	4	l	f
t3lp	a2	2	20622	1	3	l	p
t2mp	a1	1	15088	1	2	m	p
t2lp	a3	3	23393	1	2	l	p
t3sp	a1	1	9313	1	3	s	p
t1sp	a3	3	7087	1	1	s	p
t4sp	a2	2	6977	1	4	s	p
t4lp	a1	1	8264	1	4	l	p
t1lp	a1	1	7168	1	1	l	p
t2mf	a1	1	10311	1	2	m	f
t3mf	a3	3	11331	1	3	m	f
t1mf	a2	2	7455	1	1	m	f
t4mp	a1	1	8505	1	4	m	p
t4sf	a2	2	5327	1	4	s	f
t2lp	a3	1	22754	2	2	l	p
t3lf	a2	2	18159	2	3	l	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t1mf	a2	2	12536	2	1	m	f
t3mf	a3	3	10792	2	3	m	f
t2mf	a1	4	16352	2	2	m	f
t3sf	a2	2	10616	2	3	s	f
t4lp	a1	1	18599	2	4	l	p
t1lp	a1	1	7337	2	1	l	p
t2sf	a1	2	17488	2	2	s	f
t1mp	a4	4	14312	2	1	m	p
t1sp	a3	3	6953	2	1	s	p
t4mp	a1	1	11224	2	4	m	p
t2mp	a1	4	55098	2	2	m	p
t4sp	a2	2	12049	2	4	s	p
t4lf	a1	1	13087	2	4	l	f
t3mp	a4	4	15512	2	3	m	p
t2lf	a3	2	20491	2	2	l	f
t1sf	a3	3	6784	2	1	s	f
t3sp	a1	1	13056	2	3	s	p
t4mf	a1	1	11632	2	4	m	f
t1lf	a1	1	5832	2	1	l	f
t2sp	a2	2	14768	2	2	s	p
t4sf	a2	2	7080	2	4	s	f
t3lp	a2	2	18192	2	3	l	p
t3sp	a1	1	13666	3	3	s	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t4lf	a1	1	7728	3	4	l	f
t4sf	a2	2	14407	3	4	s	f
t2mf	a1	1	26960	3	2	m	f
t2sp	a2	1	46708	3	2	s	p
t1sf	a3	3	9281	3	1	s	f
t2lf	a3	3	26671	3	2	l	f
t3mf	a3	3	9528	3	3	m	f
t3lp	a2	2	23578	3	3	l	p
t1mf	a2	2	9801	3	1	m	f
t4mp	a1	1	14087	3	4	m	p
t2sf	a1	1	23872	3	2	s	f
t1lp	a1	1	6928	3	1	l	p
t1sp	a3	3	6554	3	1	s	p
t2mp	a1	1	15729	3	2	m	p
t3sf	a2	2	13695	3	3	s	f
t1mp	a4	4	12721	3	1	m	p
t2lp	a3	3	14632	3	2	l	p
t4sp	a2	2	12490	3	4	s	p
t4lp	a1	1	20865	3	4	l	p
t3mp	a4	4	14183	3	3	m	p
t1lf	a1	1	9176	3	1	l	f
t4mf	a1	1	17274	3	4	m	f
t3lf	a2	2	19096	3	3	l	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t3lp	a2	2	27800	4	3	l	p
t4lf	a1	1	12719	4	4	l	f
t3mp	a4	4	18864	4	3	m	p
t2mf	a1	1	20168	4	2	m	f
t4sf	a2	2	9951	4	4	s	f
t4mp	a1	1	12673	4	4	m	p
t1sp	a3	3	7896	4	1	s	p
t2sf	a1	1	18288	4	2	s	f
t3sp	a1	1	12039	4	3	s	p
t2lp	a3	3	20256	4	2	l	p
t1lf	a1	1	11839	4	1	l	f
t4sp	a2	2	9737	4	4	s	p
t4mf	a1	1	13055	4	4	m	f
t1mf	a2	2	10264	4	1	m	f
t2mp	a1	1	13632	4	2	m	p
t1sf	a3	3	10920	4	1	s	f
t3mf	a3	3	14833	4	3	m	f
t3sf	a2	2	9160	4	3	s	f
t2sp	a2	2	39030	4	2	s	p
t1lp	a1	1	8224	4	1	l	p
t4lp	a1	1	20264	4	4	l	p
t3lf	a2	4	12919	4	3	l	f
t2lf	a3	3	32160	4	2	l	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t1mp	a4	4	8592	4	1	m	p
t4sf	a2	2	11024	5	4	s	f
t2lf	a3	3	49347	5	2	l	f
t2mf	a1	1	12952	5	2	m	f
t3mf	a3	3	12168	5	3	m	f
t2sp	a2	2	43566	5	2	s	p
t3sf	a2	2	14440	5	3	s	f
t1mf	a2	2	9192	5	1	m	f
t1lf	a1	1	7584	5	1	l	f
t2lp	a3	4	81763	5	2	l	p
t3lp	a2	2	28663	5	3	l	p
t4mf	a1	1	8505	5	4	m	f
t4lf	a1	1	8935	5	4	l	f
t2mp	a1	1	15202	5	2	m	p
t4sp	a2	2	11865	5	4	s	p
t3mp	a4	4	24831	5	3	m	p
t2sf	a1	1	11401	5	2	s	f
t1sp	a3	3	6799	5	1	s	p
t1mp	a4	4	15879	5	1	m	p
t4mp	a1	1	10288	5	4	m	p
t1lp	a1	1	5200	5	1	l	p
t4lp	a1	1	36264	5	4	l	p
t3lf	a2	2	29083	5	3	l	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t3sp	a1	1	15712	5	3	s	p
t1sf	a3	3	6927	5	1	s	f
t4sf	a2	2	11704	6	4	s	f
t1sf	a3	3	10272	6	1	s	f
t2lf	a3	1	28073	6	2	l	f
t2mp	a1	2	36663	6	2	m	p
t4lp	a1	1	29694	6	4	l	p
t2sp	a2	2	61152	6	2	s	p
t1mf	a2	2	15743	6	1	m	f
t3mp	a4	4	25637	6	3	m	p
t4mf	a1	1	9889	6	4	m	f
t3sf	a2	2	17752	6	3	s	f
t2mf	a1	1	18247	6	2	m	f
t1sp	a3	3	7463	6	1	s	p
t3lf	a2	2	28807	6	3	l	f
t4lf	a1	1	15048	6	4	l	f
t1lp	a1	1	9871	6	1	l	p
t4mp	a1	1	12270	6	4	m	p
t2sf	a1	1	40256	6	2	s	f
t3sp	a1	1	18442	6	3	s	p
t2lp	a3	2	52095	6	2	l	p
t3lp	a2	2	26101	6	3	l	p
t1mp	a4	4	19376	6	1	m	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t3mf	a3	3	11695	6	3	m	f
t4sp	a2	2	7616	6	4	s	p
t1lf	a1	1	9605	6	1	l	f
t1mf	a2	2	20384	7	1	m	f
t2lf	a3	1	45350	7	2	l	f
t4mf	a1	1	19167	7	4	m	f
t2mp	a1	1	33495	7	2	m	p
t1lp	a1	1	14232	7	1	l	p
t4sp	a2	2	13823	7	4	s	p
t1sf	a3	3	14856	7	1	s	f
t2sp	a2	2	25056	7	2	s	p
t1mp	a4	4	18216	7	1	m	p
t4lp	a1	1	35598	7	4	l	p
t3lf	a2	3	44023	7	3	l	f
t2mf	a1	1	18230	7	2	m	f
t3sf	a2	3	24008	7	3	s	f
t3mp	a4	4	38207	7	3	m	p
t4sf	a2	2	16135	7	4	s	f
t2lp	a3	3	42919	7	2	l	p
t4lf	a1	1	22872	7	4	l	f
t1sp	a3	3	8744	7	1	s	p
t4mp	a1	1	21151	7	4	m	p
t2sf	a1	1	11848	7	2	s	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t3sp	a1	1	23591	7	3	s	p
t3lp	a2	2	27065	7	3	l	p
t1lf	a1	1	8743	7	1	l	f
t3mf	a3	3	15408	7	3	m	f
t1lf	a1	1	7759	8	1	l	f
t2lp	a3	2	25583	8	2	l	p
t4sf	a2	2	7063	8	4	s	f
t2mf	a1	1	14504	8	2	m	f
t3lp	a2	2	18319	8	3	l	p
t1lp	a1	1	8601	8	1	l	p
t4lp	a1	1	14541	8	4	l	p
t2sf	a1	3	15272	8	2	s	f
t3mp	a4	4	13232	8	3	m	p
t1sf	a3	3	6391	8	1	s	f
t3sp	a1	1	16289	8	3	s	p
t3lf	a2	2	16609	8	3	l	f
t4mp	a1	1	13056	8	4	m	p
t2lf	a3	2	23327	8	2	l	f
t4lf	a1	1	11013	8	4	l	f
t1mp	a4	4	8392	8	1	m	p
t2sp	a2	2	13064	8	2	s	p
t2mp	a1	1	14224	8	2	m	p
t1sp	a3	3	8104	8	1	s	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t3mf	a3	3	10748	8	3	m	f
t3sf	a2	2	11224	8	3	s	f
t4mf	a1	1	9007	8	4	m	f
t1mf	a2	2	7873	8	1	m	f
t4sp	a2	2	11599	8	4	s	p
t2mp	a1	1	20050	9	2	m	p
t1lp	a1	1	13617	9	1	l	p
t1sp	a3	3	7385	9	1	s	p
t3sf	a2	2	19485	9	3	s	f
t4sf	a2	2	10436	9	4	s	f
t1mp	a4	4	8596	9	1	m	p
t3mf	a3	3	13651	9	3	m	f
t4mf	a1	1	10432	9	4	m	f
t3lp	a2	2	24870	9	3	l	p
t2mf	a1	1	21901	9	2	m	f
t4lf	a1	1	14218	9	4	l	f
t1lf	a1	1	9157	9	1	l	f
t2lf	a3	4	41877	9	2	l	f
t1mf	a2	2	7949	9	1	m	f
t2sp	a2	2	25037	9	2	s	p
t3mp	a4	4	24267	9	3	m	p
t4mp	a1	1	9763	9	4	m	p
t3sp	a1	1	15765	9	3	s	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t3lf	a2	3	26485	9	3	l	f
t4lp	a1	1	29561	9	4	l	p
t1sf	a3	3	9385	9	1	s	f
t4sp	a2	2	14799	9	4	s	p
t2sf	a1	1	21049	9	2	s	f
t2lp	a3	2	30249	9	2	l	p
t1mf	a2	2	10194	10	1	m	f
t3sp	a1	1	9895	10	3	s	p
t1lf	a1	1	12113	10	1	l	f
t2sp	a2	2	34179	10	2	s	p
t2lf	a3	4	31671	10	2	l	f
t1sp	a3	3	11825	10	1	s	p
t3lp	a2	2	30405	10	3	l	p
t4sf	a2	2	13271	10	4	s	f
t4lp	a1	2	30315	10	4	l	p
t1lp	a1	1	10111	10	1	l	p
t2mp	a1	1	16057	10	2	m	p
t3sf	a2	2	10457	10	3	s	f
t3mp	a4	4	11478	10	3	m	p
t4mp	a1	1	15486	10	4	m	p
t3lf	a2	2	17583	10	3	l	f
t1mp	a4	1	8735	10	1	m	p
t2lp	a3	4	42485	10	2	l	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t2sf	a1	1	22592	10	2	s	f
t4lf	a1	1	8847	10	4	l	f
t4sp	a2	2	10001	10	4	s	p
t3mf	a3	3	12540	10	3	m	f
t1sf	a3	3	7632	10	1	s	f
t2mf	a1	1	17959	10	2	m	f
t4mf	a1	1	11400	10	4	m	f
t3lp	a2	2	28679	11	3	l	p
t2sf	a1	1	27519	11	2	s	f
t3mp	a4	4	15011	11	3	m	p
t1lp	a1	1	17079	11	1	l	p
t1mf	a2	2	7519	11	1	m	f
t2mp	a1	1	42844	11	2	m	p
t4sf	a2	2	13279	11	4	s	f
t1sf	a3	3	10768	11	1	s	f
t3lf	a2	2	36741	11	3	l	f
t2lp	a3	3	13472	11	2	l	p
t3sp	a1	1	20080	11	3	s	p
t4mf	a1	1	16016	11	4	m	f
t2sp	a2	2	30659	11	2	s	p
t4lp	a1	1	27359	11	4	l	p
t1lf	a1	1	8888	11	1	l	f
t3mf	a3	3	15664	11	3	m	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t2lf	a3	3	24283	11	2	l	f
t3sf	a2	2	12912	11	3	s	f
t1mp	a4	4	13608	11	1	m	p
t4sp	a2	2	7832	11	4	s	p
t4mp	a1	1	16192	11	4	m	p
t2mf	a1	1	12340	11	2	m	f
t1sp	a3	3	7160	11	1	s	p
t4lf	a1	1	27056	11	4	l	f
t3lp	a2	2	16943	12	3	l	p
t1sf	a3	2	4689	12	1	s	f
t2mp	a1	1	9976	12	2	m	p
t1lf	a1	1	4936	12	1	l	f
t3sp	a1	1	11211	12	3	s	p
t1sp	a3	3	5160	12	1	s	p
t2sp	a2	1	11816	12	2	s	p
t2lp	a3	1	7280	12	2	l	p
t4sf	a2	2	7960	12	4	s	f
t4lp	a1	2	20669	12	4	l	p
t3mp	a4	4	10712	12	3	m	p
t1mp	a4	4	8226	12	1	m	p
t3sf	a2	2	8286	12	3	s	f
t4mp	a1	1	8720	12	4	m	p
t2lf	a3	1	12071	12	2	l	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t4sp	a2	2	9429	12	4	s	p
t3lf	a2	2	23679	12	3	l	f
t2sf	a1	1	7265	12	2	s	f
t2mf	a1	1	10663	12	2	m	f
t3mf	a3	3	8432	12	3	m	f
t1mf	a2	2	7503	12	1	m	f
t4mf	a1	1	6822	12	4	m	f
t1lp	a1	1	4943	12	1	l	p
t4lf	a1	1	7424	12	4	l	f
t1sf	a3	3	8849	13	1	s	f
t4lf	a1	1	11808	13	4	l	f
t3lf	a2	2	18016	13	3	l	f
t2mp	a1	1	32882	13	2	m	p
t1mf	a2	2	13087	13	1	m	f
t1lp	a1	1	6416	13	1	l	p
t3sp	a1	1	7816	13	3	s	p
t3mf	a3	3	9456	13	3	m	f
t2lf	a3	3	65507	13	2	l	f
t4sf	a2	2	8049	13	4	s	f
t4lp	a1	2	56370	13	4	l	p
t1sp	a3	3	13280	13	1	s	p
t2sp	a2	2	15544	13	2	s	p
t4mp	a1	1	8896	13	4	m	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t3mp	a4	4	13496	13	3	m	p
t2mf	a1	1	11114	13	2	m	f
t1mp	a4	4	6695	13	1	m	p
t3sf	a2	2	9240	13	3	s	f
t4sp	a2	2	8088	13	4	s	p
t3lp	a2	2	15623	13	3	l	p
t4mf	a1	1	5528	13	4	m	f
t2sf	a1	1	7929	13	2	s	f
t2lp	a3	3	17179	13	2	l	p
t1lf	a1	1	6975	13	1	l	f
t4sp	a2	2	9720	14	4	s	p
t3sf	a2	2	18351	14	3	s	f
t3mf	a3	3	11072	14	3	m	f
t4mp	a1	1	8024	14	4	m	p
t2lf	a3	3	39867	14	2	l	f
t1lp	a1	1	9720	14	1	l	p
t4lp	a1	1	22137	14	4	l	p
t3lf	a2	2	30472	14	3	l	f
t2sf	a1	1	12408	14	2	s	f
t1sf	a3	3	7258	14	1	s	f
t2mf	a1	1	9946	14	2	m	f
t1mf	a2	2	7856	14	1	m	f
t3sp	a1	1	14768	14	3	s	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t4sf	a2	2	7376	14	4	s	f
t4mf	a1	1	9800	14	4	m	f
t4lf	a1	1	20553	14	4	l	f
t3mp	a4	4	23088	14	3	m	p
t1lf	a1	1	6816	14	1	l	f
t2lp	a3	2	34924	14	2	l	p
t3lp	a2	2	18615	14	3	l	p
t1mp	a4	4	11231	14	1	m	p
t1sp	a3	3	5961	14	1	s	p
t2sp	a2	2	18312	14	2	s	p
t2mp	a1	1	12161	14	2	m	p
t3lf	a2	3	23255	15	3	l	f
t1mf	a2	2	8793	15	1	m	f
t1lp	a1	1	7025	15	1	l	p
t2mf	a1	1	19352	15	2	m	f
t4sp	a2	2	13656	15	4	s	p
t3lp	a2	2	23098	15	3	l	p
t4mf	a1	1	14241	15	4	m	f
t2lf	a3	3	40568	15	2	l	f
t1sf	a3	3	8937	15	1	s	f
t3sf	a2	2	20280	15	3	s	f
t1mp	a4	4	18920	15	1	m	p
t4lp	a1	1	21228	15	4	l	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t2sf	a1	1	17680	15	2	s	f
t4mp	a1	1	17072	15	4	m	p
t2mp	a1	1	14854	15	2	m	p
t2lp	a3	3	13066	15	2	l	p
t3mf	a3	3	12751	15	3	m	f
t4sf	a2	2	8664	15	4	s	f
t1lf	a1	1	5513	15	1	l	f
t3sp	a1	1	10583	15	3	s	p
t4lf	a1	1	13280	15	4	l	f
t1sp	a3	3	5955	15	1	s	p
t2sp	a2	2	26744	15	2	s	p
t3mp	a4	4	23543	15	3	m	p
t3sf	a2	2	25479	16	3	s	f
t4lp	a1	2	35759	16	4	l	p
t1mp	a4	1	30368	16	1	m	p
t4mp	a1	1	15160	16	4	m	p
t2mf	a1	1	36864	16	2	m	f
t1mf	a2	2	19703	16	1	m	f
t3lf	a2	2	61995	16	3	l	f
t1lp	a1	1	13598	16	1	l	p
t1sp	a3	3	13744	16	1	s	p
t2sf	a1	1	29665	16	2	s	f
t2lf	a3	4	93576	16	2	l	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t4mf	a1	1	12064	16	4	m	f
t3mf	a3	3	22008	16	3	m	f
t2mp	a1	2	41004	16	2	m	p
t4sf	a2	2	10952	16	4	s	f
t1sf	a3	3	9008	16	1	s	f
t4lf	a1	1	10904	16	4	l	f
t1lf	a1	1	6408	16	1	l	f
t2lp	a3	1	18432	16	2	l	p
t2sp	a2	2	30321	16	2	s	p
t3lp	a2	3	45088	16	3	l	p
t3sp	a1	1	11337	16	3	s	p
t3mp	a4	4	72290	16	3	m	p
t4sp	a2	2	19320	16	4	s	p
t2lp	a3	3	8679	17	2	l	p
t2mf	a1	1	13249	17	2	m	f
t4sf	a2	2	7952	17	4	s	f
t1lp	a1	1	9441	17	1	l	p
t2sf	a1	1	12103	17	2	s	f
t4lp	a1	2	17719	17	4	l	p
t1sp	a3	3	6904	17	1	s	p
t1mf	a2	2	6417	17	1	m	f
t4mf	a1	1	9394	17	4	m	f
t2mp	a1	1	18144	17	2	m	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t3lp	a2	2	18328	17	3	l	p
t3mp	a4	4	14593	17	3	m	p
t3sp	a1	1	9865	17	3	s	p
t2sp	a2	2	17776	17	2	s	p
t4sp	a2	2	5800	17	4	s	p
t1sf	a3	3	8439	17	1	s	f
t4lf	a1	1	8392	17	4	l	f
t1mp	a4	4	12599	17	1	m	p
t2lf	a3	3	28450	17	2	l	f
t3sf	a2	2	10576	17	3	s	f
t3mf	a3	3	8416	17	3	m	f
t3lf	a2	2	13560	17	3	l	f
t4mp	a1	1	11395	17	4	m	p
t1lf	a1	1	9623	17	1	l	f
t2sp	a2	2	18416	18	2	s	p
t1mf	a2	2	12422	18	1	m	f
t1lf	a1	1	7904	18	1	l	f
t1sf	a3	3	8175	18	1	s	f
t3lp	a2	2	27036	18	3	l	p
t4lp	a1	1	26007	18	4	l	p
t2mp	a1	1	10329	18	2	m	p
t3sf	a2	2	14199	18	3	s	f
t4mf	a1	1	13069	18	4	m	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t3mf	a3	3	10145	18	3	m	f
t2sf	a1	1	10960	18	2	s	f
t1sp	a3	3	6655	18	1	s	p
t4sp	a2	2	12320	18	4	s	p
t1mp	a4	4	13113	18	1	m	p
t3lf	a2	2	23319	18	3	l	f
t2lp	a3	3	21934	18	2	l	p
t1lp	a1	1	9895	18	1	l	p
t2mf	a1	1	11473	18	2	m	f
t3mp	a4	4	16143	18	3	m	p
t4lf	a1	1	10804	18	4	l	f
t4mp	a1	1	12280	18	4	m	p
t3sp	a1	1	11735	18	3	s	p
t4sf	a2	2	10704	18	4	s	f
t2lf	a3	3	46171	18	2	l	f
t4sf	a2	2	11719	19	4	s	f
t2lf	a3	3	213110	19	2	l	f
t2mf	a1	1	20364	19	2	m	f
t4lf	a1	1	19632	19	4	l	f
t3mp	a4	4	16288	19	3	m	p
t3sf	a2	2	14536	19	3	s	f
t3lp	a2	2	45379	19	3	l	p
t1sf	a3	3	6841	19	1	s	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t1lp	a1	1	7924	19	1	l	p
t1mp	a4	4	6785	19	1	m	p
t4lp	a1	2	28838	19	4	l	p
t4mf	a1	1	12866	19	4	m	f
t2sp	a2	2	39594	19	2	s	p
t2mp	a1	1	20536	19	2	m	p
t1sp	a3	3	8080	19	1	s	p
t3mf	a3	3	17852	19	3	m	f
t3sp	a1	1	13472	19	3	s	p
t1lf	a1	1	8143	19	1	l	f
t4mp	a1	1	11169	19	4	m	p
t1mf	a2	2	4055	19	1	m	f
t2lp	a3	3	45813	19	2	l	p
t3lf	a2	2	26428	19	3	l	f
t4sp	a2	2	8649	19	4	s	p
t2sf	a1	1	17448	19	2	s	f
t1lf	a1	1	12328	20	1	l	f
t3sp	a1	1	11367	20	3	s	p
t1sp	a3	3	12032	20	1	s	p
t2sp	a2	2	31443	20	2	s	p
t1mf	a2	2	12160	20	1	m	f
t3lp	a2	2	40058	20	3	l	p
t2lf	a3	3	44522	20	2	l	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t4sp	a2	2	12735	20	4	s	p
t2mf	a1	1	29177	20	2	m	f
t4mf	a1	1	10751	20	4	m	f
t3sf	a2	2	18051	20	3	s	f
t1lp	a1	1	10151	20	1	l	p
t4lp	a1	1	54467	20	4	l	p
t2sf	a1	1	21567	20	2	s	f
t3lf	a2	2	24987	20	3	l	f
t3mf	a3	3	14239	20	3	m	f
t2lp	a3	3	69419	20	2	l	p
t2mp	a1	1	19929	20	2	m	p
t1sf	a3	3	9095	20	1	s	f
t1mp	a4	4	14469	20	1	m	p
t4sf	a2	2	14055	20	4	s	f
t4mp	a1	1	17784	20	4	m	p
t4lf	a1	1	17812	20	4	l	f
t3mp	a4	4	21808	20	3	m	p
t1mp	a4	4	18780	21	1	m	p
t2lf	a3	3	29296	21	2	l	f
t4sf	a2	2	9640	21	4	s	f
t2mp	a1	1	27371	21	2	m	p
t4mp	a1	1	11791	21	4	m	p
t2sf	a1	1	36031	21	2	s	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t1sf	a3	3	11152	21	1	s	f
t4lf	a1	1	18193	21	4	l	f
t1lp	a1	1	10672	21	1	l	p
t3mp	a4	3	20312	21	3	m	p
t2lp	a3	3	53292	21	2	l	p
t3lp	a2	2	38769	21	3	l	p
t3sf	a2	2	32090	21	3	s	f
t4mf	a1	1	7815	21	4	m	f
t1mf	a2	2	8583	21	1	m	f
t1sp	a3	3	8313	21	1	s	p
t2sp	a2	2	37658	21	2	s	p
t4lp	a1	1	25880	21	4	l	p
t3mf	a3	3	14927	21	3	m	f
t2mf	a1	1	18985	21	2	m	f
t3lf	a2	2	33490	21	3	l	f
t4sp	a2	2	13152	21	4	s	p
t1lf	a1	1	15328	21	1	l	f
t3sp	a1	1	15544	21	3	s	p
t3lp	a2	2	12011	22	3	l	p
t4mf	a1	1	7632	22	4	m	f
t1lf	a1	1	6504	22	1	l	f
t2sf	a1	1	13168	22	2	s	f
t4lp	a1	1	13831	22	4	l	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t2mp	a1	1	7248	22	2	m	p
t3mp	a4	4	14549	22	3	m	p
t1mp	a4	4	9128	22	1	m	p
t3sf	a2	2	10072	22	3	s	f
t1lp	a1	1	6088	22	1	l	p
t1sp	a3	3	7208	22	1	s	p
t2lp	a3	3	15703	22	2	l	p
t4mp	a1	1	6212	22	4	m	p
t3mf	a3	4	7279	22	3	m	f
t4sf	a2	2	6305	22	4	s	f
t2sp	a2	2	17520	22	2	s	p
t3lf	a2	2	16512	22	3	l	f
t3sp	a1	1	9004	22	3	s	p
t2lf	a3	3	11209	22	2	l	f
t1mf	a2	2	5703	22	1	m	f
t4lf	a1	1	7776	22	4	l	f
t2mf	a1	1	5935	22	2	m	f
t1sf	a3	3	9320	22	1	s	f
t4sp	a2	2	9647	22	4	s	p
t1mf	a2	2	12697	23	1	m	f
t2sp	a2	2	29659	23	2	s	p
t4mp	a1	1	17928	23	4	m	p
t4lp	a1	1	22744	23	4	l	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t3mf	a3	3	16571	23	3	m	f
t3sp	a1	1	11664	23	3	s	p
t2mf	a1	1	22240	23	2	m	f
t1lf	a1	1	8327	23	1	l	f
t1mp	a4	1	11394	23	1	m	p
t4sf	a2	2	17328	23	4	s	f
t3lp	a2	2	30880	23	3	l	p
t4mf	a1	1	11995	23	4	m	f
t1sf	a3	3	11240	23	1	s	f
t4lf	a1	1	18528	23	4	l	f
t2mp	a1	1	31570	23	2	m	p
t2lf	a3	3	51583	23	2	l	f
t2sf	a1	1	20443	23	2	s	f
t3sf	a2	2	12120	23	3	s	f
t1lp	a1	1	11703	23	1	l	p
t1sp	a3	3	7513	23	1	s	p
t3lf	a2	2	25705	23	3	l	f
t3mp	a4	4	20729	23	3	m	p
t2lp	a3	2	49090	23	2	l	p
t4sp	a2	2	15024	23	4	s	p
t4lf	a1	1	21967	24	4	l	f
t3sp	a1	1	22328	24	3	s	p
t2mp	a1	1	20547	24	2	m	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t1sp	a3	3	12328	24	1	s	p
t4sf	a2	2	15023	24	4	s	f
t1mp	a4	4	21196	24	1	m	p
t3mf	a3	3	21255	24	3	m	f
t2lp	a3	1	14184	24	2	l	p
t3lp	a2	2	41538	24	3	l	p
t4mf	a1	1	26280	24	4	m	f
t3sf	a2	2	18387	24	3	s	f
t2sf	a1	3	37335	24	2	s	f
t4sp	a2	2	10912	24	4	s	p
t1lp	a1	1	53378	24	1	l	p
t2lf	a3	1	45234	24	2	l	f
t3lf	a2	2	30203	24	3	l	f
t2mf	a1	1	18191	24	2	m	f
t3mp	a4	4	26303	24	3	m	p
t1mf	a2	2	23683	24	1	m	f
t4mp	a1	1	10287	24	4	m	p
t1sf	a3	3	16279	24	1	s	f
t2sp	a2	3	28451	24	2	s	p
t4lp	a1	2	23040	24	4	l	p
t1lf	a1	1	14520	24	1	l	f
t2mf	a1	1	31024	25	2	m	f
t2lf	a3	1	23184	25	2	l	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t1sp	a3	3	7848	25	1	s	p
t4sp	a2	2	8632	25	4	s	p
t3sp	a1	1	13984	25	3	s	p
t1mf	a2	2	9231	25	1	m	f
t3lp	a2	2	25281	25	3	l	p
t1lp	a1	1	5633	25	1	l	p
t2mp	a1	1	31600	25	2	m	p
t4lf	a1	1	8624	25	4	l	f
t4sf	a2	2	7480	25	4	s	f
t3mp	a4	4	20507	25	3	m	p
t1mp	a4	4	7095	25	1	m	p
t1sf	a3	3	6321	25	1	s	f
t4mf	a1	1	8384	25	4	m	f
t2sf	a1	1	10527	25	2	s	f
t1lf	a1	1	11057	25	1	l	f
t3lf	a2	2	32426	25	3	l	f
t2lp	a3	1	8872	25	2	l	p
t3mf	a3	3	11520	25	3	m	f
t4lp	a1	1	12927	25	4	l	p
t4mp	a1	1	9657	25	4	m	p
t2sp	a2	1	38979	25	2	s	p
t3sf	a2	2	13799	25	3	s	f
t3sp	a1	1	16052	26	3	s	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t1mf	a2	2	12119	26	1	m	f
t2sf	a1	2	39369	26	2	s	f
t2lp	a3	1	38530	26	2	l	p
t3lf	a2	2	28030	26	3	l	f
t1lp	a1	1	16616	26	1	l	p
t1sf	a3	3	8793	26	1	s	f
t2mf	a1	1	24475	26	2	m	f
t4lf	a1	1	21000	26	4	l	f
t4mf	a1	1	8184	26	4	m	f
t1mp	a4	3	11623	26	1	m	p
t3mf	a3	3	11856	26	3	m	f
t4sf	a2	2	9004	26	4	s	f
t2lf	a3	4	31328	26	2	l	f
t3lp	a2	2	19240	26	3	l	p
t1lf	a1	1	6991	26	1	l	f
t3sf	a2	2	12117	26	3	s	f
t2sp	a2	3	35087	26	2	s	p
t1sp	a3	3	9280	26	1	s	p
t3mp	a4	4	13200	26	3	m	p
t2mp	a1	1	33314	26	2	m	p
t4mp	a1	1	7872	26	4	m	p
t4sp	a2	1	12729	26	4	s	p
t4lp	a1	1	20783	26	4	l	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t2mp	a1	2	19010	27	2	m	p
t4mp	a1	1	11129	27	4	m	p
t2lp	a3	2	25983	27	2	l	p
t4sf	a2	2	7177	27	4	s	f
t3sf	a2	2	10632	27	3	s	f
t1lp	a1	1	13378	27	1	l	p
t4lp	a1	1	10936	27	4	l	p
t3mp	a4	4	18904	27	3	m	p
t1sf	a3	3	7961	27	1	s	f
t3lp	a2	2	16842	27	3	l	p
t2sp	a2	2	23527	27	2	s	p
t1mp	a4	4	7887	27	1	m	p
t4sp	a2	2	5840	27	4	s	p
t1lf	a1	1	4912	27	1	l	f
t2lf	a3	2	17080	27	2	l	f
t3mf	a3	3	8411	27	3	m	f
t1sp	a3	3	5760	27	1	s	p
t3sp	a1	1	6551	27	3	s	p
t4mf	a1	1	8425	27	4	m	f
t2sf	a1	1	29655	27	2	s	f
t1mf	a2	2	4347	27	1	m	f
t4lf	a1	1	6840	27	4	l	f
t3lf	a2	4	13288	27	3	l	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t2mf	a1	2	17743	27	2	m	f
t2sf	a1	2	27472	28	2	s	f
t2lf	a3	2	18513	28	2	l	f
t4lp	a1	1	22864	28	4	l	p
t3mp	a4	4	48842	28	3	m	p
t4mf	a1	1	17696	28	4	m	f
t1lf	a1	1	11080	28	1	l	f
t1sp	a3	3	14256	28	1	s	p
t2mp	a1	1	41921	28	2	m	p
t3lf	a2	1	55692	28	3	l	f
t3sf	a2	2	24842	28	3	s	f
t2lp	a3	1	19200	28	2	l	p
t4mp	a1	1	16808	28	4	m	p
t4lf	a1	1	12783	28	4	l	f
t4sp	a2	2	13427	28	4	s	p
t1mp	a4	4	27920	28	1	m	p
t3mf	a3	3	34499	28	3	m	f
t1lp	a1	1	14599	28	1	l	p
t3lp	a2	2	38623	28	3	l	p
t2mf	a1	2	25402	28	2	m	f
t1sf	a3	3	9073	28	1	s	f
t2sp	a2	2	23695	28	2	s	p
t4sf	a2	2	11843	28	4	s	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t1mf	a2	2	12208	28	1	m	f
t3sp	a1	1	15927	28	3	s	p
t3mf	a3	3	15976	29	3	m	f
t4sp	a2	1	17976	29	4	s	p
t4lf	a1	1	15144	29	4	l	f
t3sp	a1	1	10871	29	3	s	p
t3lf	a2	1	12752	29	3	l	f
t1lf	a1	1	10696	29	1	l	f
t4mp	a1	1	13456	29	4	m	p
t2sf	a1	1	26008	29	2	s	f
t1sf	a3	3	8824	29	1	s	f
t2mp	a1	1	10240	29	2	m	p
t4lp	a1	1	20257	29	4	l	p
t3mp	a4	4	26473	29	3	m	p
t4sf	a2	2	7503	29	4	s	f
t1mp	a4	2	15096	29	1	m	p
t2lp	a3	1	13403	29	2	l	p
t1sp	a3	3	12950	29	1	s	p
t1lp	a1	1	16223	29	1	l	p
t2sp	a2	2	18355	29	2	s	p
t4mf	a1	1	6865	29	4	m	f
t3lp	a2	2	17920	29	3	l	p
t3sf	a2	2	12792	29	3	s	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t1mf	a2	2	11583	29	1	m	f
t2lf	a3	3	24139	29	2	l	f
t2mf	a1	1	10871	29	2	m	f
t4sf	a2	2	10000	30	4	s	f
t2lp	a3	2	48716	30	2	l	p
t1mp	a4	4	14784	30	1	m	p
t1lf	a1	1	8719	30	1	l	f
t3sp	a1	1	24481	30	3	s	p
t3mp	a4	4	20452	30	3	m	p
t4lp	a1	1	14504	30	4	l	p
t2sp	a2	2	28223	30	2	s	p
t4sp	a2	2	9948	30	4	s	p
t4mp	a1	1	23055	30	4	m	p
t1sp	a3	3	9473	30	1	s	p
t2mf	a1	1	19785	30	2	m	f
t3lf	a2	2	30165	30	3	l	f
t4lf	a1	1	12759	30	4	l	f
t3sf	a2	2	17699	30	3	s	f
t2sf	a1	1	41152	30	2	s	f
t3mf	a3	3	13168	30	3	m	f
t1sf	a3	3	7346	30	1	s	f
t1lp	a1	1	8808	30	1	l	p
t4mf	a1	1	10711	30	4	m	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t1mf	a2	2	10640	30	1	m	f
t2mp	a1	1	48987	30	2	m	p
t3lp	a2	2	29278	30	3	l	p
t2lf	a3	3	46163	30	2	l	f
t4mp	a1	1	12616	31	4	m	p
t4lp	a1	1	22399	31	4	l	p
t1sf	a3	3	9347	31	1	s	f
t3lf	a2	3	40999	31	3	l	f
t2mp	a1	1	11738	31	2	m	p
t2sf	a1	1	12608	31	2	s	f
t1lf	a1	1	9111	31	1	l	f
t3sf	a2	2	23688	31	3	s	f
t2lf	a3	3	13304	31	2	l	f
t1sp	a3	3	6803	31	1	s	p
t1mp	a4	4	8808	31	1	m	p
t4lf	a1	1	14384	31	4	l	f
t4sf	a2	2	8328	31	4	s	f
t3mf	a3	3	13551	31	3	m	f
t2mf	a1	1	8971	31	2	m	f
t1lp	a1	1	9327	31	1	l	p
t4mf	a1	1	13616	31	4	m	f
t3lp	a2	2	36424	31	3	l	p
t1mf	a2	2	6157	31	1	m	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t2sp	a2	3	11568	31	2	s	p
t2lp	a3	3	18327	31	2	l	p
t3mp	a4	4	21016	31	3	m	p
t3sp	a1	1	13453	31	3	s	p
t4sp	a2	2	15840	31	4	s	p
t3mp	a4	4	11944	32	3	m	p
t2mf	a1	1	7920	32	2	m	f
t3sf	a2	2	20224	32	3	s	f
t1mp	a4	4	13211	32	1	m	p
t4lp	a1	1	11312	32	4	l	p
t2sp	a2	2	15536	32	2	s	p
t4sp	a2	2	10103	32	4	s	p
t3lp	a2	2	19138	32	3	l	p
t4mf	a1	1	5680	32	4	m	f
t1sf	a3	3	7488	32	1	s	f
t3mf	a3	3	19375	32	3	m	f
t2mp	a1	1	7568	32	2	m	p
t1lp	a1	1	10726	32	1	l	p
t2lp	a3	3	36732	32	2	l	p
t1mf	a2	2	9895	32	1	m	f
t4lf	a1	1	9209	32	4	l	f
t4mp	a1	1	10690	32	4	m	p
t2sf	a1	1	7751	32	2	s	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t3sp	a1	1	12217	32	3	s	p
t4sf	a2	2	7127	32	4	s	f
t1lf	a1	1	7001	32	1	l	f
t3lf	a2	2	22171	32	3	l	f
t1sp	a3	3	5478	32	1	s	p
t2lf	a3	3	62291	32	2	l	f
t2lp	a3	1	12415	33	2	l	p
t3mp	a4	4	14224	33	3	m	p
t3sf	a2	2	17018	33	3	s	f
t3lf	a2	2	32800	33	3	l	f
t1mp	a4	4	18667	33	1	m	p
t1lp	a1	1	11144	33	1	l	p
t2sp	a2	3	23759	33	2	s	p
t2mf	a1	1	36123	33	2	m	f
t4sp	a2	2	22233	33	4	s	p
t4lf	a1	1	19246	33	4	l	f
t4mf	a1	1	11924	33	4	m	f
t1sp	a3	3	5286	33	1	s	p
t1mf	a2	2	6449	33	1	m	f
t2lf	a3	3	14920	33	2	l	f
t3sp	a1	1	9951	33	3	s	p
t2mp	a1	1	12736	33	2	m	p
t3lp	a2	4	17707	33	3	l	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t4sf	a2	2	9840	33	4	s	f
t1sf	a3	3	4800	33	1	s	f
t4mp	a1	1	18072	33	4	m	p
t3mf	a3	3	16834	33	3	m	f
t1lf	a1	1	7271	33	1	l	f
t2sf	a1	1	12168	33	2	s	f
t4lp	a1	1	29049	33	4	l	p
t1lp	a1	1	7520	34	1	l	p
t3mf	a3	3	20903	34	3	m	f
t3lf	a2	2	10295	34	3	l	f
t2sf	a1	2	21099	34	2	s	f
t2lf	a3	3	44286	34	2	l	f
t1lf	a1	1	11268	34	1	l	f
t1mf	a2	2	6943	34	1	m	f
t4mf	a1	1	18888	34	4	m	f
t1sp	a3	3	5456	34	1	s	p
t4lf	a1	1	19288	34	4	l	f
t2mf	a1	2	34570	34	2	m	f
t3mp	a4	4	13231	34	3	m	p
t3sp	a1	1	13591	34	3	s	p
t1mp	a4	4	15219	34	1	m	p
t4sf	a2	2	10186	34	4	s	f
t2sp	a2	2	26046	34	2	s	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t1sf	a3	3	6287	34	1	s	f
t3lp	a2	2	23771	34	3	l	p
t2mp	a1	1	12864	34	2	m	p
t3sf	a2	2	13326	34	3	s	f
t4lp	a1	1	16323	34	4	l	p
t4mp	a1	1	9641	34	4	m	p
t2lp	a3	1	11943	34	2	l	p
t4sp	a2	2	8536	34	4	s	p
t4lf	a1	1	9803	35	4	l	f
t2lp	a3	3	20542	35	2	l	p
t3mp	a4	3	13369	35	3	m	p
t1mp	a4	4	9048	35	1	m	p
t2sp	a2	2	17499	35	2	s	p
t4mp	a1	1	12281	35	4	m	p
t4sf	a2	2	8551	35	4	s	f
t1sp	a3	3	10592	35	1	s	p
t3lf	a2	2	22127	35	3	l	f
t2mf	a1	1	11562	35	2	m	f
t3sf	a2	2	12552	35	3	s	f
t2sf	a1	1	14304	35	2	s	f
t1mf	a2	2	5192	35	1	m	f
t2lf	a3	2	13471	35	2	l	f
t1lp	a1	1	5321	35	1	l	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t1sf	a3	3	7384	35	1	s	f
t3lp	a2	2	26155	35	3	l	p
t4mf	a1	1	8071	35	4	m	f
t3mf	a3	3	10929	35	3	m	f
t2mp	a1	1	10103	35	2	m	p
t3sp	a1	1	10080	35	3	s	p
t4lp	a1	1	19384	35	4	l	p
t4sp	a2	2	7156	35	4	s	p
t1lf	a1	1	4880	35	1	l	f
t1sf	a3	3	8527	36	1	s	f
t3lf	a2	2	22032	36	3	l	f
t2lp	a3	1	28205	36	2	l	p
t3mf	a3	3	16344	36	3	m	f
t1sp	a3	3	12911	36	1	s	p
t3mp	a4	4	23327	36	3	m	p
t1mf	a2	2	11299	36	1	m	f
t4mp	a1	1	14391	36	4	m	p
t1lp	a1	1	14440	36	1	l	p
t4sp	a2	2	19040	36	4	s	p
t3sp	a1	1	12138	36	3	s	p
t2lf	a3	2	27249	36	2	l	f
t2sp	a2	3	15559	36	2	s	p
t4lp	a1	1	16842	36	4	l	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t1mp	a4	4	16752	36	1	m	p
t2mf	a1	1	24376	36	2	m	f
t1lf	a1	1	15426	36	1	l	f
t4sf	a2	2	9024	36	4	s	f
t3sf	a2	2	12552	36	3	s	f
t2sf	a1	1	12192	36	2	s	f
t3lp	a2	2	24567	36	3	l	p
t4mf	a1	1	10740	36	4	m	f
t4lf	a1	1	15104	36	4	l	f
t2mp	a1	1	16712	36	2	m	p
t2sf	a1	1	18497	37	2	s	f
t3sp	a1	1	13303	37	3	s	p
t1sf	a3	3	8694	37	1	s	f
t1mf	a2	2	13266	37	1	m	f
t3lf	a2	4	18282	37	3	l	f
t3mf	a3	3	17301	37	3	m	f
t4lp	a1	1	25431	37	4	l	p
t1lp	a1	1	7217	37	1	l	p
t3sf	a2	2	11636	37	3	s	f
t2mp	a1	1	15764	37	2	m	p
t4sf	a2	2	15884	37	4	s	f
t1mp	a4	4	7833	37	1	m	p
t1sp	a3	3	7552	37	1	s	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t4mf	a1	1	12800	37	4	m	f
t2sp	a2	2	12779	37	2	s	p
t2lp	a3	2	15898	37	2	l	p
t3lp	a2	2	20437	37	3	l	p
t4lf	a1	1	10549	37	4	l	f
t2mf	a1	1	9534	37	2	m	f
t3mp	a4	3	18765	37	3	m	p
t4mp	a1	1	8715	37	4	m	p
t4sp	a2	2	9253	37	4	s	p
t1lf	a1	1	7081	37	1	l	f
t2lf	a3	2	12098	37	2	l	f
t3sp	a1	1	12155	38	3	s	p
t2sp	a2	2	24928	38	2	s	p
t1lf	a1	1	5551	38	1	l	f
t4lf	a1	1	11992	38	4	l	f
t1sf	a3	3	9127	38	1	s	f
t2lf	a3	3	42181	38	2	l	f
t3mp	a4	4	24151	38	3	m	p
t2mp	a1	1	20216	38	2	m	p
t4sf	a2	2	9320	38	4	s	f
t1mf	a2	2	5080	38	1	m	f
t3lf	a2	2	24027	38	3	l	f
t4mp	a1	1	6865	38	4	m	p

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t1lp	a1	1	4567	38	1	l	p
t1sp	a3	3	6344	38	1	s	p
t3mf	a3	3	9224	38	3	m	f
t2sf	a1	1	15208	38	2	s	f
t2mf	a1	1	9895	38	2	m	f
t4lp	a1	1	19251	38	4	l	p
t4mf	a1	1	8344	38	4	m	f
t3sf	a2	2	12128	38	3	s	f
t3lp	a2	2	22832	38	3	l	p
t1mp	a4	4	8447	38	1	m	p
t4sp	a2	2	7401	38	4	s	p
t2lp	a3	3	18603	38	2	l	p
t1sf	a3	3	11400	39	1	s	f
t3mp	a4	4	18800	39	3	m	p
t3sp	a1	1	13768	39	3	s	p
t1mf	a2	2	13186	39	1	m	f
t1lp	a1	1	7913	39	1	l	p
t3lf	a2	2	44991	39	3	l	f
t2sf	a1	2	31210	39	2	s	f
t2mf	a1	1	32400	39	2	m	f
t2lp	a3	2	40898	39	2	l	p
t4sf	a2	2	9841	39	4	s	f
t4lf	a1	1	13794	39	4	l	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t3sf	a2	2	20175	39	3	s	f
t3lp	a2	2	51091	39	3	l	p
t1lf	a1	1	5696	39	1	l	f
t4mp	a1	1	9944	39	4	m	p
t2sp	a2	2	12584	39	2	s	p
t2lf	a3	3	38291	39	2	l	f
t4lp	a1	1	18744	39	4	l	p
t3mf	a3	3	25785	39	3	m	f
t1sp	a3	3	9792	39	1	s	p
t2mp	a1	1	24416	39	2	m	p
t4sp	a2	2	15704	39	4	s	p
t4mf	a1	1	11091	39	4	m	f
t1mp	a4	4	16424	39	1	m	p
t1sp	a3	3	8664	40	1	s	p
t4mp	a1	1	13810	40	4	m	p
t2sp	a2	2	34791	40	2	s	p
t1mp	a4	4	13958	40	1	m	p
t4lp	a1	1	27037	40	4	l	p
t4sp	a2	2	10577	40	4	s	p
t2mf	a1	1	16615	40	2	m	f
t3mf	a3	3	21264	40	3	m	f
t1lp	a1	1	10559	40	1	l	p
t1sf	a3	3	11084	40	1	s	f

Continued on next page

Table B.3 – continued from previous page

Question ID	Correct Answer	Answer Given	Time (ms)	User ID	Task	Size	Method
t3lp	a2	2	30960	40	3	l	p
t3sp	a1	1	18024	40	3	s	p
t2lf	a3	2	45851	40	2	l	f
t4lf	a1	1	14871	40	4	l	f
t2sf	a1	1	13561	40	2	s	f
t1mf	a2	2	14896	40	1	m	f
t4sf	a2	2	14232	40	4	s	f
t1lf	a1	1	9233	40	1	l	f
t3lf	a2	2	41513	40	3	l	f
t4mf	a1	1	12151	40	4	m	f
t2mp	a1	1	14683	40	2	m	p
t3sf	a2	2	16328	40	3	s	f
t2lp	a3	3	83332	40	2	l	p
t3mp	a4	4	15247	40	3	m	p

B.5.2 Preferences

Table B.4: Preference results

User ID	Number of Force Preferences	Number of Pattern Preferences
1	1	2
Continued on next page		

Table B.4 – continued from previous page

User ID	Number of Force Preferences	Number of Pattern Preferences
2	1	2
3	3	0
4	2	1
5	3	0
6	1	2
7	3	0
8	3	0
9	0	3
10	3	0
11	3	0
12	2	1
13	3	0
14	3	0
15	3	0
16	1	2
17	3	0
18	3	0
19	0	3
20	0	3
21	3	0
23	1	2
24	0	3
25	1	2

Continued on next page

Table B.4 – continued from previous page

User ID	Number of Force Preferences	Number of Pattern Preferences
27	3	0
26	3	0
28	2	1
29	0	3
30	2	1
22	2	1
31	3	0
32	2	1
33	2	1
34	2	1
35	2	1
36	3	0
37	1	2
38	2	1
39	0	3
40	2	1