ON THE CONTROL OF OPINION IN SOCIAL NETWORKS

by

Andrew Runka

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

School of Computer Science

at

CARLETON UNIVERSITY

Ottawa, Ontario
January, 2016

© Copyright by Andrew Runka, 2016
# Table of Contents

List of Tables vi

List of Algorithms vii

List of Figures viii

List of Symbols x

Abstract xiv

Chapter 1 Introduction 1

1.1 Thesis Objective 2

1.2 Problem Statement & Motivation 2

1.3 Summary of Contributions 5

1.4 List of publications 6

1.5 Thesis Organization 6

Chapter 2 Background 7

2.1 Introduction 7

2.2 Metaheuristics 7

2.2.1 Artificial Neural Networks 8

2.2.2 Evolutionary Algorithms 10

2.2.3 Evolutionary Neural Networks 12

2.3 Fitness Landscape Analysis 14

2.4 Network Characteristics 16

2.5 Random Network Generation 17

2.6 Summary 19
I Problem Definition

Chapter 3  Network Control & Related Problems

3.1 Introduction .................................................. 21

3.2 The Network Control Problem .................................. 21
  3.2.1 The Objective Function .................................. 22
  3.2.2 The Social Network ....................................... 23
  3.2.3 The Diffusion Model ....................................... 23
  3.2.4 The Control System Configuration ......................... 25
  3.2.5 The Control System Behaviour ............................. 27

3.3 Existing Network Control Problems ............................. 28
  3.3.1 Influence Maximization Problem ......................... 28
  3.3.2 Structural Controllability ................................. 32
  3.3.3 m-Effectors Problem ...................................... 34
  3.3.4 Immunization Problem ..................................... 35

3.4 Summary ...................................................... 37

Chapter 4  The $\theta$-Consensus Avoidance Problem

4.1 Introduction .................................................. 39

4.2 Motivation .................................................... 40
  4.2.1 Pole Balancing Problem .................................. 41

4.3 Problem Definition ........................................... 42
  4.3.1 Objective Function ....................................... 42
  4.3.2 Network Structure ......................................... 42
  4.3.3 Diffusion Model ........................................... 43
  4.3.4 Control System Configuration ............................ 44
  4.3.5 Control System Behaviour ................................ 45
  4.3.6 Discussion ................................................ 45

4.4 Summary ...................................................... 47
## II Experimental Evaluation

### Chapter 5 Behavioural Control

5.1 Introduction .................................................. 49
5.2 Design ..................................................... 50
   5.2.1 Null Controller (NC) ................................. 50
   5.2.2 Random Controller (RC) ......................... 50
   5.2.3 Locally Optimal Controller (LO) ............... 51
   5.2.4 Anti-Majority Controller (AM) ................. 51
   5.2.5 Artificial Neural Network (ANN) ............... 51
   5.2.6 Evolutionary Neural Network (ENN) .......... 53
   5.2.7 Evolutionary Connectionist Anti-Majority Controller (ECAM) 55
   5.2.8 Experimental parameters ....................... 57
5.3 Results ................................................... 59
   5.3.1 Analysis of Uncontrolled Diffusion .......... 59
   5.3.2 Comparative Analysis of Heuristics .......... 62
   5.3.3 Evolutionary Neurocontroller Analysis ....... 68
   5.3.4 Landscape Analysis ............................... 75
5.4 Summary .................................................. 82

### Chapter 6 Configuration Control

6.1 Introduction ................................................ 84
6.2 Motivation ................................................ 85
   6.2.1 Desirability Measures ............................ 85
6.3 Design ................................................... 88
   6.3.1 Configuration Heuristics ....................... 89
   6.3.2 Experimental Parameters ....................... 96
   6.3.3 Comparison Metrics ............................. 97
6.4 Results ................................................... 98
   6.4.1 Comparative Analysis ......................... 99
   6.4.2 Network Structure Dependency ............... 106
6.4.3 Optimal Property Verification .......................... 112
6.4.4 Behaviour-Configuration Dependency ..................... 114
6.5 Summary .................................................. 117

Chapter 7  Real Value Voter Model 119
7.1 Introduction ................................................. 119
7.2 Motivation .................................................. 120
7.3 Problem Definition ......................................... 120
  7.3.1 Objective Functions ................................. 120
  7.3.2 Network Structure .................................... 122
  7.3.3 Real-valued Voter Model .......................... 122
  7.3.4 Control System Configuration ....................... 123
  7.3.5 Control System Behaviour .......................... 124
  7.3.6 Discussion ............................................. 124
7.4 Experimental Results ...................................... 126
  7.4.1 Uncontrolled RVM Behaviour ......................... 126
  7.4.2 Real-valued $\theta$-Consensus Avoidance Problem .... 128
  7.4.3 $\theta$-Extremism Avoidance Problem .................. 131
7.5 Summary .................................................. 133

Chapter 8  Conclusion 135
8.1 Summary of Results ....................................... 135
8.2 Future Research .......................................... 139
  8.2.1 Additional breadth of intelligent algorithms for the $\theta$-CAP 139
  8.2.2 Greater depth of $\theta$-CAP analysis .................... 140
  8.2.3 Towards the general NCP ............................ 141

Appendix A  Ramped Evolution 143

Bibliography 148
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Uncontrolled run lengths</td>
<td>60</td>
</tr>
<tr>
<td>5.2</td>
<td>Location of surface ridge with $\theta_G=50$</td>
<td>62</td>
</tr>
<tr>
<td>5.3</td>
<td>Location of surface ridge with $\theta_G=30$</td>
<td>63</td>
</tr>
<tr>
<td>5.4</td>
<td>Controller heuristic run times</td>
<td>65</td>
</tr>
<tr>
<td>5.5</td>
<td>Comparison of Neurocontroller landscapes</td>
<td>79</td>
</tr>
<tr>
<td>6.1</td>
<td>Configuration heuristic ranking</td>
<td>101</td>
</tr>
<tr>
<td>6.2</td>
<td>Uncontrolled run lengths per network structure</td>
<td>107</td>
</tr>
<tr>
<td>6.3</td>
<td>Configuration heuristic comparison by network</td>
<td>109</td>
</tr>
<tr>
<td>6.4</td>
<td>Predictive accuracy of configuration properties</td>
<td>113</td>
</tr>
<tr>
<td>6.5</td>
<td>Run length comparison of behaviour-configuration combinations</td>
<td>115</td>
</tr>
<tr>
<td>6.6</td>
<td>Configuration heuristic ranking per behaviour</td>
<td>117</td>
</tr>
<tr>
<td>7.1</td>
<td>RVM step size comparison</td>
<td>127</td>
</tr>
<tr>
<td>7.2</td>
<td>Uncontrolled RVM diffusion properties</td>
<td>128</td>
</tr>
</tbody>
</table>
## List of Algorithms

1. Pseudo-code of the Evolutionary Algorithm metaheuristic .......................... 10
2. Pseudocode of the NCP simulation process ............................................. 25
3. Pseudo-code of the heuristic approach to a directed maximum matching, adapted from [16]. ................................................................. 87
4. Pseudo-code of the Farness configuration heuristic. ................................. 93
5. Pseudo-code of the cost-sensitive bit-flip mutation operator for the structural ECAM algorithm. ................................................................. 96
6. Pseudo-code of the Ramped evolution ENN for the $\theta$-CAP. ................. 144
List of Figures

<table>
<thead>
<tr>
<th>Figure Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Feed forward neural network</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Permutation problem example</td>
<td>13</td>
</tr>
<tr>
<td>3.1</td>
<td>NCP phases</td>
<td>24</td>
</tr>
<tr>
<td>3.2</td>
<td>Controller configuration types</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>Pole-balancing problem</td>
<td>41</td>
</tr>
<tr>
<td>4.2</td>
<td>$\theta_G$ Consensus Threshold</td>
<td>43</td>
</tr>
<tr>
<td>5.1</td>
<td>ENN controller architecture</td>
<td>54</td>
</tr>
<tr>
<td>5.2</td>
<td>CAM controller architecture</td>
<td>55</td>
</tr>
<tr>
<td>5.3</td>
<td>ENN controller architecture</td>
<td>56</td>
</tr>
<tr>
<td>5.4</td>
<td>Heuristic parameter-space surfaces</td>
<td>60</td>
</tr>
<tr>
<td>5.5</td>
<td>Sampling distribution of fitnesses</td>
<td>61</td>
</tr>
<tr>
<td>5.6</td>
<td>ANN learning curve</td>
<td>66</td>
</tr>
<tr>
<td>5.7</td>
<td>ENN learning curve</td>
<td>67</td>
</tr>
<tr>
<td>5.8</td>
<td>ENN surfaces</td>
<td>69</td>
</tr>
<tr>
<td>5.9</td>
<td>ECAM vs. ENN coverage</td>
<td>70</td>
</tr>
<tr>
<td>5.10</td>
<td>ECAM vs. ENN fitness differences</td>
<td>71</td>
</tr>
<tr>
<td>5.11</td>
<td>ECAM vs. ENN significance</td>
<td>72</td>
</tr>
<tr>
<td>5.12</td>
<td>ECAM vs. ENN Behavioural comparison</td>
<td>74</td>
</tr>
<tr>
<td>5.13</td>
<td>ECAM vs. ENN execution time</td>
<td>76</td>
</tr>
<tr>
<td>5.14</td>
<td>Characteristic landscape curves</td>
<td>77</td>
</tr>
<tr>
<td>5.15</td>
<td>Multiple evaluation landscape walks</td>
<td>80</td>
</tr>
<tr>
<td>5.16</td>
<td>Landscape differences using multiple evaluations</td>
<td>80</td>
</tr>
<tr>
<td>5.17</td>
<td>ECAM vs. ENN multiple evaluation landscapes</td>
<td>81</td>
</tr>
<tr>
<td>5.18</td>
<td>$\theta$-CAP behavioural algorithm decision tree</td>
<td>82</td>
</tr>
<tr>
<td>6.1</td>
<td>Configuration heuristic examples</td>
<td>89</td>
</tr>
<tr>
<td>6.2</td>
<td>Structural ECAM controller architecture</td>
<td>94</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>6.3</td>
<td>Configuration heuristic performance comparison</td>
<td>100</td>
</tr>
<tr>
<td>6.4</td>
<td>Configuration heuristic similarity (small budget)</td>
<td>103</td>
</tr>
<tr>
<td>6.5</td>
<td>Configuration heuristic similarity (large budget)</td>
<td>104</td>
</tr>
<tr>
<td>6.6</td>
<td>Configuration centrality and clustering</td>
<td>105</td>
</tr>
<tr>
<td>6.7</td>
<td>Graph generator properties</td>
<td>106</td>
</tr>
<tr>
<td>6.8</td>
<td>Configuration heuristic performance over all graphs</td>
<td>108</td>
</tr>
<tr>
<td>6.9</td>
<td>Proportional rank sum by graph</td>
<td>110</td>
</tr>
<tr>
<td>6.10</td>
<td>Proportional rank sum by instance parameters</td>
<td>111</td>
</tr>
<tr>
<td>6.11</td>
<td>Configuration rank sum by behaviour</td>
<td>116</td>
</tr>
<tr>
<td>7.1</td>
<td>RVM transmission process</td>
<td>123</td>
</tr>
<tr>
<td>7.2</td>
<td>$\theta$-CAP$_{RV}$: ECAM vs. Random</td>
<td>130</td>
</tr>
<tr>
<td>7.3</td>
<td>$\theta$-CAP$_{RV}$: Farness vs. Random</td>
<td>130</td>
</tr>
<tr>
<td>7.4</td>
<td>$\theta$-CAP$_{RV}$ vs. $\theta$-EAP</td>
<td>131</td>
</tr>
<tr>
<td>7.5</td>
<td>$\theta$-EAP: ECAM vs. Random</td>
<td>132</td>
</tr>
<tr>
<td>7.6</td>
<td>$\theta$-EAP: ECAM vs. Zero</td>
<td>133</td>
</tr>
</tbody>
</table>
### List of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>Significance threshold</td>
<td>59</td>
</tr>
<tr>
<td>β</td>
<td>Watts-Strogatz algorithm randomness parameter</td>
<td>17</td>
</tr>
<tr>
<td>ϵ</td>
<td>Landscape analysis sensitivity parameter</td>
<td>14</td>
</tr>
<tr>
<td>η</td>
<td>Global neural network learning rate</td>
<td>8</td>
</tr>
<tr>
<td>λ</td>
<td>Exponential distribution rate parameter</td>
<td>59</td>
</tr>
<tr>
<td>Ψ</td>
<td>Random walk transition ensemble</td>
<td>14</td>
</tr>
<tr>
<td>θ-CAP</td>
<td>θ-Consensus Avoidance Problem</td>
<td>1</td>
</tr>
<tr>
<td>θ-CAP_{RV}</td>
<td>Real-valued θ-Consensus Avoidance Problem</td>
<td>121</td>
</tr>
<tr>
<td>θ-EAP</td>
<td>θ-Extremism Avoidance Problem</td>
<td>121</td>
</tr>
<tr>
<td>θ_C</td>
<td>Local consensus threshold</td>
<td>46</td>
</tr>
<tr>
<td>θ_G</td>
<td>Consensus threshold parameter</td>
<td>42</td>
</tr>
<tr>
<td>σ(c, v, t)</td>
<td>Control signal from controller c to node v at time t</td>
<td>25</td>
</tr>
<tr>
<td>D</td>
<td>Domain of state values</td>
<td>23</td>
</tr>
<tr>
<td>acq.</td>
<td>Acquaintance configuration heuristic</td>
<td>91</td>
</tr>
<tr>
<td>avg_icpl</td>
<td>Average intra-configuration path length</td>
<td>97</td>
</tr>
<tr>
<td>avg_decay</td>
<td>Average decay centrality</td>
<td>97</td>
</tr>
<tr>
<td>A(v)</td>
<td>Adjustment strategy of node v</td>
<td>23</td>
</tr>
<tr>
<td>AM</td>
<td>Anti-Majority controller</td>
<td>51</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
<td>8</td>
</tr>
<tr>
<td>broke.</td>
<td>Brokerage configuration heuristic</td>
<td>91</td>
</tr>
<tr>
<td>B</td>
<td>The maximum configuration budget</td>
<td>27</td>
</tr>
<tr>
<td>B^*</td>
<td>Diffusionless threshold budget</td>
<td>62</td>
</tr>
<tr>
<td>c</td>
<td>A individual controller in C</td>
<td>25</td>
</tr>
<tr>
<td>cheap.</td>
<td>Cheapest configuration heuristic</td>
<td>88</td>
</tr>
<tr>
<td>C</td>
<td>The control System</td>
<td>21</td>
</tr>
<tr>
<td>CAM</td>
<td>Connectionist Anti-Majority controller</td>
<td>55</td>
</tr>
<tr>
<td>Con_v</td>
<td>Network constraint measure</td>
<td>87</td>
</tr>
<tr>
<td>$\text{cost}(v)$</td>
<td>The cost of adding node $v$ to $V_C$</td>
<td>27</td>
</tr>
<tr>
<td>$d$</td>
<td>Degree parameter of a network generator algorithm</td>
<td>17</td>
</tr>
<tr>
<td>$d(v)$</td>
<td>Undirected degree of node $v$</td>
<td>28</td>
</tr>
<tr>
<td>$\text{deg.}$</td>
<td>Degree configuration heuristic</td>
<td>90</td>
</tr>
<tr>
<td>$\text{diff}(A, B)$</td>
<td>Rectilinear distance between sets $A$ and $B$</td>
<td>97</td>
</tr>
<tr>
<td>DBI</td>
<td>Density Basin Information</td>
<td>14</td>
</tr>
<tr>
<td>DM</td>
<td>Diffusion Model</td>
<td>21</td>
</tr>
<tr>
<td>$e_{u,v}$</td>
<td>Edge from node $u$ to node $v$</td>
<td>23</td>
</tr>
<tr>
<td>evo.</td>
<td>Evolved configuration heuristic</td>
<td>93</td>
</tr>
<tr>
<td>$E$</td>
<td>Edges in the social network</td>
<td>17</td>
</tr>
<tr>
<td>EA</td>
<td>Evolutionary Algorithm</td>
<td>10</td>
</tr>
<tr>
<td>ECAM</td>
<td>Evolutionary Connectionist Anti-Majority controller</td>
<td>55</td>
</tr>
<tr>
<td>ENN</td>
<td>Evolutionary Neural Network</td>
<td>12</td>
</tr>
<tr>
<td>ER</td>
<td>Erdos-Renyi graph generator</td>
<td>17</td>
</tr>
<tr>
<td>$f(C, V_C)$</td>
<td>Fitness of an applied control system</td>
<td>21</td>
</tr>
<tr>
<td>far.</td>
<td>Farness configuration heuristic</td>
<td>92</td>
</tr>
<tr>
<td>$G$</td>
<td>Social network structure</td>
<td>17</td>
</tr>
<tr>
<td>$I(v, t)$</td>
<td>Information received at node $v$ at time $t$</td>
<td>23</td>
</tr>
<tr>
<td>$Id$</td>
<td>Identity vector for $V_C$</td>
<td>97</td>
</tr>
<tr>
<td>IC</td>
<td>Information Content</td>
<td>14</td>
</tr>
<tr>
<td>IMP</td>
<td>Influence Maximization Problem</td>
<td>28</td>
</tr>
<tr>
<td>IP</td>
<td>Immunization Problem</td>
<td>35</td>
</tr>
<tr>
<td>$L(v, I)$</td>
<td>Learning strategy of node $v$ with information $I$</td>
<td>23</td>
</tr>
<tr>
<td>LO</td>
<td>Locally Optimal controller</td>
<td>51</td>
</tr>
<tr>
<td>$m$</td>
<td>Upper bound on the size of $V_C$</td>
<td>28</td>
</tr>
<tr>
<td>match.</td>
<td>Matching configuration heuristic</td>
<td>92</td>
</tr>
<tr>
<td>$N(v, t)$</td>
<td>Neighbourhood of node $v$ at time $t$</td>
<td>23</td>
</tr>
<tr>
<td>$N^k(v, t)$</td>
<td>Neighbourhood of node $v$ at time $t$ with state $k$</td>
<td>35</td>
</tr>
<tr>
<td>NC</td>
<td>Null Controller</td>
<td>50</td>
</tr>
<tr>
<td>NCP</td>
<td>Network Control Problem</td>
<td>21</td>
</tr>
<tr>
<td>Symbol</td>
<td>Definition</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
<td>------</td>
</tr>
<tr>
<td>NN</td>
<td>Nearest Neighbour graph generator</td>
<td>17</td>
</tr>
<tr>
<td>p</td>
<td>Significance testing p-value</td>
<td>99</td>
</tr>
<tr>
<td>P</td>
<td>The set of proxy nodes in G</td>
<td>25</td>
</tr>
<tr>
<td>Pr</td>
<td>Observed probability</td>
<td>14</td>
</tr>
<tr>
<td>PA</td>
<td>Preferential Attachment graph generator</td>
<td>17</td>
</tr>
<tr>
<td>PIC</td>
<td>Partial Information Content</td>
<td>14</td>
</tr>
<tr>
<td>R(V_C)</td>
<td>Expected return from selecting the configuration V_C</td>
<td>28</td>
</tr>
<tr>
<td>rand.</td>
<td>Random configuration heuristic</td>
<td>89</td>
</tr>
<tr>
<td>RC</td>
<td>Random Controller</td>
<td>50</td>
</tr>
<tr>
<td>RVM</td>
<td>Real valued Voter Model diffusion model</td>
<td>122</td>
</tr>
<tr>
<td>s(v, t)</td>
<td>The state value of node v at time t</td>
<td>23</td>
</tr>
<tr>
<td>s(V, t)</td>
<td>The state of the entire network at time t</td>
<td>23</td>
</tr>
<tr>
<td>S(C, t)</td>
<td>The visible state of the network to C at time t</td>
<td>25</td>
</tr>
<tr>
<td>Sh(v)</td>
<td>Sharing strategy of node v</td>
<td>23</td>
</tr>
<tr>
<td>step</td>
<td>RVM step size parameter</td>
<td>122</td>
</tr>
<tr>
<td>SIR</td>
<td>Susceptible-Infected-Removed diffusion model</td>
<td>35</td>
</tr>
<tr>
<td>SIS</td>
<td>Susceptible-Infected-Susceptible diffusion model</td>
<td>35</td>
</tr>
<tr>
<td>SW</td>
<td>Small World graph generator</td>
<td>17</td>
</tr>
<tr>
<td>t</td>
<td>A timestep</td>
<td>23</td>
</tr>
<tr>
<td>temp(V_C)</td>
<td>Temporal advantage of set V_C</td>
<td>112</td>
</tr>
<tr>
<td>tSize</td>
<td>Tournament selection size</td>
<td>10</td>
</tr>
<tr>
<td>U(t)</td>
<td>Utility function</td>
<td>42</td>
</tr>
<tr>
<td>Uni[a, b]</td>
<td>Uniform random number between a and b</td>
<td>50</td>
</tr>
<tr>
<td>v</td>
<td>A vertex in V</td>
<td>23</td>
</tr>
<tr>
<td>V</td>
<td>Vertices in the social network</td>
<td>17</td>
</tr>
<tr>
<td>V_k(t)</td>
<td>The set of nodes with state k at time t</td>
<td>42</td>
</tr>
<tr>
<td>V_C</td>
<td>The set of controlled nodes</td>
<td>21</td>
</tr>
<tr>
<td>VM</td>
<td>Voter Model diffusion model</td>
<td>28</td>
</tr>
<tr>
<td>w(e_{u,v}, t)</td>
<td>Weight on the given edge</td>
<td>23</td>
</tr>
<tr>
<td>Zero</td>
<td>Zero controller</td>
<td>124</td>
</tr>
</tbody>
</table>
“Everything we hear is an opinion, not a fact. Everything we see is a perspective, not the truth.”

Marcus Aurelius, Meditations
Abstract

Social networks are an ubiquitous element of our daily experience. A question that naturally arises in the operation of such networks is whether they can be controlled. With domains ranging from financial markets to extremism avoidance, the study of control of opinion in social networks is extremely relevant in modern social media culture.

Automated control of the flow of information in large-scale non-deterministic social networks is a complex problem requiring both a search for the optimal configuration of connections to the network, and a behaviour that determines the required control signals. This thesis formalizes the Network Control Problem (NCP) as a means of describing the field of diverse social network control problems. To date, problems that can be described as NCP examples deal primarily with the configuration rather than the behavioural component.

The $\theta$-Consensus Avoidance Problem ($\theta$-CAP) is defined in this thesis as a novel NCP which has the objective of avoiding consensus in a social network of agents. It is an important problem representing the avoidance of extreme views that may lead to extreme behaviour, such as bubble or panic events in a socially-connected market. The $\theta$-CAP is intended as a foundational benchmark problem in the NCP domain. Experimentation is developed to demonstrate the utility of the $\theta$-CAP as a practical benchmark problem with scalable difficulty parameters.

A number of heuristic and metaheuristic implementations of both configuration and behaviour are compared in this thesis. Special consideration is given to the application of evolutionary neurocontrollers toward learning to optimize the behaviour component. Analysis indicates a variety of conditions that affect the quality of given approaches, including network structure, instance difficulty parameters, and variations among control approaches. Consistent, observable trends were determined in the solution space that were verified across behaviours, configurations, network structures, and objectives.
Acknowledgements

“No act of kindness, however small, is ever wasted.”

Aesop

- To Dr. Mario Ventresca for plucking me out of the back row and setting me on the path.
- To Dr. Betty Ombuki-Berman for fostering my abilities with determination to see me succeed.
- To Prof. Tony White for your patient guidance, substantive discourse, and insightful revisions.
- To friends and family for support, shenanigans, refreshment, and sanity.
- To those who were lost for giving us an unending motivation.

To all the teachers of various forms who have collectively added fuel and catalyst to my spark: without you all, this fire would have sputtered out long ago. I am deeply grateful to the village of helping hands that held me up and brought me to this point.
Chapter 1

Introduction

“Opinion has caused more trouble on this little earth than plagues or earthquakes.”

Voltaire

The behaviour of individuals is strongly influenced by their peers. Simply stated, “people frequently do what they do because of what they think (relevant) others do” [108]. In fact, in one experiment it was found that social influence was such an overwhelming motivation that it often outweighed the direct evidence of an individual’s own senses [3, 119]. Opinion, and by extension the ability to influence the opinions of others, has tangible consequences via the collective behaviour of those individuals. Networked interactions of individual agents coalesce to form anything from biological entities (e.g., multi-cellular organisms, flocks, or swarms) to complex social structures (e.g., gangs, political parties, or grass-roots organizations). With the rapid expansion of communication technology and the resulting interconnection of the global community, social organization is shifting from a vertical chain of authority towards horizontal peer pressure [14]. The effect of this shift, relative to the formation of individual opinions and beliefs, has advertisers and consulting firms scrambling to adapt to the new media of social propaganda [124]. An effective application of social influence (e.g., propaganda or advertising) has the potential to deliberately shape behaviours of entire networked populations. This imposition of external influence on the opinions of agents within social networks is an important problem with wide interest, implication, and varied application.

A social network is a formal abstraction that describes the interactions of a set of agents. The state of an agent (e.g., its opinion, infection status, wealth, etc.) is determined largely through interactions with neighbouring agents in the network, who
were, in turn, affected through prior interactions. The state of all individual agents, each determined personally, collectively define the network state. The network state changes as opinions flow from agent to agent via social interaction. This flow of opinions, or *information diffusion*, defines the dynamics of the time varying network state. Deliberate manipulation of this information diffusion via *social influence* is, therefore, a means of coordinating the network state via interactions at the agent state level.

Social influence has been formally described in a number of ways. In this thesis, the term influence (v.) is used to refer to the social marketing view: Influence is the deliberate and systematic act of providing crafted messages specifically designed to alter the opinions of individuals [49]. Without ambiguity, the term influence (n.) may also be used to refer to the property of an individual or group, denoting their utility as targets for such an act (relative to some objective) [25, 38, 57].

The widespread use of social media is levelling the information exchange hierarchy, allowing access to social propaganda for wider varieties of information and sources [11]. New steps must now be taken to coordinate this distribution of information such that it can be harnessed towards the deliberate benefit of society.

### 1.1 Thesis Objective

The primary objective of this thesis is to determine whether the state of social networks can be controlled; specifically, if such control is possible via social influence. Beyond establishing this goal, this thesis aims to investigate the conditions under which such control can be effectively achieved.

### 1.2 Problem Statement & Motivation

To frame the objectives as a problem, this thesis considers the question: Can targeted external influence prevent a social network from catastrophe? The development of a control system that successfully achieves this goal can be considered a proof by construction of the conditional ability to control a social network using social influence. The outline of steps this thesis takes to this end are as follows:
- A formal description of the general problem was put forth to unify the literature review of existing social network control problems
- A gap was identified in the literature relating to the behavioural subproblem
- A novel optimization problem was defined to address this gap
- Various learning and heuristic controllers were applied to this new problem to investigate the conditions under which the control system is effective

These steps are motivated and explained in more detail below.

The concept of social network control is formalized in the Network Control Problem (NCP). The NCP formalization (see Chapter 3, [102]) defines a pair of dependent optimization subproblems in terms of discovering both the structural and behavioural components of a control system. The objective of the control system is to guide the state of the social network via interactions at the node-state level. The amount of direct influence a control system has on the social network is constrained by a budget placed on the size of the set of controlled nodes. Thus the control system is required to determine both which nodes to control, and the series of control signals to be applied to them, in order to optimize the state of the social network.

The NCP is a general description of a class of specific problem instances. In all NCP instances the objective remains to control the state of a network; however, the specific goal and evaluation of states varies with each instance. As examples, one may seek to maximize the market adoption of a product [25, 57], or minimize the spread of contagion [19, 90]. The goal states change between each problem, and the control systems differ accordingly. Both maximizing and minimizing spread of information throughout the network correspond to selecting the best nodes to influence. Similarly, in [75], the problem of structural controllability is examined as a search for the optimal set of control sites. Each of the existing problems in the influence diffusion literature offers insight to the optimization of the configuration component of the NCP, but not the behavioural component. To address this, this thesis proposes a novel benchmark NCP instance, the $\theta$-Consensus Avoidance Problem ($\theta$-CAP).

Consensus in a social network is a terminal state relative to the diffusion of new information. Such a lack of diversity can leave a population susceptible to severe consequences. As examples, a consensus of opinion in a financial market can lead to
a bubble or a market run [41, 78], a lack of diversity in genetic expression can leave a population susceptible to disease [44], a lack of choice in a product market can lead to monopolistic prices [54], or a consensus of opinion regarding the importance of a particular policy can undermine public input of broader government policies [112]. In such cases, an automated control system may observe and inform human oversight, or where appropriate, take action with more speed and precision than human counterparts. So, understanding the necessary and/or sufficient conditions for an automated control system to guide a network of agents to avoid consensus is not only valuable in its application to broader network control (i.e., NCP), but has intrinsic importance as well.

The θ-CAP is defined as an NCP instance with both configuration and behavioural search requirements (see Chapter 4 [102]). In addition to considering which nodes to direct, the control system must also determine the step by step instructions it will give. The goal of the θ-CAP is to maintain the network state within a bounded range of equilibrium. More specifically, to determine a control system capable of achieving such a goal. Successful intelligent controlled avoidance behaviour, as such, is a demonstration of the potential for generalized social network control.

Exploration of the θ-CAP problem space is undertaken using a comparative analysis of a number of heuristic and metaheuristic approaches for each component subproblem (see Chapters 5 and 6). The estimation, learning, and evolutionary algorithms that are applied are drawn from related NCP and control systems literature where appropriate. To this author’s knowledge this is the first application of a learning system to the behavioural component of an NCP. The learning and evolutionary systems considered in this work are particularly interesting for their ability to find unsupervised solutions that may have been overlooked or otherwise unexpected.

The observed results of the control systems are considered over a variety of network structures, difficulty instances, and behaviour-configuration combinations to evaluate their consistency. Further exploration of landscape analysis and related problems are applied to determine the robustness of observed trends.
1.3 Summary of Contributions

The contributions of this work include:

- Formalization of the Network Control Problem (Chapter 3)
- Definition of the $\theta$-Consensus Avoidance Problem (Chapter 4)
- Comparative analysis of heuristic and metaheuristic solutions to the behavioural subproblem of the $\theta$-CAP (Chapter 5)
  - Definition of the Evolutionary Connectionist Anti-Majority algorithm variant (Section 5.2)
  - Performance analysis of Evolutionary Neurocontrollers used for the $\theta$-CAP (Section 5.3.3)
  - Landscape analysis of the $\theta$-CAP search space (Section 5.3.4)
- Comparative analysis of heuristic and metaheuristic solutions to the configuration subproblem of the $\theta$-CAP (Chapter 6)
  - Consideration of dependency of solutions on network structures (Section 6.4.2) and behaviour-configuration combinations (Section 6.4.4)
- Establishment of benchmark results for the $\theta$-CAP (Sections 5.3 and 6.4)
- Definition of the Real-valued Voter Model, the Real-valued $\theta$-CAP, and the $\theta$-Extremism Avoidance Problem (Chapter 7)

Each of these contributions is motivated by the foregoing discussion. The formalization of the NCP speaks to the desire for generalized social network control. This problem, however, is too broad to approach given the current state of literature. To this end, the $\theta$-CAP is defined as a simple problem to mitigate a single form of disaster in social networks. The study of the $\theta$-CAP problem space centres around three objectives: 1) demonstrating that the problem has merit as a benchmark (challenging/optimizable), 2) discovering the consistent traits of good solutions, and 3) establishing the problem and developed solutions as applicable to the broader NCP domain.
1.4 List of publications

The publications arising from this dissertation to date include:


1.5 Thesis Organization

The remainder of this thesis is organized as follows. The next chapter, Chapter 2, presents descriptions of related background material.

Following this, the thesis is divided into two main parts. First, the definitions of the problems introduced in this work are presented in Chapters 3 and 4, corresponding to the general NCP and the specific $\theta$-CAP definitions respectively.

The second part of the thesis considers the experimental results and analysis pertaining to the $\theta$-CAP. The study begins with a comparative analysis of solutions for the behavioural subproblem in Chapter 5. Following this, the results are extended to include the dependent optimization of the configuration subproblem in Chapter 6. Finally, the applicability of the developed solutions are tested against two real-valued problem variants in Chapter 7. The thesis concludes with a summary of results and a discussion of future research intent in Chapter 8.
Chapter 2

Background

“Most people are other people.
Their thoughts are someone else’s
opinions, their lives a mimicry, their
passions a quotation.”

Oscar Wilde, De Profundis

2.1 Introduction

This chapter presents introductory material on several background topics used throughout the remainder of the thesis. Readers who are familiar with any of the topics can skip the respective section without any loss of information regarding the contributions of this work itself. Section 2.2 describes the metaheuristics used; specifically, Feed Forward Artificial Neural Networks (Section 2.2.1), Evolutionary Algorithms (Section 2.2.2), and Evolutionary Neural Networks (Section 2.2.3). Section 2.3 explains information theoretic fitness landscape analysis. Section 2.4 explains the complex network analysis measures used. Finally, Section 2.5 defines a number of random network classes used in this thesis and how they are generated.

2.2 Metaheuristics

Metaheuristics are stochastic general purpose search algorithms that can be used to intelligently sample complex search spaces in order to obtain approximations of the optimal solution within reasonable time [77]. This section outlines the background details of three such approaches used in this thesis.
2.2.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are a universal function approximation meta-heuristic modelled after the functioning of the brain [43]. This thesis considers the use of feed forward ANNs exclusively. For a complete description of ANNs see, for example, [28, 83, 84].

A feed forward ANN (see Figure 2.1) consists of an input-to-output mapping represented as a layered structure of nodes, called neurons, connected by directed and weighted edges, called axons. A neural network is used by setting values at the input layer of neurons, which are then fed forwards typically through a single hidden layer, then to the output layer. The output of the final layer is considered the output of the mapping. The number of hidden layers as well as the number of neurons in each layer are experimental parameters. Increases in the size of either lead to increased expressiveness of the resulting ANN, as well as a corresponding increase in the complexity of training.

The mapping transforms information through a process of combination and linear separation. That is, all values entering a given neuron are combined using a weighted sum, where each individual value corresponds to the output of a neuron in the previous layer times the weight on the axon which it travels. Next, a threshold activation function is applied to the sum to generate the output of each neuron, that is then fed forward to the next layer. Two common threshold functions used in this thesis are the Logistic sigmoid (output range [0,1]), and Tangent sigmoid (output range [-1,1])
activation functions:

\[
\text{logSig}(x) = \frac{1}{1 + e^{-x}} \quad (2.1)
\]

\[
\text{tanSig}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.2)
\]

where \( x \) here refers to the weighted summed inputs to the neuron, and \( e \) refers to Euler’s constant.

Training a neural network consists of adjusting the weights of each axon, typically in a supervised context, to reduce the observed error from an expected output. A common approach for this adjustment is the use of gradient descent techniques, most notably backpropagation (BP) [129]. Standard BP propagates the error from the output nodes backwards through the network, to determine the size and direction of adjustment on every axon. This is done by propagating the observed error backwards through the network, and subsequently updating each axon weight using the following equation:

\[
\Delta w(a_{u,v}, t) = \eta \cdot out(u, t) \cdot -err(u, v, t) \quad (2.3)
\]

where \( \eta \) is the global learning rate across the network, \( err(u, v, t) = \frac{\delta \text{Error}(t)}{\delta w(a_{u,v}, t)} \) is the derivative of the (propagated) error with respect to the given weight at time \( t \), and \( out(u, t) \) is the output of neuron \( u \) at time \( t \). This method is dependent on the existence of a differentiable activation function as well as an expected output to determine the error at the output layer.

An alternative to the BP algorithm is the RPROP algorithm of Reидmiller and Braun [99], which uses only the sign of the error derivative to determine weight adjustments. Individual adjustment rates per axon in the network are maintained in order to increase the convergence speed and accuracy over traditional backpropagation. Using this method, the weights on an axon between neurons \( u \) and \( v \) will be updated as follows:

\[
\Delta w(a_{u,v}, t) = \begin{cases} 
-\Delta_{u,v}(t), & \text{if } err(u, v, t) > 0 \\
+\Delta_{u,v}(t), & \text{if } err(u, v, t) < 0 \\
0, & \text{otherwise}
\end{cases} \quad (2.4)
\]
Algorithm 1 Pseudocode of EA metaheuristic with a single individual elitism.

**Require:** $Pop$ is the initial population of solutions

1: procedure $TrainEA(instance, maxGen, Pop)$
2:  \hspace{1em} $gen \leftarrow 1$
3:  \hspace{1em} while not Termination($gen, maxGen, Pop$) do
4:  \hspace{1.5em} Evaluation( Decode($Pop$), $instance$ )
5:  \hspace{1.5em} $best \leftarrow \text{maxFitness}(Pop)$
6:  \hspace{1.5em} $Pop \leftarrow \text{Selection}(Pop, best)$
7:  \hspace{1.5em} $Pop \leftarrow \text{Variation}(Pop)$
8:  \hspace{1em} $gen \leftarrow gen + 1$
9: return $best$

where $\Delta_{u,v}(t)$ is the adjustment amount, defined as follows:

$$
\Delta_{u,v}(t) = \begin{cases} 
\eta^+ \Delta_{uv}(t - 1), & \text{if } err(u, v, t) \cdot err(u, v, t - 1) > 0 \\
\eta^- \Delta_{uv}(t - 1), & \text{if } err(u, v, t) \cdot err(u, v, t - 1) < 0 \\
\Delta_{uv}(t - 1), & \text{otherwise}
\end{cases}
$$

(2.5)

where $0 < \eta^- < 1 < \eta^+$ are parameters controlling changes in adjustment size.

**2.2.2 Evolutionary Algorithms**

Evolutionary Algorithms (EAs) are an unsupervised, population-based, stochastic search metaheuristic inspired by the concepts of biological evolution and survival of the fittest. The pseudocode description of an EA is presented in Figure 1. For a full description of EAs see, for example, [77, 103].

In brief, a population of potential solutions is initialized randomly and each is evaluated according to some fitness function. Individuals are selected stochastically for the next generation based on their relative fitness among the current population, and are subsequently combined and/or mutated to form the next generation. This process repeats until a solution of sufficient quality has been found or until the population ceases to improve over time.

A specific implementation of an Evolutionary Algorithm can be described via the following components. Each are explained here in the general case.
Population:

An evolutionary search is primarily concerned with the initialization, maintenance, and improvement of a population of potential solutions over a span of generations. The size of the population is considered an experimental parameter, with increased size typically improving coverage of the search space but requiring additional evaluations per generation. The representation of each solution is problem dependent and may be encoded for improved applicability of the search operators. This encoded form of individuals, known as the *genotype*, is used throughout the evolutionary algorithm, but must be decoded into its *phenotype* form prior to applying the solution to the problem at hand.

Evaluation:

In order to determine the worth of solutions in the population, the phenotypic representation of each is applied to the problem at hand. A problem dependent fitness function is used to determine a ranking of solutions. The best solution(s) in the population are kept aside as the *elitism* individual(s). These are placed in the next generation to ensure the best-so-far solution(s) are always present in the population.

Selection:

The highest ranked individuals in the population are stochastically selected to create the next generation of the population. The selection operator can be implemented in a variety of ways, though this thesis considers the *tournament* selection operator exclusively. There, a set of $tSize$ individuals is selected uniformly at random from the population, and the single best of those considered is added to the next generation. This process is repeated until the new generation’s population is full. The tournament parameter $tSize$ controls the selection pressure, with larger values of $tSize$ creating a selection with greater bias towards the best solutions while a $tSize = 1$ is equivalent to a random population search.
Variation:

The newly selected population undergoes the genetic variation operators, most commonly crossover and mutation. Crossover involves combining the solutions of two individuals in the population to produce new offspring. Mutation creates random variation in a single individual. The specifics of the variation operators are dependent on the genotypic representation.

Termination:

The evolutionary search proceeds in a process of repeated variation, evaluation, and selection until a termination condition is met. Commonly, this takes the form of a parameter-specified number of generations, a stagnation condition among the population, or a predefined level of accuracy in the best solution. Once the termination condition is met, the elitism individual is returned as the approximately optimal result of the search.

2.2.3 Evolutionary Neural Networks

As an alternative to the supervised ANN training techniques described in Section 2.2.1, this section considers Evolutionary Neural Networks (ENN) [85, 133, 134]. In comparison to standard BP, EA is claimed to be a more robust search operator with improved scalability and decreased likelihood to be trapped in local optima [111, 133]. Experimental comparisons on various function approximation and classification test problems demonstrate consistently better performance in terms of both speed and accuracy using EA over BP [85, 110, 111, 134]. Difficulty was noted in problems of fine-tuning very small numbers of weights, which is a task better suited to gradient descent techniques.

In an ENN system, the heuristic search method of EA is used to optimize the weights of a neural network. This allows the optimization of the ANN parameters without requiring knowledge of the error gradient or supervisory information. Additional variants of this approach include evolving the ANN connection structure and evolving parameters for the BP learning rule [24, 130, 133, 134]. This thesis will limit consideration to the case of evolving axon weight values. The EA framework
remains the same as previously described except for the representation of solutions and corresponding changes to the variation operators.

**Population:** The population in a typical ENN algorithm consists of a number of initially random real-valued weight matrices as the genotype. The dimensions of the matrices are fixed to a single value for every axon in the phenotypic neural network.

An issue known as the permutation problem (a.k.a., the competing conventions problem) [59, 85, 116] was identified as a challenge to neuroevolution when using an encoding of real value strings. Specifically, the change in the location of a neuron in any hidden layer of a fully connected neural network does not alter the its functionality; so, multiple optima with different encodings but equivalent performance may compete within the population. An example instance of the permutation problem is presented in Figure 2.2. There, the change in location between hidden neurons ‘a’ and ‘b’ results in functionally equivalent networks with distinctly different encodings. Alternative advanced ENN techniques such as ESP [33], CoSyNE [32], or Neat [116], have been presented to counteract this problem however, in the interest of focussing on the difficulty of the problem space itself, use of these are left for future research. It was suggested that the use of mutation as the primary genetic operator could reduce the effects of the permutation problem [59, 134]; this is the approach taken throughout this thesis.

**Variation:** The real-valued encoding of the genotype allows standard real-valued genetic operators to be used; however, Montana and Davis [85] present several crossover and mutation operators that take ANN features into account.
2.3 Fitness Landscape Analysis

The fitness space of a problem is the set of evaluations of all feasible solutions, ordered according to some neighbourhood function. Fitness landscape analysis considers samples of this space ordered according to the EA representation of solutions. Random walks are sampled through the fitness space using repeated applications of a search operator (e.g., mutation) to obtain neighbouring solutions. This process is repeated until a walk of sufficient length is acquired.

Measurements are performed on the sampled walks to determine features of the landscape, with a goal of predicting the difficulty of a given problem or explaining the observed performance [91, 123]. In effect, fitness landscape analysis is used to discern whether the neighbourhood function of two compared samples differs, and to present a characteristic description of their differences that may offer insight to their relative difficulty.

A local optimum is a solution which has a greater fitness than all of its neighbours. The epistasis of a problem is the level of inter-dependency of genes within the solution. A landscape with many local optima or a high epistasis is said to be rugged, while one with fewer optima is relatively smooth [113, 120]. A rugged landscape is typically considered to be more difficult to search as similar steps in the landscape have less correlation in fitness. An attractive basin is a series of solutions in the region of a landscape surrounding a local optimum which form a progressively improving gradient of fitness as it approaches the optimum itself. Small attractive basins are considered isolated, while larger attractive basins are considered to make the overall landscape smoother and thus easier to search [50]. A plateau is a series of solutions which have a difference of fitness within some specified sensitivity parameter, $\epsilon$. Here, $\epsilon = 0$ indicating the most sensitive measurement as in [123].

A number of information theoretic landscape measurements, as proposed in [121], are used in this thesis. It is shown in [123] that these measures correlate with the performance of EA using various mutation and crossover operators. To allow the application of these landscape analysis measures, the fitness space is commonly assumed to be statistically isotropic [50, 121, 128], this is the case in related work with ENNs as well [114, 120, 122]. For a detailed review of a variety of fitness landscape
measures see [91]. The remainder of this section defines the specific measures used in this paper.

The Information Content (IC) characterizes the variety and frequency of transitions along the random walk. A lower IC corresponds to more consistency among transitions which likely implies a less rugged landscape. The measure is calculated as:

$$ IC(\epsilon) := - \sum_{p \neq q} Pr_{pq} \cdot \log_6(Pr_{pq}) $$

where \( Pr_{pq} \) is the observed probability of transitioning from state \( p \) to state \( q \) in the random walk ensemble \( \Psi \):

$$ \Psi(i, \epsilon) = \begin{cases} 
1, & \text{if } f_i - f_{i-1} < -\epsilon \\
0, & \text{if } |f_i - f_{i-1}| \leq \epsilon \\
1, & \text{if } f_i - f_{i-1} > \epsilon 
\end{cases} $$

where \( f_i \) is the fitness observed in step \( i \) of the random walk, and \( \epsilon \) is a sensitivity parameter used to classify steps in the random walk as increasing, decreasing, or plateaued.

The Partial Information Content (PIC) measures the modality of the landscape as the ratio of peaks and valleys in the walk to the length of the entire walk itself. Large values here indicate a highly modal landscape while smaller values suggest a flat or smooth landscape.

$$ PIC(\epsilon) := \frac{\mu}{n} $$

where \( n \) is the length of the walk and \( \mu \) is the result of the summarized walk, \( \mu = \Phi(1, 0, 0) \) computed using the following recursion:

$$ \Phi(i, j, k) = \begin{cases} 
\Phi(i + 1, j, k + 1) & \text{if } j = 0, \Psi(i) \neq 0, i > n \\
\Phi(i + 1, j, k + 1) & \text{if } j > 0, \Psi(i) \neq 0, \Psi(i) \neq \Psi(j) \\
\Phi(i + 1, j, k) & \text{otherwise}
\end{cases} $$
which has the effect of counting all of the transitions from a decreasing walk to an increasing walk and vice versa while excluding plateaus and smooth edges.

Finally, the Density Basin Information (DBI) characterizes the smoothness of the landscape using an entropic measure of the regions of the ensemble which do not transition.

\[
DBI(\epsilon) := - \sum_{p \in \{1,0,1\}} P_r[pp] \cdot \log_3(P_r[pp])
\]  

(2.10)

where \(P_r[pp]\) represents the observed probability of two consecutive elements in the ensemble \(\Psi\) having the same state. Landscapes with large basins of attraction or plateaus will have a higher DBI.

### 2.4 Network Characteristics

This section defines the network analysis terms and measures discussed in this thesis. For additional measures from the network analysis field of literature see [46].

A network is represented as a graph \(G = (V, E)\) which is a set of nodes (or vertices) \(V\), connected by edges \(E\). An edge is a relation between two nodes over which transmissions occur. An edge may be directed, meaning it has unique source and destination nodes, and transmission may only occur in the indicated direction, or it may be undirected allowing transmission to travel in either direction. Symmetrical edges are directed but imply the existence of a complimentary edge from destination to source (opposite direction) for every given edge. Edges may be weighted, meaning each contains some value applied to transmissions along that edge, or unweighted meaning all transmissions occur unaltered between nodes.

The neighbourhood \(N_v\) of a node \(v\) is the set of all nodes connected to \(v\) by a single edge. In the case of directed edges a neighbourhood may be inbound \(N_v^i = \{u \in V | \exists e_{u,v}\}\) meaning all nodes with edges for which \(v\) is the destination node, or outbound \(N_v^o = \{u \in V | \exists e_{v,u}\}\) meaning all nodes with edges from which \(v\) is the source node. The degree of a node \(d(v)\) is equal to the size of its neighbourhood \(|N(v)|\). Similar inbound and outbound definitions of degree exist given directed edges.

A path is a sequence of nodes each consecutively connected in the graph. The path length \(pl(u, v)\) between two nodes \(u\) and \(v\) is the number of edges on the shortest
possible path between them. The average path length of a graph is calculated as follows:

\[
pl_{\mu} = \frac{\sum_{u,v \in V} p_l(u,v)}{|V| \cdot (|V| - 1)}
\]  

(2.11)

A graph is considered strongly connected iff there exists a path of finite length between every pair of nodes. A graph is considered fully connected (a.k.a. a complete graph) if there exists an edge between every pair of nodes. A clique is a fully connected portion of a graph. A cycle is a path following unique edges which begins and ends at the same node. A graph is considered acyclic if no such cycles exist.

The average local clustering coefficient is a measure of the cliqueishness of a graph as the proportion of unique connected triangles among each nodes’ neighbourhood. It is calculated as follows:

\[
lcc_{\mu} = \frac{1}{|V|} \cdot \sum_{v_i \in V} \frac{|\{e_{v_j,v_k} : v_j, v_k \in N_i, e_{v_j,v_k} \in E\}|}{|N_i| \cdot (|N_i| - 1)}
\]  

(2.12)

Centrality is a broad term encompassing a number of measurements of a node’s importance in the graph. A simple centrality measure is the degree centrality of a node which is simply the node’s degree, with larger values corresponding to a node that is more central in the network.

The closeness centrality of a node is measured as the inverse of the average path length of a given node to every other node in the graph. The decay centrality is a form of closeness centrality that uses a decay parameter \(0 < \xi < 1\) to weight short path lengths more heavily in the centrality than long ones as follows:

\[
decay(v) = \sum_{u \in V, u \neq v} \xi^{p_l(v,u)}
\]  

(2.13)

2.5 Random Network Generation

The social networks considered throughout this thesis take the form of randomly generated graphs, \(G = (V, E)\). This section discusses the graph generation algorithms used. This thesis considers only strongly connected graphs, with symmetric edges, and all nodes are assumed to have a self-loop edge. Each generator discussed here creates a random graph on the basis of the number of nodes \(|V|\), and the degree
parameter, $d$. For the purposes of experimental comparison, each graph was measured in terms of their average path length and local clustering coefficient (see Figure 6.7); this information is discussed here for reference.

**Nearest Neighbour**

The random nearest neighbour (NN) graph generator [117] creates a spatially clustered graph by first placing $|V|$ nodes randomly on a 2D plane. Then, uniformly at random, nodes from the graph are selected and connected to their nearest unconnected neighbour until a sufficient number of edges have been added. A degree parameter, $d$, determines the number of connections added to the graph as $d \cdot |V|$. Note that the use of symmetrical edges means that two edges are considered to exist between each pair of connected nodes, so only $\left\lfloor d \cdot |V|/2 \right\rfloor$ edges are added randomly. Additionally, all nodes contain self-loops, therefore the average node degree is $d + 1$, and the total number of edges is $|E| = d \cdot |V| + |V|$. This geographic construction creates a high degree of local clusters that remain spread out over large average path lengths.

**Scale Free**

A common property among many large real-world networks is the scale-free power-law distribution of node degrees [46]. This property is captured in the *preferential attachment* (PA) method of Barabási and Albert [6]. Starting from a fully connected seeding graph of $d + 1$ nodes, the graph is generated by adding one node at a time and attaching said node to $d$ other target nodes in the network with probability proportionate to the target nodes’ degree. With the use of self-loops and symmetrical edges, the total number of edges in the graph is $|E| = 2d \cdot |V| + |V|$. The clustering and path length are dependent upon the number of edges added to each new node, with increases in the degree parameter, $d$, leading to more compact networks with lower average path length and higher local clustering.
**Erdős-Rényi**

The random graph model of Erdős and Rényi [26] is defined as the selection of a graph containing $|V|$ nodes and $|E|$ edges from the set of all such possible graphs. Erdős-Rényi (ER) graphs are generated by independently selecting $|E|$ unique edges with uniform probability, and adding them to the graph of $|V|$ nodes. Given the use of symmetric edges and a self-loop on every node, the number of random edges is determined as $(|V| \cdot d)/2$. Thus the total number of edges $|E| = (|V| \cdot d) + |V|$. Of the considered graph types, the ER graphs possess the shortest average path lengths, and a very low average clustering coefficient.

**Small World**

The small world (SW) graphs are a common structure among many real-world social networks. The Watts-Strogatz method [127] provides a tunable means of generating small world networks relative to a parameter $0 \leq \beta \leq 1$. In brief, the graph is initialized as a $d$-regular ring lattice (thus $|E| = (|V| \cdot d) + |V|$). A proportion of the edges corresponding to $\beta$ are then adjusted randomly, with $\beta = 0$ corresponding to the unaltered ring, and $\beta = 1$ corresponding to a random graph. For the sake of comparison in this thesis, $\beta = 0.05$, giving the SW graphs a high average clustering and somewhat higher path length relative to the other graph types considered.

**2.6 Summary**

This chapter presented a collection of the required descriptions of topics used throughout the remainder of this thesis, with references to further reading for those interested. The next part of the thesis (Part I) defines the problem space to be considered. The part that follows (Part II) applies the background topics discussed in this chapter to the analysis of that problem space.
Part I

Problem Definition
Chapter 3

Network Control & Related Problems

“The conscious and intelligent manipulation of the organized habits and opinions of the masses is an important element in democratic society. ... Vast numbers of human beings must cooperate in this manner if they are to live together as a smoothly functioning society.”

Edward L. Bernays, Propaganada

3.1 Introduction

This chapter first presents the Network Control Problem (NCP) formalization in Section 3.2. Following this, the formalism is applied to a literature review of related problems in Section 3.3.

3.2 The Network Control Problem

The Network Control Problem (NCP) is a formalization of the general case of a range of related optimization problems that can be meaningfully described by the formal definition below. Common among all such problems is the use of limited quantities of external influence with the goal of directing the state of a social network or to have an optimal strategy of control within the constraints given.

The component-based definition presented here emphasizes the similarities among instances of the NCP class of problems. This supports the relation and common
solutions between instances, with the ultimate goal of developing a solution to social network control that is supported in increasingly more general forms of NCP.

Specifically, an NCP is a tuple $NCP = (f, G, DM, C)$ comprised of the following components:

- An **objective function** ($f$) that quantitatively evaluates the state of the system relative to the specific NCP’s objective,
- a **social network** ($G$) of connected individuals, each containing some state information,
- a **diffusion model** ($DM$) governing the communication of information between individuals, and
- a **control system** ($C$) specified as two primary components:
  - a **control system configuration** ($V_C$) specifying the structure of the control system and its connection to the network, and
  - a **control system behaviour** ($H_C$) that determines the content of limited interactions between the control system and the network.

Instances of the class of NCP arise through specification of the above components. Each of these components are described in turn below. The process of NCP simulation is detailed in Algorithm 2.

### 3.2.1 The Objective Function

A fitness function, $f(C)$, quantitatively evaluates the effectiveness of a given control system relative to the outcome of a problem specific network state utility function $U(t)$. Consequently, the related objective function $f$, defines the goal of the NCP under consideration.

Instances of the NCP formalism are distinguished by variations in their objectives and utility function, and subsequently the instantiations of the remaining NCP components. Three main classes of NCP instances are identified here: **Targeted**, **Trajectory**, and **Homeostatic**. In targeted NCPs, the goal is to drive the network state towards a specific goal state. A trajectory following NCP is an asynchronous targeted NCP, in which the goal is to have each node pass through a specific set or sequence
of states at some point during the simulation. Finally, a homeostatic NCP has a goal of maintaining the network within a narrow range of states when the natural diffusion of the network tends away from such stasis. The scope of experimentation in this thesis considers homeostatic NCP instances exclusively.

3.2.2 The Social Network

The social network is modelled as a graph \( G = (V, E) \), where \( V \) is the set of nodes and \( E \) is the set of directed edges including self loops. Each node \( v \in V \) represents an agent, and contains an (instantaneous) state value \( s(v, t) \in D_v \) at each time step \( t \) from the problem specific domain \( D_v \). The set of all node states cumulatively define the network state \( s(V, t) = \{(v, s(v, t)): v \in V\} \). Each edge \( e \in E \) is a directed relation between two nodes (i.e., \( e_{u,v} \) is the edge from node \( u \) to node \( v \)), and is associated with a weight value \( w(e_{u,v}, t) \). NCPs are not limited to any specific network structure.

3.2.3 The Diffusion Model

The Diffusion Model (DM) defines the behaviours of nodes as communicating agents in the social network. Here, a framework is presented that formalizes the existing definitions.

A DM can be defined as the tuple of behaviours \( DM(v) = (Sh(v), L(v, I), A(v)) \) for each node \( v \in V \). Note, this thesis is restricted to homogenous networks of agents. The first two parts of the DM correspond to two phases of communication in the social network as depicted in Figure 3.1. First, each node shares some information (typically in regards to its current state) along each outbound edge in its neighbourhood according to its sharing strategy \( Sh(v) \). Next, each node adjusts its state according to its learning strategy given the inbound state information, \( L(v, I) \). An optional adjustment phase, \( A(v) \), may contain operations that modify the neighbourhood or connected edge weights according to new information. NCPs which employ this phase are considered dynamic, whereas those that do not are considered static. The scope of this thesis is static NCPs. Each phase depicted in Figure 3.1 proceeds in synchronous fashion among the nodes, and the entire process is repeated for each time step until some termination condition is met. The process of state transition for a given node
Figure 3.1: Phases of the NCP execution. 1. State information is shared between neighbouring nodes in the network. 2. Nodes adopt shared information into their own states. 3. An optional adjustment of network edges. 4. The control system applies its control signals, $\sigma$, to the controlled node(s), based on their shared information, $I$.

can now be described formally as follows:

$$s(v, t + 1) = L(v, I(v, t))$$  \hspace{1cm} (3.1)$$

$$I(v, t) = \{(u, w(e_{u,v}, t), Sh(u)) \mid u \in N_i(v, t)\}$$  \hspace{1cm} (3.2)$$

Here $N_i(v, t) = \{u \mid e_{u,v} \in E\}$ is the inbound neighbourhood of $v$ at time $t$, and $I(v, t)$ is the set of information tokens transmitted from all such neighbouring nodes. An information token is the output of a node’s sharing strategy (e.g., the node’s state or other output information) paired with the weight of the edge along which the token travelled and the index of the source node.

Some additional definitions for diffusion models are as follows. A progressive DM is one in which the state of a node can only be changed a single time. Non-progressive models are those for which node states may change any number of times during simulation. The learning strategy of a node may define it as memoryless in that it models only a single state at a time, or it may contain a history of states. The domain of node state values is problem dependent, typically defined in a manner that
Algorithm 2 Pseudo-code of the NCP simulation process.

Require: $G = (V, E)$ is the social network to control

Require: $C$ is the control system

1: procedure NCP$(G,C)$
2: $V_C \leftarrow \text{Configure}(G,C)$
3: $t \leftarrow 0$
4: $U \leftarrow \text{Evaluate}(G,C)$
5: while not Termination$(U)$ do
6:     for $v$ in $V$ do $\triangleright$ Diffusion
7:         for $u$ in Neighbourhood$(v)$ do
8:             $I.$append$((u,\text{weight}(u,v,t),\text{Share}(u)))$
9:             $v.$state $\leftarrow \text{Learn}(v,I)$
10:            $G \leftarrow \text{Adjust}(G)$ $\triangleright$ Optional
11:           signal $\leftarrow \text{Control}(G,C)$
12:          for $v$ in $V_C$ do $\triangleright$ Control
13:              $v.$state $\leftarrow$ signal$[v]$
14:          $U \leftarrow \text{Evaluate}(G,C)$
15:        $t \leftarrow t + 1$

is relevant to the diffusion process itself. For example, in addition to state value, nodes may use trust, similarity, or other measures to alter the weights on their connected edges. Nodes may contain any type of value or potentially multiple values so long as the properties of the node state are accounted for in the learning and sharing DM components.

3.2.4 The Control System Configuration

The control system, $C$, is a set of one or more controllers that connect to the social network and alter its state via interactions at the node-state level. There are two parts to the control system corresponding to two dependent optimization subproblems. First, there is a configuration component that dictates the structure of the control system’s connections to the network and, by extension, the structure of the control
signals. Second, there is a \textit{behavioural} component that defines the content of each
time dependent control signal. While the exact nature of each of these two parts is
dependent on the goals of the problem at hand; the generalized details are specified
in this and the next section.

A controller configuration defines the locations in the network to which the con-
troller will apply its signals, and from which it observes the state of the network. The
controller may connect to nodes or edges in the network. The former concerns the
injection of information into the state space of the network, while the latter concerns
limiting information flow between nodes. Both forms of control ultimately effect the
state of the network towards some goal. While both forms of control have merit, this
thesis is concerned exclusively with node connections.

A control system $C$ may consist of one or more controllers $c \in C$. Distributed
controllers ($|C| > 1$, see Figure 3.2c,d) may operate in isolation or in communica-
tion with one another; they may be homogeneous or heterogeneous with respect to their
design and their decisions. The scope of this thesis however, is limited to single
controller systems ($|C| = 1$).

The set of nodes connected to a controller $c$, are known as controlled nodes, and
are defined as $V_c = \{ v \in V \cup P \mid \exists e_{c,v} \}$, where $P$ is the set of proxy nodes (described
below). The set of all controlled nodes is $V_C = \{ V_c \mid c \in C \}$. The state of each
controlled node is directly set to the value of the control signal it receives, that is
$s(v, t) = \sigma(c, v, t)$ iff $v \in V_c$, where $\sigma(c, v, t)$ is the component of the control signal
from controller $c$ for node $v$ at time $t$. Therefore, the structure of the control signals
is confined to a single output for each controlled node per time step.

The control system may control the state of the social network \textit{directly} or \textit{indirectly}. Direct control (see Figure 3.2a,c) indicates that a controller will directly set
the states of controlled nodes \textit{within} the network at each time step. Indirect control
(see Figure 3.2b,d) indicates that a controller is setting the states of nodes \textit{outside}
of the network, which then communicate with nodes in the network via the diffusion
model. These external ‘proxy’ nodes ($P$) have state and connections like a typical
node, but neither alter their own state as a result of influence diffusion, nor does their
state contribute to the network state. The distinction between indirect and direct
control is that in the direct control scheme $V_C \subseteq V$, whereas in the indirect $V_C = P$. This thesis considers only the former.

Constraining the search for an optimal configuration, each node $v \in V$ is associated with a cost, $\text{cost}(v)$, and each controller is given a finite budget, $B$, with which to select nodes for the controlled set:

$$\sum_{v \in V_C} \text{cost}(v) \leq B$$

The node cost formula varies by implementation in existing instances of NCP from unit costs per node to problem-defined cost distributions. This thesis considers the cost of connecting a controller to a node to be equal to the number of outbound edges originating from the node, where the degree of the node is considered to be an estimate of the influence of that node [19, 27].

3.2.5 The Control System Behaviour

The behavioural component $H_c$ of a controller $c$ is responsible for determining a mapping between the inputs drawn from the social network and the content of the disseminated control signals, $I(V_c, t) \rightarrow (\sigma(c, v, t) \mid v \in V_c)$. The means used to generate these values are not restricted in the NCP definition.

The output values specified are restricted to the domain of the controlled nodes’ allowable states, $\mathcal{D}_\sigma = \mathcal{D}_v$. The mapping is executed once per time-step, thereby generating an input-dependent time series of heterogeneous values for each controlled node.
The input information regarding the network that is visible to the control system, \( I(V_C, t) \), is problem specific corresponding to the control system’s available sensors. Immediately prior to determining a control signal, the control system probes the visible state of the network. Throughout the remainder of this thesis, the information regarding the network that is visible to the control system, \( S(C, t) = s(N(V_C, t')) \), is the collection of all states of nodes neighbouring controlled nodes at the time of command. The parameter \( t' \) is used here to denote the inclusion of information passed between nodes during the sharing phase.

A locally optimal behaviour is one which optimizes the objective function, \( f \), over the set \( V_C \). A globally optimal behaviour is one which can be shown to be the best choice of action relative to the optimization of \( f \) over the set of all nodes \( V \), given the constraints of the problem.

### 3.3 Existing Network Control Problems

The Network Control Problem and Diffusion Model formalisms defined in the previous section are derived from commonalities among existing problems and models. This section illustrates their usage by presenting a number of existing problems and models as members of these unified formats.

#### 3.3.1 Influence Maximization Problem

The Influence Maximization Problem (IMP) is a targeted NCP first presented as an optimization problem by Richardson and Domingos [25, 98] and later formalized by Kempe et al. [57, 58]. The objective of the IMP is to select a static configuration of nodes in the network that maximizes the spread of a given node state.

**Objective Function:** Formally, given a social network graph with node states \( \{0, 1\} \), the goal is to find:

\[
arg\max_{V_C \subseteq V} R(V_C)
\]

(3.4)

given that \( s(v, t) \) is the state of node \( v \) at time \( t \), \( s(v, 0) = 1 \ \forall v \in V_C \) and \( s(v, 0) = 0 \ \forall v \notin V_C \), subject to the given diffusion model and the budgetary constraint (see Equation 3.3), where \( R(V_C) = E[|V^1(t^*)|] \) is the expected return in terms of the
number of nodes with state 1 at the time of quiescence $t^*$ given the initial set $V_C$.  

**Network Structure**: The IMP is commonly studied in a 2-state variant form ($\mathcal{D}_\infty = \{0, 1\}$), where nodes are initially all inactive, and become active via the diffusion of control signals. Variants of the IMP with greater than two states exist, in which each additional state represents a competing source of influence (e.g., rival product) in the network [10, 15]. The IMP has been studied over a variety of real-world network structures including co-authorship graphs [15, 57, 68], blog links [61], and recommender networks [98, 25].

**Diffusion Model**: A number of existing DMs have been applied to the IMP. Prominent examples include the progressive variants of the Linear Threshold (LT) and Independent Cascade (IC) models, presented in [57, 58]. In the LT, a node becomes activated if there is sufficient pressure from activated neighbouring nodes. Formally,

$$ S_h(v) := s(v, t) $$

$$ L(v, I) := \begin{cases} 1, & \text{iff } \sum_{(w,s,u) \in I(v,t)} s \cdot w \geq \theta_v \\ 0, & \text{otherwise} \end{cases} $$

where $\theta_v$ is the threshold of activation for node $v$.

In the progressive IC, once a node is activated, it has a one-time chance of activating each neighbouring node on the subsequent time-step. Formally,

$$ S_h(v) := \begin{cases} s(v, t), & \text{iff } s(v, t-1) \neq s(v, t) \\ 0, & \text{otherwise} \end{cases} $$

$$ L(v, I) := \begin{cases} 1, & \text{with probability } \sum_{(w,s,u) \in I(v,t)} s \cdot w \\ s(v, t), & \text{otherwise} \end{cases} $$

It is proven in [57] that the conclusions made in regard to the IC and LT in the progressive case, such as the optimality of the greedy approximation algorithm (see below), extend to the non-progressive implementations of the model. Despite this, the non-progressive models have received very little attention in the literature. This trend is noted in [29], where efforts are made to extend the results of [57] to the
non-progressive strict majority model. This is a model equivalent to the LT with \( \theta_v = \lceil \frac{d(v)+1}{2} \rceil \). Due to the possibility of activated nodes becoming deactivated, the objective in this model is to seek a steady state which maximizes the number of activated nodes.

Even-Dar and Shapira [27] consider the problem of influence maximization in the context of the (non-progressive) Voter Model (VM). In the VM, as introduced in [42], a node changes its own state by adopting that of a random neighbouring node. Formally,

\[
Sh(v) := s(v, t) \tag{3.9}
\]

\[
L(v, I) := \begin{cases} 
1, \text{ with probability } \frac{|\{s=1|(u,s,u)\in I(v,t)\}|}{|I(v,t)|} \\
0, \text{ with probability } \frac{|\{s=0|(u,s,u)\in I(v,t)\}|}{|I(v,t)|} 
\end{cases} \tag{3.10}
\]

**Control System Configuration:** The influence maximization problem corresponds to the search for an optimal configuration with which to initiate an influence cascade. It is shown to be NP-hard regardless of the diffusion model used [15, 57]. As such, many techniques have been suggested for addressing the problem effectively and efficiently. A greedy hill-climbing approach to node selection is described in [57, 58], and is experimentally demonstrated to out-perform high-degree and centrality heuristics on the IC and LT models. The greedy algorithm iteratively adds to the initial set, the node with the highest marginal gain in influence diffusion across the network, as estimated through extensive Monte Carlo simulations. It is shown that the greedy algorithm approximates the optimal set to within a factor of \( (1 - 1/e) \). Leskovec et al. [69] consider a proportional version of the greedy algorithm, which factors in the node cost. That is, a node \( v \) is selected such that:

\[
\argmax_{v \in V - V_{C_{m-1}}} \left( \frac{R(V_{C_{m-1}} \cup \{v\}) - R(V_{C_{m-1}})}{\text{cost}(v)} \right) \tag{3.11}
\]

where \( V_{C_{m}} \) is the set of selected nodes of size \( m \), and \( R(V_{C_{m}}) \) is the return function given the initial set \( V_{C_{m}} \). The bounds are proven to be within a factor of \( \frac{1}{2} (1 - 1/e) \) of the optimal solution. Leskovec et al. also suggest the use of lazy evaluation to take advantage of the sub-modular property of the problem, and gain a significant speed
up in place of evaluating the marginal gain of every node at every step in the greedy algorithm.

An alternative means of increasing the efficiency of evaluation is the use of approximation heuristics. Gui-sheng et al. [37] compare the performance of Genetic Algorithms, Differential Evolution, and Particle Swarm Optimization with the preferential attachment heuristic. Their findings show that the highest quality performance is achieved using genetic algorithms. However, the results are neither compared in terms of run time nor against the greedy heuristic for quality. Nevertheless, approximation heuristics are a promising approach to this NP-hard combinatorial optimization problem.

Although Kempe et al. demonstrate that the degree and centrality heuristics are inferior to the greedy algorithm for the LT and IC models, several extensions have been applied to the greedy algorithm using graph measures to improve speed, quality, or both. Chen et al. [18] suggest the use of degree discount heuristics derived for the IC and LT models. These provide a considerable increase in speed over the greedy algorithm while improving the return on influence over standard degree or centrality heuristics. It is shown in [17] that the degree discount heuristic is the most scalable to large networks, when compared to the lazy-evaluation greedy algorithm of [69], a shortest-path heuristic from [60], the PageRank algorithm [89], and the improved arborescence algorithm of [17]. Finally, in their analysis of the voter model, Even-Dar and Shapira [27] prove that when all nodes have the same cost, the degree heuristic is an optimal solution to the IMP.

A number of additional means have been considered for measuring the spread of information using various deterministic and nondeterministic diffusion models. Hajian applies a recursive method for determining the expected outcome of influencing a given set on a complex diffusion model in [38]. Kitsak et al. [62] show through use of infection models that the most influential spreaders are not necessarily the highest connected, but the best distributed that most significantly impact the diffusion process.

**Control System Behaviour:** The objective for the IMP presumes the optimal control behaviour, thus eliminating the search for an optimal instantiation of this NCP
component. The control system outputs a single value (active) to every controlled node. Formally,

$$\sigma(c, v, t) = 1, \forall v \in V_C$$  \hspace{1cm} (3.12)

**Discussion:** The IMP is a prominent combinatorial optimization problem. It represents a simplified version of the general NCP by removing the search for an optimal behavioural component, allowing a great deal of research effort to focus solely on the configuration component. However, disregarding variety in the behavioural component, as well as overlooking the dependence of the two component subproblems, ultimately limits the utility of the IMP in terms of leading to a complete solution for the NCP.

### 3.3.2 Structural Controllability

The structural controllability of large scale networks was studied by Liu et al. [75]. This is another example of a targeted NCP. The objective is to select the minimum required set of controlled nodes in order to deliver independent signals to every node in the network, given knowledge of the exact deterministic transmission process.

**Objective Function:** The goal of structural controllability is to find the minimum set of controlled nodes which maintain Kalman’s controllability rank condition [52, 75]:

$$\text{argmin}_{V_C \subseteq V} (\text{rank}(M) = |V|)$$  \hspace{1cm} (3.13)

$$M = (E_C, EE_C, E^2E_C, ..., E^{|V|-1}E_C)$$  \hspace{1cm} (3.14)

where $E_C = \{e_{c,v} | c \in C, v \in V_C \}$ is the matrix of edges connecting the set of controllers $C$ to the nodes $V$, and $E^2$ is the square of the edge matrix $E$.

**Network Structure:** The controllability of a large number of real world networks as well as instances of the Erdös-Rényi and scale-free network models was studied in [75]. It was discovered that the controllability of a network was encoded by its underlying degree distribution, with networks of higher average degrees requiring larger configurations to control.
**Diffusion Model:** The problem is defined using linear time-invariant dynamics ($\mathbb{D}_{\mathbb{R}} = \mathbb{R}$). In terms of the DM framework:

$$Sh(v) := s(v, t)$$  \hspace{1cm} (3.15)

$$L(v, I) := \sum_{(w, s, u) \in I(v, t)} s \cdot w$$  \hspace{1cm} (3.16)

Non-linear network dynamics are estimated using successive linear steps in [126].

**Control System Configuration:** In [76] it is shown that the minimum number of required control nodes is 1 if a perfect directed matching exists on the graph. Otherwise, the minimum number and location of the control sites can be determined efficiently by computing a directed maximum matching on the graph, and then applying control to any unmatched nodes. Using this system it was observed that the unmatched nodes were more commonly of low degree than of high degree.

**Control System Behaviour:** The determination of specific signals to be passed into the network is not considered as part of the structural controllability problem. Given a diffusion model defined as linear time-invariant dynamics, solutions to the specific control problem can be determined analytically by solving the linear equations of the system. This is prohibitive, however, as the problem size and complexity increase, or given an unknown or stochastic diffusion process.

**Discussion:** The structural controllability problem combines the fields of network analysis and control theory and, as such, was a significant motivation for the development of the NCP. The lack of behavioural effects in this problem may put the derived results at odds with other NCPs. For example, the strong correlation found between a network’s degree distribution and its controllability in [75], may be at odds with the findings of [57] that a greedy selection of controlled nodes w.r.t. marginal influence gain outperforms a degree distribution heuristic given the DMs used therein.

Like the IMP, the structural controllability problem omits the search for appropriate control system behaviour. However, by leaving this component unspecified (as opposed to rigidly defined), there is an opportunity to extend the work to consider more complex objectives.
3.3.3 $m$-Effectors Problem

The $m$-Effectors problem is a targeted NCP presented by Lappas et al. [68]. The objective is to find the set of controlled nodes of size $m$ which is most likely to lead to the observed activation pattern.

**Objective Function:** Formally, given a social network graph with node states in the domain $D_\approx = \{0, 1\}$, where inactive nodes have a state 0 and active nodes have state 1, and the desired activation vector $s'(V, t^*)$, the goal is to find a configuration $V_C$ such that the following error function is minimized:

$$\text{argmin}_{V_C \subseteq V} \sum_{v \in V} |s'(v, t^*) - Pr(v = 1|V_C)|$$

(3.17)

where $|V_C| \leq m$, and $Pr(v = 1|V_C)$ is the probability that node $v$ is active at the end of the diffusion process, given that all of the nodes in $V_C$ were activated initially.

**Network Structure:** Lappas et al. [68] study the 2-state $m$-Effectors problem over tree-structured networks (directed acyclic graphs), where the weight on edge $u,v$ is considered to be the probability of node $u$ activating node $v$. Other directed graph structures were converted to tree structures such that the resulting tree spans all active nodes in the target activation pattern and maximizes a total likelihood over all included edge weights. Conversion to a tree structure is required for their proposed solution (see below).

**Diffusion Model:** While the $m$-Effectors problem is studied exclusively on the progressive IC model of [57], it is claimed that some results from [68] can be generalized to any probabilistic DM.

**Control System Configuration:** Three solutions to the $m$-Effectors problem are proposed in [68]. The first is an optimal dynamic programming algorithm that recursively evaluates the cost of adding a given subtree to the set of initially active nodes. The second sorts all nodes according to the error function evaluation when they are activated exclusively, and then picks the $m$ best nodes. The third solution selects the $m$ active nodes with the highest weighted out-degree. All three are shown to perform comparably, with the optimal algorithm requiring more space and time for greater accuracy.

\footnote{There it was called the $k$-Effectors problem, but is renamed here for consistency.}
Control System Behaviour: The control system of the $m$-Effectors problem sets the state of all selected nodes to be active, as in the IMP.

Discussion: The $m$-Effectors problem is a generalization of the IMP. The IMP can be said to be an instance of the $m$-Effectors problem in which the target $s'(V, t^*)$ is maximally active. To that end, both problems lack the requirement for a control system behaviour search, instead dictating activation of the controlled set. The heuristics used in configuration here are similar to those used in the IMP: greedy w.r.t. marginal gain, and w.r.t. node degree. Both are demonstrated to be effective approaches to each of these problems.

3.3.4 Immunization Problem

The Immunization Problem (IP) studied by Cohen et al. [19], and Pastor-Satorras and Vespignani [90], is an homeostatic NCP. The IP is defined as determining a targeted immunization strategy; that is, selecting an optimal set of nodes from the network to immunize in order to prevent the spread of an infectious diffusion process.

Objective Function: The objective of the IP is to select a minimal subset of the population to immunize such that the the number of uninfected individuals is maximized at the end of simulation, $t^*$. Formally,

$$\argmax_{V_C \subseteq V} \left( \sum_{u \in V} s(u, t^*)|s(v, t) = 1 \ \forall v \in V_C, \forall t \right)$$

while minimizing the budget parameter, $B$, from the NCP constraint (Equation 3.3), with $cost(v) = 1 \ \forall v \in V$.

Network Structure: Infection models are prolific in the influence literature [11] owing to their long history and real world applicability [5]. As such, the analysis is accordingly well spanned across network types, both generated and real. The IP is considered on small world in [90], and on scale free in both [19] and [90] due to the commonality of that structure in real-world social networks. Edges between nodes are typically undirected and unweighted due to the use of model-specific global transmission rates in the standard infection DMs.

Diffusion Model: The domain of standard infection models varies among several interrelated models. The construction of each specific model is commonly described
by an abbreviation of the domain states $\mathbb{D} = \{MSEIR\}$ (Passively Immune, Susceptible, Exposed, Infected, Removed), in a specific ordering which explains the infection process under consideration [40]. Here, the SIS and SIR models are discussed, however the states are translated to $\mathbb{D}_{\mathbb{S}} = \{1, 0, -1\}$. An individual with state 1 represents an uninfected (susceptible) individual, and state 0 represents an infected individual. The third state, -1, corresponds to a removed individual. Initial conditions, including the introduction of the infection, are considered dependent on the specific instance under consideration.

The IP is studied under the Susceptible-Infected-Susceptible (SIS) DM in [90]. Healthy nodes become infected with probability $\nu$ if they are connected to infected nodes, and infected nodes are cured with probability $\delta$ becoming susceptible to infection once again. Both $\nu$ and $\delta$ are problem specific parameters. Formally,

$$S_h(v) := s(v, t) \quad \text{(3.19)}$$

$$L(v, I) := \begin{cases} 0, & \text{with probability } \nu, \text{ if } |N^0(v, t)| \geq 1 \\ 1, & \text{with probability } \delta, \text{ if } s(v, t) = 0 \\ s(v, t), & \text{otherwise} \end{cases} \quad \text{(3.20)}$$

where $N^k(v, t) = \{u \mid s(u, t) = k \in I(v, t)\}$ is the set of nodes states equal to $k$ in the information token $I$ received by node $v$ at time $t$.

Another DM from the standard infection literature is studied in [19], the Susceptible-Infected-Removed (SIR) model. The SIR works like the SIS, except infected nodes do not become susceptible again, and are instead removed from the network modelling dead or cured and immune individuals.

$$S_h(v) := s(v, t) \quad \text{(3.21)}$$

$$L(v, I) := \begin{cases} 0, & \text{with probability } \nu, \text{ if } |N^0(v, t)| \geq 1 \\ -1, & \text{with probability } \delta, \text{ if } s(v, t) = 0 \\ s(v, t), & \text{otherwise} \end{cases} \quad \text{(3.22)}$$
Following the state transition phases of the DM framework, the SIR invokes the optional network adjustment phase to extract the removed nodes from the network:

\[ A(v) := N_v(t + 1) = N_v(t) \setminus \{ u | s(u, t) = -1 \} \]  \hspace{1cm} (3.23)

**Control System Configuration:** Targeted immunization strategies for scale-free networks are presented in [19] and [90], and two different configuration strategies were employed. In the former, nodes are selected for immunization proportionally according to their degree. In the latter, the immunized node is determined by first selecting a node from the graph at random, and then immunizing one of its neighbouring nodes. These strategies have similar effects of selecting the most highly connected individuals, with the latter being a stochastic estimate of such based on local information.

**Control System Behaviour:** In the standard IP literature, immunized nodes are removed from the network thereby preventing infection spread over their associated links. This behaviour can be modelled as a direct control NCP as follows,

\[ \sigma(c, v, t) = 1, \forall c \in C, \forall v \in V_c, \forall t \]  \hspace{1cm} (3.24)

This effectively prevents the infection state from spreading through any controlled nodes.

**Discussion:** The IP is limited to a search for the optimal configuration. The control system behaviour is again dictated explicitly by the problem definition, and, like the IMP, takes the form of explicit single state output. However, the IP contrasts with the IMP, in that the immunization state does not spread as the active state of the IMP does. So, while both non-dynamic control systems directly output the state to be maximized to a set of desirable nodes, the means by which this alters the diffusion of information in the network differs sharply. One creates the root of a diffusion tree, while the other breaks links in the network; one initiates diffusion, while the other prevents it.

### 3.4 Summary

The NCP formalism provides a modular description of a field of related instances of this general class of problems. The specific details can vary greatly between instances
however they all relate in the high-level objective of optimizing a control system to influence the dynamic state of a network. The foregoing discussion brings together some of the existing problems that can be considered to fit in the NCP framework. The intent here is to illustrate the variety and means by which the NCP framework encompasses existing literature, and elucidate the relationships between diverse problems.

In all observed implementations of existing NCP instances there has yet to be defined an instance that requires the search for an optimal control system behaviour component. Chapter 4 will address this shortcoming by introducing a novel instance of the NCP.
Chapter 4

The $\theta$-Consensus Avoidance Problem

“One by one they had all burst, and their bursting showed that they had been temporary solutions to long-term problems, maybe evasions of those problems, distractions. With so many bubbles - so many people chasing ephemera, all at the same time - it was clear that things were fundamentally not working.”

George Packer, The Unwinding

4.1 Introduction

The definition of the Network Control Problem in the previous chapter inspires the development of a testbed instance that entails the complete objective of both configuration and behavioural subproblems. This chapter introduces the $\theta$-Consensus Avoidance Problem (\(\theta\)-CAP) as such. Section 4.2 revisits the motivation for this problem originally discussed in Section 1.2. Section 4.3 then defines the $\theta$-CAP in the standard NCP format. A preliminary analysis of the $\theta$-CAP is discussed, including a mapping of the problem to the well-known 0/1-knapsack problem, as well as the theoretical definition of diffusionless class of problem instances within the $\theta$-CAP parameter space.
4.2 Motivation

Standard mechanisms of information diffusion in social networks (e.g., group polarization, peer pressure, etc.) can lead the network to a state of decreased diversity and therefore vulnerable to catastrophe. As examples, a consensus of opinion in a financial market can lead to a bubble or a market run [41, 78], a lack of diversity in genetic expression can leave a population susceptible to disease [44], a lack of choice in a product market can lead to monopolistic prices [54], or a consensus of opinion regarding the importance of a particular policy can undermine public input of broader government policies [112].

Social propaganda exploits the standard mechanisms of information diffusion in social networks to the intent of the propagandist [49]. The effectiveness of social propaganda is derived from the notion that “people accept statements that they hear, with relatively little critical appraisal” [31]. Propaganda utilizes this behaviour by presenting individuals with a well crafted image of society, with the hopes that individuals will internalize their perceived views of those around them [49]. This is particularly true when the manipulative intent of the propagandist is not salient to the individual [88, 108]. The Voter Model (VM, considered here) simulates this process via discrete opinion states, either completely for or against some issue. The opinion state of a single individual is then free to switch from one extreme to the other by randomly adopting opinions from their immediate neighbourhood. This models individuals as particularly gullible, and ultimately leads to a consensus of opinion throughout the social network [27]. Given the reputation of propaganda as sinister or unethical, this model presents an auspicious environment for manipulation. However, in line with the notion of propaganda as a neutral tool biased only by those that use it, the task considered in this work is the balance of opinions in the social network. That is, social propaganda, called social influence throughout this work, is used in this instance as a means of preventing catastrophe (e.g., mass panic or bubble behaviour), thereby facilitating a productive society.
4.2.1 Pole Balancing Problem

The $\theta$-CAP, as a testbed for the NCP, is derived from the failure avoidance class of problems from the adaptive control literature, most notably the pole-balancing problem [2, 72, 99] (see Figure 4.1). The objective of this type of problem is to minimize a state-utility function over an infinite time span, in a system where one or more failure states exist corresponding to terminal states with high state-values.

The pole balancing problem tasks a control system with balancing a hinged pole affixed to a moving cart via horizontal forces applied to the cart. The pole is subject to standard physical laws of gravity and thus naturally tends towards the failure state of falling over. Formally, the objective is to minimize the following utility function:

\[
U(t) = \begin{cases} 
0, & \text{if } |a(t)| < \theta_a \text{ and } |x(t)| < \theta_x \\
1, & \text{otherwise}
\end{cases}
\]

where $0^\circ < \theta_a \leq 90^\circ$ and $0 < \theta_x$ are bounds on the pole angle and cart position respectively. The potential knowledge of the system available to the control system includes the parameter $(\theta_a, \theta_x)$, the state values $(a(t), x(t))$, their rates of change, and the physical mechanics of the system. In practice, however, the available knowledge can vary from complete to partial knowledge, or even a single failure signal.
4.3 Problem Definition

The $\theta$-CAP is a novel homeostasis-type NCP derived from the pole balancing problem. The following subsections each describe this problem in terms of the NCP component definition from Chapter 3.

4.3.1 Objective Function

The objective of the $\theta$-CAP is to maintain the state of a social network within a bounded range of equilibrium despite its tendency towards consensus. The $\theta$-CAP simulation proceeds under the standard NCP phases (Figure 3.1) until the state of the network exceeds a consensus threshold parameter, $\theta_G$ (see Figure 4.2). Formally, the goal of the $\theta$-CAP is to find a time series of control signals that will minimize the following state utility function over an infinite time span:

$$U(t) = \begin{cases} 0, & \text{if } \frac{|V^1(t)| - |V^a(t)|}{|V|} \leq \theta_G \\ 1, & \text{otherwise} \end{cases}$$

(4.2)

where $V^k(t) \subseteq V$ is the set of nodes at time $t$ with state set to $k$, $\theta_G$ is a threshold parameter representing the upper bound of allowable disparity in the network, and $0 \leq \theta_G < 1$. A terminal failure state occurs when the proportional difference of node states in the network exceeds $\theta_G$. The fitness, $f(C, V_C)$, of a control system $C$ applied to the $\theta$-CAP, is the number of time-steps that the control system maintains the network within the first case of Equation 4.2. This is known as the run length fitness.

$$f(C, V_C) = \min(t^*): U(t^*) = 1, \ 0 < t^*$$

(4.3)

4.3.2 Network Structure

This definition of the $\theta$-CAP does not constrain the problem to any specific network structure (several types of random graph are considered in Chapter 6). Owing to the diffusion model used (see below), all considered networks contain a self-loop on each node, and use symmetrical unweighted edges. As with the pole-balancing problem system information, the information regarding the network that is available to the
control system for this problem can vary from knowing nothing to knowing the network class or complete structure. The configuration heuristics applied to the $\theta$-CAP in Chapter 6 vary in their assumed knowledge from nothing to the requisite calculation of all-pairs shortest path. Behavioural implementations in this thesis are limited in their sensory input to the neighbourhood of their controlled set, as discussed in Section 3.2.4.

4.3.3 Diffusion Model

The $\theta$-CAP can be instantiated with any non-progressive diffusion model. The domain of node states in the standard $\theta$-CAP variant is $\mathbb{D} = \{0, 1\}$. The non-progressive requirement (see Section 3.2.3) ensures that nodes in the social network are not biased towards either state, that is, the network state should be capable of falling to consensus of either allowable state in the node domain. It is intended that the DM used should fall to consensus within the entire network if left uncontrolled. While solutions to the $\theta$-CAP may exist on DMs that do not exhibit this property, special consideration must be given in order to distinguish between successful control and natural convergence to non-consensus states.

This thesis considers the $\theta$-CAP exclusively under the dynamics of the Voter Model (VM). The VM is a simple DM with a guarantee to converge to consensus (if left uncontrolled) with high probability in $O(|V|^3 \log(|V|))$ time, and guaranteed in $O(|V|^5)$ time \cite{27, 42}. Specifically, the state change of a node occurs by adopting the...
state of a random neighbouring node at each time step as follows:

\[ Sh(v) := s(v, t) \]  

(4.4)

\[ L(v, I) := \begin{cases} 
1, \text{ with probability } \frac{|\{s=1|(w,s,u)\in I(v,t)\}|}{|I(v,t)|} \\
0, \text{ with probability } \frac{|\{s=0|(w,s,u)\in I(v,t)\}|}{|I(v,t)|}
\end{cases} \]  

(4.5)

where \( Sh(v), L(v, I), \) and \( I(v, t), \) are the sharing strategy, learning strategy, and set of information tokens respectively (see Figure 3.1 and Equations 3.1 and 3.2).

### 4.3.4 Control System Configuration

The control system configuration is limited by the NCP budget constraint:

\[ \sum_{v \in V_C} cost(v) \leq B \]  

(4.6)

The node costs for the \( \theta \)-CAP definition are not constrained to a specific function. However, the scope of this thesis is such that the cost of each node is equal to its degree. This function was chosen because 1) the degree of a node has been shown to be a simple indicator of a node’s influence in the network [18, 19, 27], and 2) for consistent costs with the indirect control style (see Section 3.2.4).

A number of potential solutions for the configuration search exist in the literature (see Section 3.3). Variations of approaches such as structural controllability [75], clustering [20], greedy w.r.t. influence gain [57, 69] or node degree [19, 27], or an evolutionary search [37] may all be considered appropriate. While these techniques have been effective for related problems such as the IMP, the goal of the \( \theta \)-CAP is not maximization and so the optimal configuration is expected to differ as well.

Optimization of the configuration component of the \( \theta \)-CAP maps to a variant of the well-known knapsack problem [27, 56, 77]. Optimization of the standard 0/1-knapsack problem is an NP-hard constrained subset selection problem with the following definition\(^1\):

\[ \text{Maximize } \sum_{v} f_i \cdot id_i, \text{ subject to } \sum_{i=1}^{V} cost(v_i) \cdot id_i \leq B, \]  

(4.7)

\(^1\)These equations have been adapted to fit the symbolic nomenclature of this thesis.
where, \( id_i \in \{0, 1\} \) corresponds to an element of an identification vector in which a value of 1 signifies the selection of an item, \( f_i \) is the contribution of item \( i \) to the fitness of the total subset commonly called the value of item \( i \), and \( cost(v_i) \) is the cost of adding item \( i \) to the solution, commonly referred to as the weight of item \( i \). Note that the maximized sum term here is equivalent to the configuration problem given that \( V_C = \{v_i \in V|id_i = 1\} \).

The configuration subproblem of the \( \theta \)-CAP is equivalent to a version of the knapsack problem in which the desirability measure of a single node/item, \( f_i \), is unknown. The value of the selected set must instead be determined experimentally via execution of the dependent behavioural simulation. Additionally, selection of individual nodes cannot be assumed independent of one another in this problem as items are in the standard knapsack problem. For example, two nodes that were close together were found to be less valuable than two nodes that were far apart in terms of their propensity to cause a cascade of diffusion in [18, 38, 62].

### 4.3.5 Control System Behaviour

The control system behaviour component is required to determine an optimal mapping between input node states and control signals. The visible input state of the network for the controller at each step is equal to the neighbourhood of all controlled nodes, \( I(V_C, t) = \{s(u, t)|u \in N(v, t), v \in V_C\} \). An optimal mapping for the \( \theta \)-CAP is one that minimizes the utility function (Equation 4.2) over an infinite time span, or put another way, one that avoids \( \theta \)-consensus in the network for as long as possible.

This thesis considers the application of several heuristic and metaheuristic algorithms for this component. These are described in Chapter 5.

### 4.3.6 Discussion

The control system configuration is governed directly by the budget parameter \( B \), and indirectly by the consensus threshold parameter \( \theta_G \). Larger values of \( B \) allow a greater number of nodes to be controlled, and larger values of \( \theta_G \) refer to larger allowable ranges of imbalance. Therefore, intuitively, as the parameters \( \theta_G \) and \( B \) increase, the difficulty of the problem decreases. This provides a two-dimensional
gradient of difficulty over instances of the $\theta$-CAP.

**Theorem 1.** The control system in a direct control scheme is capable of sustaining the network from reaching consensus according to $\theta_G$, regardless of the states of the uncontrolled nodes, if the following holds:

\[
\theta_G + \frac{|V_C|}{|V|} \geq \theta_C + 1
\]  

(4.8)

where $\theta_C \left( \frac{1}{|V|} \leq \theta_C \leq \theta_G \right)$ is the upper bound of allowable difference of opinions among the subset of controlled nodes, $(V_C)$.

**Proof.** Let $V_U = V - V_C$ be the set of uncontrolled nodes in the network, and let $\theta'_G = \lfloor \theta_G \times |V| \rfloor$ be the upper bound on allowed imbalance without consensus in terms of number of nodes. A *locally optimal* controller for the $\theta$-CAP is defined here as one that maintains a minimal amount of imbalance amongst its controlled nodes. Given that the number of controlled nodes may be even or odd, a locally optimal controller can at best achieve an imbalance of 1 node in the general case.

If $|V_C| = |V| - \theta'_G + 1$ then $|V_U| = \theta'_G - 1$. A complete consensus among the set $V_U$ would leave the network two nodes short of a failure state, thus requiring an imbalance within $V_C$ of at least two nodes before failure could occur. A locally optimal controller will provide at most an imbalance of one. Therefore, a locally optimal controller with direct control to a set $V_C$ of sufficient size can prevent consensus within the entire social network without considering the states of uncontrolled nodes if the following inequality is satisfied:

\[
|V_C| \geq |V| - \theta'_G + 1
\]  

(4.9)

Furthermore, the amount which $|V_C|$ exceeds $|V| - \theta'_G$ defines the amount of allowable imbalance within $V_C$. Therefore the controller can sustain the network from reaching $\theta'_G$-consensus if the problem instance satisfies:

\[
\theta'_G + |V_C| \geq \theta'_C + |V|
\]  

(4.10)

where $1 \leq \theta'_C \leq \theta'_G$ is the upper bound on allowable imbalance within the set of controlled nodes. Normalizing this equation by $|V|$ yields Theorem 1. □
For instances in which Equation 4.8 holds true, the controller can prevent \( \theta \)-consensus using a non-specific set of nodes, even if every uncontrolled node has converged to a single state. This defines the class of direct-control \( \theta \)-CAP instances designated here as diffusionless instances. Instances for which Equation 4.8 is false are termed diffusive.

Formulation of Theorem 1 using \( \theta_C \) defines a simpler objective for the diffusionless class of instances. The goal of these is to minimize the following state utility function:

\[
U_C(t) = \begin{cases} 0, & \text{if } \frac{|V^2_C(t)| - |V^0_C(t)|}{|V|} \leq \theta_C \\ 1, & \text{otherwise} \end{cases}
\] (4.11)

where \( |V^k_C(t)| \) is the number of controlled nodes with state equal to \( k \) at time \( t \) and \( |V^0_C(t)| + |V^1_C(t)| = |V_C| \). This is simpler than Equation 4.2 in that it considers only those nodes whose states are directly settable by the controller, neither needing to account for the other \( |V| - |V_C| \) nodes nor the stochastic process of diffusion among them. Note, that while this view of the problem objective limits the amount of uncertainty that a controller would be faced with, it also narrows the threshold of \( \theta \)-consensus requiring a more effective ability to balance local states.

### 4.4 Summary

This chapter defined the \( \theta \)-Consensus Avoidance Problem (\( \theta \)-CAP) as a novel and complete testbed instance of the NCP. Motivation for the social relevance of this problem is also presented. The \( \theta \)-CAP requires the optimization of both the behavioural and configuration components of a control system in order to successfully optimize the balancing of the state of a social network. The next part of this thesis is dedicated to the examination of solutions to the \( \theta \)-CAP. First, in Chapter 5, the behavioural subproblem is considered exclusively. Then, in Chapter 6, the dependence of the complete problem (both behaviour and configuration) is studied.
Part II

Experimental Evaluation
Chapter 5

Behavioural Control

“Many a small thing has been made large by the right kind of advertising.”

Mark Twain, A Connecticut Yankee in King Arthur’s Court

5.1 Introduction

The $\theta$-CAP, as defined in the previous chapter, is intended to be a foundational and tunably-difficult problem within the space of complete NCP instances. This chapter presents an explorative study of the $\theta$-CAP behavioural subproblem. The experimental results presented in this chapter are thus intended, first, to highlight the utility of the $\theta$-CAP as an intelligent algorithms testbed; second, to develop an understanding of the tunable difficulty of $\theta$-CAP instances, and third, to establish high quality benchmark solutions. Section 5.2 provides the motivation and description of the set of algorithms used to implement the control system behaviour. Following this, Section 5.3, presents the experimental results. First, it is shown in Section 5.3.1 that a gradient of difficulty of the problem exists relative to changes in the parameter values of the provided instance. A comparative analysis of the considered algorithms is presented to demonstrate the problem as optimizable. This comparison first looks at a broad scope of heuristic and meta-heuristic controller behaviours in Section 5.3.2. Following this, the study is narrowed to a competition between two well-tuned evolutionary neurocontrollers in Section 5.3.3. An information theoretic landscape analysis is applied in Section 5.3.4 for the purposes of validating the observed trends in the $\theta$-CAP parameter and search spaces.
5.2 Design

In order to study the $\theta$-CAP, the following algorithms were implemented as the control system’s behavioural component: Null, Random, Locally Optimal, Anti-Majority, Artificial Neural Network, Evolutionary Neural Network, and Evolutionary Connectionist Anti-Majority. Each of these algorithms are described in turn below. The details of experimentation are presented at the end of this section in Section 5.2.8. Note, that in the behavioural experimentation of the $\theta$-CAP, all control systems were configured by selecting nodes for the controlled set uniformly at random from the social network until the exhaustion of a given budget. The configuration component subproblem is the subject of study in Chapter 6.

5.2.1 Null Controller (NC)

The Null controller is a place-holder for a control system. It does not output control signals and therefore has no effect on the simulation of the social network’s convergence. This is used to understand the behaviour of the uncontrolled network diffusion. Functionally, the null controller is equivalent to dictating the state that is already at each controlled node:

$$\sigma(c, v, t') = s(v, t)$$  \hspace{1cm} (5.1)

5.2.2 Random Controller (RC)

The Random controller determines a control signal by generating a uniform random output for each controlled node. This control system is used as a baseline of control, as an intelligent algorithm is typically expected to outperform a random one. Note, however, that random noise in the space of social opinion has the potential to be very effective at preventing consensus. Formally, the RC outputs a signal to each controlled node as follows:

$$\sigma(c, v, t') = \text{round}(Unif[0, 1])$$  \hspace{1cm} (5.2)

where $Unif[0, 1]$ is a uniform random number between 0 and 1.
5.2.3 Locally Optimal Controller (LO)

The Locally Optimal controller for the \( \theta \)-CAP outputs a balanced state at its controlled nodes without consideration of the observed state of the network. Being locally optimal, this controller should exhibit perfect control over the set of diffusionless instances (see Theorem 1). However, on the diffusive instances, the fixed behaviours of the LO controller are not expected to exhibit high performance. The LO controller is implemented by alternating the output state for each node in the configuration resulting in a balanced output, +/- 1 node if there are an odd number of nodes in the controlled set \((V_C)\). Formally,

\[
\sigma(c, v, t') = \begin{cases} 
0, & \text{if index of } v \text{ is even} \\
1, & \text{if index of } v \text{ is odd}
\end{cases}
\] (5.3)

where the index of node \( v \) is a unique sequential numerical identifier.

5.2.4 Anti-Majority Controller (AM)

The Anti-Majority controller uses a heuristic method that reads the state of all visible nodes around a given controlled node (i.e., their neighbourhood including self-loop), and sets the state of the controlled node to the least common state observed. In the case of a tie the state is set uniformly at random. Formally,

\[
\sigma(c, v, t') = \begin{cases} 
0, & |N^0(v, t)| < |N^1(v, t)| \\
1, & |N^0(v, t)| > |N^1(v, t)| \\
\text{round}(\text{Uni}[0,1]), & \text{otherwise}
\end{cases}
\] (5.4)

where \( N^k(v, t) \) is the neighbourhood of node \( v \) with state \( k \) at time \( t \).

5.2.5 Artificial Neural Network (ANN)

The ANN controller trains and utilizes an Artificial Neural Network to determine the input-output mapping required of the control system. As noted in [45], ANN-controllers, or neurocontrollers, have become pervasive in various fields of control literature. This section looks at the architecture and training required to create an appropriate neurocontroller for the \( \theta \)-CAP.
The use of gradient descent techniques for training neural networks are limited to situations in which a differentiable error gradient exists, and are prone to convergence to local optima [133]. This makes these techniques less amenable to situations for which error information is not known or costly to compute, as is the case for the $\theta$-CAP, since the optimal control signals are not known in advance. In such unsupervised training scenarios, there is no direct feedback regarding the quality of the control signals generated. Indirect feedback is available in terms of successive network state evaluations and other computed values of the network state itself. However, in order to make informed adjustments to the weights in the neural network the error information with respect to its outputs must be determined. If state evaluations are not determined in a differentiable manner, such as through a supervisory critic network [7, 8, 72], then other means must be explored.

One simple approach to obtaining this information is to perform guess-and-check hill climbing, in which small changes are made to the neural network and are made permanent if they are evaluated as performance improvements. This is a naïve approach that is likely prone to the slow performance and local optima pitfalls of hill climbing in large search spaces [103]. However, a more robust search mechanism such as Evolutionary Algorithms (EA) could make this an effective solution to the problem at hand. See Section 5.2.6 for discussion of this approach.

Alternatively, the gradient of the state evaluation function with respect to the control signals can be estimated. One method of estimating this information is to use successive evaluations to determine the change in fitness with respect to changes in controller outputs. This is the method used by Anderson [2] in his application of neurocontrol to the pole balancing problem. The gradient can be estimated as:

$$\frac{\delta U}{\delta \sigma(C, V_C)} \approx \frac{U(t + 1) - U(t)}{\sigma(C, V_C, t + 1) - \sigma(C, V_C, t)}$$

where $U(t)$ is some evaluation of the state of the controlled system at time $t$, and $\sigma(C, V_C, t)$ is the output of the neurocontroller at time $t$ (a.k.a., the control signal). In practice, the unrounded outputs of the neural network are used to evaluate the gradient, the output values are then rounded for use in the discrete domain of the control signal.

Once an error gradient is established, the weights in the neural network can be
updated using the standard BP algorithm. While this estimate may be sufficient in simple problems, it is likely to give insufficient accuracy for the fine-tuning of the controller’s precision. The RPROP algorithm of Reidmiller and Braun [100] may alleviate this issue due to its use of only the sign of the derivative in its weight adjustment rules. If the gradient can be estimated as above using successive controller state evaluations then the RPROP algorithm should give more accurate adjustments, as it ignores the imprecision of the estimated gradient, instead supplying its own weight adjustment magnitudes. This provides a simple means of training a neurocontroller in an unsupervised context.

All neural networks used in this thesis consist of three layers. For each network, the input layer was given sufficient neurons for all information received at the controlled nodes (i.e., $|S(V_C,t)|$, less overlap). The output layer contains an output neuron for each controlled node in the network (i.e., $|V_C|$). The hidden layer for the ANN consists of 10 neurons. This value was not experimentally tuned for the ANN controller, and was kept small to minimize run-times. All neurons use a logistic sigmoid activation function (see Equation 2.1).

The ANN network was trained using RPROP in batch-mode, with updates occurring after each complete simulation. Early stopping of training occurred if the fitness of the training simulation remained unchanged for 5 successive runs.

### 5.2.6 Evolutionary Neural Network (ENN)

The Evolutionary Neural Network controller uses a structure similar to that of the ANN controller to handle the state to signal mappings. However, in place of a gradient descent technique for learning, it uses an Evolutionary Algorithm to search for optimal weights in the weight space. Successful applications of ENNs to various control problems have been reported [63, 74, 86, 92], including variations of the relevant pole balancing problem [34, 51, 107, 131].

**Population:** The population used throughout experimentation consists of 100 individuals. The structure and encoding used for each individual is depicted in Figure 5.1. Representation of an ANN in the EA population as a list of real values has been demonstrated to be effective over a variety of test problems [85, 134]. The structure
of the networks used are the same as the ANN controller, with 3 layers and the same number of neurons in the input and output layers. Initially, the hidden layer for the ENN control systems were set to 10 neurons as in the ANN system. However, in order to compare to the ECAM control system (See Section 5.2.7), the number of hidden neurons for ENN was adjusted to match, specifically $|V_C|$. More neurons beyond this was found to decrease performance. The ENN results of this chapter refer to this adjusted version.

**Evaluation:** Each individual in the evolution corresponds to a single control system, that is evaluated via simulation of the $\theta$-CAP. The fitness function used is the run length achieved by the resulting control system in simulation (see Equation 4.3). Elitism is used with a single best individual. Due to variance in fitness for any given solution (see page 59), the elitism individual is tested (re-evaluated) once at the end of each generation.

**Selection:** Standard tournament selection with $tSize = 3$ is used throughout this thesis to maintain relatively high diversity despite the small population size.

**Variation:** The search operator used is the outbound variant of the *mutate-nodes* operator defined in [85], in which neurons are selected at random from the given network and all weights on axons originating from the selected neuron are given
a small random shift (Uni[-0.5,0.5]). This is the search operator used throughout the thesis unless otherwise specified. Additional mutation operators were tested but changes in results were found to not be consistently statistically significant. The mutation rate for ENN was empirically tuned to 0.1.

**Termination:** EA executions in this thesis use an upper bound of 100 generations. Early termination occurs if the population converges to a standard deviation of fitness less than 0.0001, or if the elitism individual achieves the same testing fitness for 5 consecutive generations. Once the training is complete, the final elitism individual is submitted for testing.

### 5.2.7 Evolutionary Connectionist Anti-Majority Controller (ECAM)

The Evolutionary Connectionist Anti-Majority control system is proposed in this thesis as a middle ground between the ENN control system and the AM heuristic. The ECAM algorithm was developed in response to the initial results of Section 5.3.2, Note, this excludes the ECAM algorithm from the initial comparison of heuristic controllers.

To understand the ECAM algorithm, first note that the AM heuristic can be modelled as a connectionist architecture with fixed weights as shown in Figure 5.2. This form of the AM is termed the Connectionist Anti-Majority (CAM) heuristic. The CAM control system requires no training, and computes an identical input-to-signal mapping as the standard AM control system. The number of input nodes is set to
Figure 5.3: Architecture of the ECAM algorithm. On the left are the genotype and phenotype representations corresponding to the ECAM used to control the social network on the right.

the number of unique observable nodes in the social network ($|N_V|$). The number of hidden and output nodes are both set to the number of controlled nodes ($|V_C|$). Axons between the input layer and hidden layer are then added if and only if there exists an edge in the social network between the corresponding node and controlled node. This is done so as to match the input-to-hidden axons to the observable structure of the social network. Inputs to the CAM control system are scaled and shifted to the range [-1,1] (from the VM-defined social network states as discrete values in [0,1]). Weights on axons between the input and hidden layer are fixed at 1, and the hidden layer neurons output a value in the range [-1,1] (via the hyperbolic tangent activation function, Equation 2.2). This value corresponds to which of the two states was more prevalent in the neuron’s input, encoded as a positive or negative valued output. Thus the hidden layer neurons encode an observed majority feature for each controlled node. The hidden layer connects in a 1-to-1 manner to the output layer. Weights on the hidden-to-output axons are fixed at -1 to produce an anti-majority value at each output, which are then processed to produce a value in the required [0,1] range (via the logistic sigmoid activation function, Equation 2.1). Values are rounded to produce a discrete value for the resulting control signal, with values of 0.5 (corresponding to an equal proportion of observed states) given a random signal.
value as in the original AM heuristic. This produces a mapping that is functionally equivalent to the anti-majority heuristic via a connectionist architecture.

The ECAM algorithm is implemented by fully connecting the hidden-to-output layer of the CAM network. This effectively reduces the number of axons, as compared to the ENN scheme, which require settable weights, while allowing learned communication between nodes in the neurocontroller. The ECAM structure is depicted in Figure 5.3. Axons between the input and hidden layers of the ECAM are configured as in the CAM structure, and weights are fixed at 1.0. The hidden-to-output layer is fully-connected and adjustable; however, it was found during experimental tuning that initializing the ‘horizontal’ axons (bold in Figure 5.3) to -1 as in the CAM scheme, but still allowing them to be adjusted through the EA training, improved the quality of results. All other axons are initialized randomly in the standard [-0.5,0.5) range.

The EA system used for ECAM is identical to that described above for the ENN algorithm, with the exception that the mutation rate for ECAM was empirically tuned to a higher rate of 0.4 (likely due to the bias of its initialization). Further experimental tuning of the ENN and ECAM algorithms was performed using the Ramped Evolution technique described in Appendix A.

5.2.8 Experimental parameters

Instances

The network structure is determined by the generation algorithm, the number of nodes, a degree parameter, and a random generator seed. In this chapter, two generation algorithms are considered: Nearest Neighbour (NN) and Preferential Attachment (PA) (see Section 2.5). For experimentation involving additional graph types, see Chapter 6. Problem instances are described by a parametrized abbreviation of this information describing the structure of the social network and the instance parameters $\theta_G$ and $B$. As an example, the instance label ‘mn.n100.d5.b10.t60.i0’ corresponds to an instance that uses the Nearest Neighbour graph generation algorithm (mn), with 100 nodes (n100), degree parameter 5 (d5), and an instance seed of 0 (i0). The budget is set to 10% (b10) meaning the configuration can select nodes worth up to
10% of all edges (60 edges in the case of an NN-type graph), and the $\theta_G$ threshold is set to 60% (t60) meaning a difference of greater than 60% of the nodes in the network constitutes a consensus state (e.g. $|V^1| \geq 81$). Where there is no ambiguity some of these values may be left out or assumed in a given usage. The number of nodes is kept constant at 100 for simplicity of exposition throughout this thesis. NN-type graphs use a degree parameter, $d = 5$, and thus $|E| = 600$, PA-type graphs use $d = 2$, and thus $|E| = 500$ in this chapter.

The $(\theta_G, B)$-parameter space is considered in the range $\theta_G: [0, 90] \%, B: [0-100] \%$ divided by 10% increments unless otherwise specified. Initially this full range will be explored. Problem instances with large parameter values, such as the diffusionless class of instances (see Section 4.3.6), may be considered unrealistic and are slow to simulate. Owing primarily to a limit on computation time, the focus of experimentation will be narrowed towards relevant and more difficult regions of the parameter space as research progresses.

Testing

The presented results are each derived from 30 independent runs on a given instance. Each run uses a different random seed for the controller (separate from the instance seed mentioned above), corresponding to a unique random configuration and changes in stochastic behaviour. The same random configurations are used for each of the behaviour implementations. Each run consists of an optional training phase followed by 100 test simulations. The presented results are averaged over all 30 runs, which are in turn each averaged over all 100 tests.

The quality of a control system is primarily measured as the duration with which it can maintain the social network within the bounds of $\theta_G$. This run length fitness (see Equation 4.3) is capped at $5 \cdot 10^5$ steps of the social network simulation. This bound is justified by the uncontrolled convergence run length, which is observed to be consistently less than $O(|V|^2)$ in the next section.
5.3 Results

The results of this section are organized as follows: First, the uncontrolled diffusion of states is measured in Section 5.3.1. Next, a comparative analysis of the heuristic algorithms for control system behaviour is presented in Section 5.3.2. This initial comparison between the broad range of heuristics leads to the specialized comparison between the two considered neurocontroller variants in Section 5.3.3. Following this, a landscape analysis of the neurocontroller algorithms is examined in Section 5.3.4. The analysis sections which follow each pose questions to guide the corresponding discussions.

The varying difficulty of the $\theta$-CAP is most readily depicted in the gradient surface plots of Figure 5.4. By tuning values of the $\theta_G$ and $B$ parameters, problem instances vary in difficulty, as measured by simulation run lengths, from immediate failure to reaching the simulation step limit. The closer to the $(\theta_G, B)$-space origin that the apparent ‘cliff’ resides in a given surface plot, the more capable of control the given algorithm can be considered. These plots are referred to throughout the next two sections that follow.

5.3.1 Analysis of Uncontrolled Diffusion

What are the properties of the uncontrolled $\theta$-CAP?

The Null controller surface (Figure 5.4(a)) shows changes in the problem difficulty with respect to $\theta_G$; that is, the speed of convergence of the network in the absence of a control system’s intervention. The gradual incline of the run length in conjunction with increased threshold values ($\theta_G$) depicts the rate of diffusion through increasing proportions of a fixed network. This provides a relatively stable point of comparison for the run lengths of the control heuristics that follow. The average and standard deviation run lengths over 10 different network structures per type using the NC with a 90% $\theta_G$ threshold are presented in Table 5.1. Intuitively, faster convergence in the PA-type networks corresponds to shorter average path lengths. This intuition is corroborated in Section 6.4.2.

Interestingly, across all diffusive instances (those below the main diagonal on the
Figure 5.4: Surface plots of run lengths achieved over the \((\theta_C, B)\)-parameter space. Values are scaled logarithmically for greater clarity. Instance nn.n100.d5.i0. a) Null controller (NC), b) Random controller (RC), c) Locally Optimal controller (LO), d) Anti-majority controller (AM), d) Artificial Neural Network controller (ANN), e) Evolutionary Neural Network controller (ENN)

Table 5.1: Average and standard deviation of run lengths given \(\theta_C=90\%\) using the Null controller over 10 network structures of each type.

<table>
<thead>
<tr>
<th>Type</th>
<th>Average</th>
<th>St.Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>135.74</td>
<td>43.28</td>
</tr>
<tr>
<td>PA</td>
<td>35.66</td>
<td>3.04</td>
</tr>
</tbody>
</table>
(\theta_G, B)$-parameter space, see Theorem 1), the stochastic diffusion leads to a probabilistic failure rate. This is observed for experiments using any form of controller behaviour considered in this thesis. Figure 5.5 illustrates a sampled frequency distribution of a single fully trained control system tested 10000 times under identical initial conditions. A geometric distribution of outcomes is observed across all such considered frequency distributions. For the presented example (mean testing fitness $f_\mu=6145$), a Kolmogorov-Smirnov test [81] against an exponential distribution ($\lambda = \frac{1}{f_\mu} = 0.000162732$) indicates similarity significant at $\alpha = 0.01$ ($D = 0.00668$, $D_{\text{crit}} = 0.0163$). This result is consistent among all observed test results across all behaviour implementations, including the right-censored results of runs reaching simulation step limit. The probabilistic failure rate requires additional consideration throughout this thesis, including additional testing of potential solutions in Section 5.3.3 and repeated sampling of landscape walks in Section 5.3.4.
5.3.2 Comparative Analysis of Heuristics

What baseline performance is observed with simple controller behaviours?

The second baseline of comparison is derived from the ability of a random generator to act as the control system’s behaviour component. The performance of the **Random controller** demonstrates the potential for undirected control (i.e., noise) to prevent consensus beyond the durations of the uncontrolled simulations. The surface of results (Figure 5.4(b)), demonstrates changes in the difficulty of the problem space over combinations of $\theta_G$ and $B$. For easier instances (corresponding to larger values of these two parameters), the RC provides sufficient input to the social network to keep it from $\theta$-consensus until the simulation run limit. Intuitively, the entropy of random noise is a reasonable strategy to oppose the uniformity of consensus.

A sharp increase in run length is observed as the instance parameters approach the diffusionless threshold. The budget value required to create a diffusionless instance with a given $\theta_G$ can be approximated using Equation 4.8, given that $B \approx \frac{|V_C|}{|V|}$ with the assumption of all nodes in $|V_C|$ having the average cost (degree). For example, given $\theta_G = 0.7$, $B^* \geq (1/|V| + 1 - \theta_G) = 0.31 \approx 186$ edges. These boundary budget values, $B^*$, are found near the sharp increase of the RC surface, particularly for the $\theta_G \geq 50\%$, but are seen more clearly in the LO controller surface (Figure 5.4(c)). Sample results on and around the $B^*$ boundary for a fixed $\theta_G$ are presented in Tables 5.2 and 5.3. The values in these tables present the span of roughly two squares in the surface graph.

Table 5.2: Sample results around the required budget for a given instance to become diffusionless. The instance used here is mm.n100.d5.i1, $\theta_G = 0.5$, $B^* = 0.51 = 306$ edges, $5\% = 30$ edges $\approx 5$ nodes.

<table>
<thead>
<tr>
<th></th>
<th>Budget</th>
<th>$B^*$-10%</th>
<th>$B^*$-5%</th>
<th>$B^*$</th>
<th>$B^*$+5%</th>
<th>$B^*$+10%</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC</td>
<td></td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>RC</td>
<td>2952</td>
<td>7584</td>
<td>21574</td>
<td>56019</td>
<td>128068</td>
<td></td>
</tr>
<tr>
<td>LO</td>
<td>161027</td>
<td>474118</td>
<td>500000</td>
<td>500000</td>
<td>500000</td>
<td></td>
</tr>
<tr>
<td>AM</td>
<td>478077</td>
<td>475092</td>
<td>445030</td>
<td>350479</td>
<td>90768</td>
<td></td>
</tr>
<tr>
<td>ANN</td>
<td>170225</td>
<td>382270</td>
<td>360314</td>
<td>377192</td>
<td>400428</td>
<td></td>
</tr>
<tr>
<td>ENN</td>
<td>442508</td>
<td>472207</td>
<td>483721</td>
<td>492958</td>
<td>494277</td>
<td></td>
</tr>
</tbody>
</table>
Table 5.3: Sample results around the required budget for a given instance to become diffusionless. The instance used here is pa.n100.d2.i3, $\theta_G = 0.3$, $B^* = 0.71 = 355$ edges, 5% = 25 edges $\approx$ 5 nodes.

<table>
<thead>
<tr>
<th>Budget</th>
<th>NC</th>
<th>RC</th>
<th>LO</th>
<th>AM</th>
<th>ANN</th>
<th>ENN</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B^*-10%$</td>
<td>5</td>
<td>71</td>
<td>5159</td>
<td>20</td>
<td>26997</td>
<td>442508</td>
</tr>
<tr>
<td>$B^*-5%$</td>
<td>5</td>
<td>105</td>
<td>127297</td>
<td>12</td>
<td>46995</td>
<td>472207</td>
</tr>
<tr>
<td>$B^*$</td>
<td>5</td>
<td>153</td>
<td>413938</td>
<td>8</td>
<td>219890</td>
<td>483721</td>
</tr>
<tr>
<td>$B^*+5%$</td>
<td>5</td>
<td>225</td>
<td>500000</td>
<td>6</td>
<td>275770</td>
<td>492958</td>
</tr>
<tr>
<td>$B^*+10%$</td>
<td>5</td>
<td>313</td>
<td>500000</td>
<td>5</td>
<td>383371</td>
<td>494276</td>
</tr>
</tbody>
</table>

An exponential growth is observed over this span in Table 5.2 for the RC. In Table 5.3, however, this growth is not observed given the tight $\theta_G=0.3$ constraint. A similar lack in growth of results is observed over $\theta_G$ values below this range in Figure 5.4(b). This saturation of noise seems to be a consequence of the RC’s probability of outputting imbalanced states. The \{0,1\} outputs of the RC are drawn independently from a uniform distribution, meaning the combined state of the controlled nodes is governed by a binomial distribution. This is most clearly seen in the results of instances with a full-control budget, $B = 100\%$ ($|V_C| = |V|$). Here the output distribution has a mean of 50 (meaning it outputs a given state to 50 nodes and the opposing state to the remaining $|V| - 50 = 50$ nodes) and a standard deviation of $+/−5$. Note, that $+5$ of one state is also $−5$ of the other, resulting in a network state of $+/−10$. By the central limit theorem, a consensus threshold set to one standard deviation of the output ($\theta_G = 10\% = 10$ nodes), corresponds to $1−0.682 \approx 32\%$ chance of failing with each independent control signal. The expected run length of the simulation is thus approximated by a geometric distribution, modelling the number of discrete time steps prior to the occurrence of a control signal that deviates beyond the critical range from the mean, where $\theta_G$ denotes the critical range. Given this information, the expected duration of a simulation with $\theta_G = 40\%$, is approximately 31000 steps, which approximately matches the observed average of 30903. This suggests an upper bound on the effectiveness of additional noise added to the network state per time step, relative to the consensus threshold parameter. To summarize, the RC has
a limit to its effectiveness as the threshold parameter narrows. Additionally, as the budget parameter increases, the local balance of the control system’s output becomes more relevant to the global state. Both of which indicate that a more intentional control scheme is required in the general case.

The results of the **Locally Optimal** controller (see Figure 5.4(c)), are as expected. Given a diffusionless instance, the LO controller successfully balances the state of the network throughout the simulation step limit. To that end, it was the only controller to successfully balance the b100.t0 instances to the simulation step limit over all test simulations. However, the indiscriminate nature of the control signals leads to a sharp decrease in fitness just past the diffusionless threshold. Instances on or slightly above the diffusionless threshold show signs of imperfect performance, owing to the random configuration inconsistently selecting a controlled set of sufficient size. In its lowest performances, throughout the diffusive instances, the LO controller achieves fitnesses statistically equivalent to that of the RC. This implies that given no additional knowledge, a locally optimal controller is a reasonable rule of thumb.

**Can the control system performance be improved upon via intelligent control behaviours?**

In order to demonstrate the optimizability of the θ-CAP, three (meta)heuristic controllers are evaluated in terms of their ability to extend the range of long duration control (i.e., $f(C) > 10^5$) to more constrained regions of the parameter space. These controllers are distinguished from the simple behaviours in that each uses input from the social network in the determination of their outputted control signals.

The **Anti-majority** controller achieves high quality results on instances of lower budget. Despite its lack of adaptation through any learning mechanism, it achieves some statistically significant increases in performance as compared to the LO controller. A sample t-test using a 95% confidence interval over the nn.i0 parameter space yields significant results in the majority of instances with $B < 50\%$. The ratio of significant results in this range is 31:0:9 (AM:LO:neither, nn.i0). The AM controller experiences difficulty, however, as the budget grows past 50% control of the
network. Intuitively this makes sense, as the controlled set \( (V_C) \) becomes the majority, the observed node states that the controller is opposing in its decision making, are more likely to be from its own output. Controlled cliques of initially diverse node states will all be set to the same minority state in the next time step, which then becomes the majority both within the clique, and its nearby neighbours. This tail-chasing behaviour leads to results that decline in performance over the \( B \geq 50\% \) range for all observed networks. In this range, compared to LO, the ratio becomes 1:45:14 (AM:LO:neither, nn.i0). Real world applications of automated control, however, are not likely to have such large degrees of command over the network state that they wish to influence. One final point to the credit of the AM controller is its fast runtime as seen in Table 5.4. There a comparison of average run times on the nn.n100.d5.b50.t90.i0 instance (for which all but the Null controller reached the simulation step limit on all runs) is presented. This comparison ignores costs associated with resetting the network between simulations and weight adjustments in the trained neurocontrollers, and assumes no early termination of training. Experiments involving measurements of time were all measured independently on an Intel Core i7-3632QM processor with 8GB of RAM.\(^1\). In addition to fast individual runtimes, the lack of any training requirement decreases the overall runtime of the AM substantially as compared to the learning metaheuristics that follow.

<table>
<thead>
<tr>
<th></th>
<th>Single</th>
<th>Num. Runs</th>
<th>Aggregated</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC</td>
<td>1.09ms</td>
<td>100</td>
<td>109ms</td>
</tr>
<tr>
<td>RC</td>
<td>2s853ms</td>
<td>100</td>
<td>4m45s</td>
</tr>
<tr>
<td>LO</td>
<td>1s605ms</td>
<td>100</td>
<td>2m41s</td>
</tr>
<tr>
<td>AM</td>
<td>2s513ms</td>
<td>100</td>
<td>4m11s</td>
</tr>
<tr>
<td>ANN</td>
<td>8s450ms</td>
<td>100 + 100</td>
<td>28m10s</td>
</tr>
<tr>
<td>ENN</td>
<td>8s101ms</td>
<td>100(^2) + 200</td>
<td>22h57m</td>
</tr>
</tbody>
</table>

The **ANN** controller shows improvement over the RC across the majority of the parameter space (significant with 47:1:52 ANN:RC:neither, nn.i0). When compared

\(^1\)Implementation is available online: http://influence.sourceforge.net
to the LO controller, however, ANN is commonly outperformed, with the only improvements using ANN found at the lowest budget and threshold values where LO typically does its worst (3:41:56, ANN:LO:neither, nn.i0).

The uncertainty in the state of the uncontrolled portion of the social network leading to a stochastic fitness, is likely the cause of the gradient descent algorithm’s difficulty. Susceptibility to poor local optima is a known weakness of gradient descent learning, particularly when posed with highly irregular fitness landscapes [111]. The difficulty of training a neural network for the $\theta$-CAP using gradient descent is clearly illustrated by the erratic training curve displayed in Figure 5.6. This presents the ANN controller training scores for an instance on which it achieved an average run length of $\approx 18000$ in testing. A slight improvement is seen over the course of training; however, the erratic convergence is indicative of the unsuitability of gradient descent as applied to this problem. While gradient descent proved useful in the training of pole balancing ANNs [2] - the problem motivating the use of neurocontrollers for the $\theta$-CAP - the lack of a clear, unambiguous gradient here leads to the difficulties seen in the result of Figure 5.6. Confirmation of this difficulty was the original motivation for the use of evolutionary algorithms to train the ANN weights.

The use of a neural network representation for the control system behaviour component is validated by the results of the ENN controller. Despite the inconsistency
of the ANN controller, the ENN consistently achieves comparatively high quality results over all instances considered. The training phase of the ENN provides a much smoother learning curve as depicted in Figure 5.7. It should be noted that the elitism curve shown therein contains abnormal depressions in fitness over the course of training due to the practice of testing the elite individual at the end of every generation (see Section 5.2.6). An individual with inconsistent performance that is selected as the elite individual of a generation may report poor performance during the elitism test leading to the observed erratic curve during the training phase. These inconsistent individuals are expected to be out-performed in successive generations, leading to an overall trend of improvement in the population.

The ENN controller has performance which is less than or equal to the LO controller once the diffusionless threshold is exceeded, but otherwise, ENN outperforms LO by several orders of magnitude throughout large regions of the diffusive instances. The LO controller performs optimally across the diffusionless instances, whereas the narrowest $\theta_G$ ranges of the diffusionless instances remain a difficult region for the ENN to control. The significance ratio is 37:20:43 (ENN:LO:neither) across the whole parameter space, but 37:2:6 over the space of diffusive instances alone.

On a number of instances with shorter path lengths and small budgets, the AM controller achieves results with greater average run lengths than the ENN (12:15:13 ENN:AM:neither, nn.i0.b\leq40). The ENN is also occasionally outperformed by the
AM in instances for which the majority of tests reach the simulation step limit. Overall, the ENN is better performing than the AM, given the drastic decline in performance at high budgets experienced by the AM, the full parameter space significance ratio becomes 58:15:27 (ENN:AM:neither, m.i0). The high quality results of the ENN controller, however, come at the cost of very high run times. This is primarily due to the need for 10000 simulations in the training phase (ignoring the early stopping criteria).

**Summary:** Considering the heuristic comparison results so far, each of the intelligent controllers outperformed the RC in some respect. The AM controller achieves high quality results on low budget instances with a fast run time. The ANN controller shows moderate improvements over the RC in or near the class of diffusionless instances, but fails to exhibit considerable improvement over the LO controller to justify its required training time. Finally, the ENN requires significant training time to converge to comparatively high quality results over the majority of instances versus any of the simple controllers, but remains competitive with the much faster AM heuristic at the (more challenging) low budget instances.

### 5.3.3 Evolutionary Neurocontroller Analysis

The unsupervised search and expressive power of the ENN metaheuristic leads to control systems that are less biased at the outset (as compared to a fixed heuristic controller), allowing for discovery of behaviours rather than simply enacting prescribed ones. Given the results of the previous section, the AM heuristic was shown to rival the ENN’s performance under certain conditions. The ECAM algorithm (see Section 5.2) was thus created here to offer the ability to learn to overcome the AM heuristic’s weaknesses. This section considers an in-depth comparison between the standard ENN algorithm and the novel ECAM variant, with the intent of understanding what constitutes an effective behavioural component for the $\theta$-CAP.
Figure 5.8: Average testing fitness over the \((\theta_G, B)\)-parameter space using the ENN and ECAM control systems.

**Does the performance of the ENN and ECAM algorithms differ by problem instance?**

To consider the quantitative differences in solution quality between the ENN and ECAM control systems, first observe the results of using either algorithm over the \((\theta_G, B)\)-parameter space. A visual representation of the surfaces of average test fitness are presented in Figure 5.8. The algorithms here are capable of extending the high quality results \((f(C) > 10^5)\) to instances beyond the diffusionless threshold line, indicating their learned ability to cope with limited amounts of uncontrolled influence diffusion. For example, on the network considered in Figure 5.8, if \(\theta_G = 70\%\) then the diffusionless threshold budget is \(B^* = 186 \ (\approx 30\%)\). However, the observed results indicate average control fitmesses in the \(10^5\) order of magnitude with a \(B = 10\%\), and the ECAM algorithm, in particular, reaches the simulation run limit in approximately half of the considered configurations with \(B = 20\%\).

Figure 5.9 illustrates the contours of low and high fitness results extracted from the surfaces shown in Figure 5.8. There it can be readily observed that both algorithms achieve high quality results over a majority of the parameter space. The location of this ridge is comparable between both algorithms, with ENN spreading slightly wider, particularly over instances of high budget and narrow \(\theta_G\). From Figures 5.8 and 5.9, the ENN algorithm appears to begin the lower end of the ridge with lower \(\theta_G\) values, while the ECAM algorithm appears to rise more sharply bringing both algorithms to the simulation step limit at approximately the same range of instances. By means
of example, the average rate of increase over $\theta_G$ values with $B$ fixed at 40% is 7100 ($\frac{\Delta f}{\Delta \theta_G}$) for ECAM and 6300 for ENN. The ECAM algorithm is observed to have a lower standard deviation of testing fitnesses across a majority of instances (70% of the nn.i0 surface), which could lead the ENN results to appear lower as the average testing fitness approaches the simulation step limit.

Finally, it is observed that the ECAM algorithm overcomes the difficulty in controlling large budget instances suffered by the AM control system. Compared to the AM, ECAM achieves several orders of magnitude improvement over much of the $B \geq 50\%$ instances (47:0:13, ECAM:AM:neither, nn.i0.b$\geq$50). This improvement is insufficient to exceed the performance of the LO controller over several high budget diffusive instances, but by a lesser margin than the AM.

**Is the ECAM algorithm variant a statistically significant improvement over the standard ENN?**

A more direct comparison of these two neurocontrollers is performed using the difference of their average solution quality. The results are presented as log ratios in Figure 5.10 to prevent the significantly differing magnitudes from diminishing the scale of
more difficult problem instances. A Wilcoxon Signed-Rank test paired by configuration seed with a 99% confidence interval is used to determine if the observed results differ significantly by behavioural algorithm alone. Indications of such are presented as dots along the horizontal axis in Figure 5.10.

Figure 5.10: Comparison of fitness of averaged best results of ECAM vs. ENN control systems over the \((\theta_G, B)\)-parameter space. Values presented are logarithmically scaled differences grouped by \(\theta_G\).

The most noteworthy result from Figure 5.10 is the performance of the ENN algorithm on the \(\theta_G = 20, 30\) instances, in which its average results outperform the ECAM algorithm by up to nearly two orders of magnitude. These results, while interesting, are somewhat misleading. The high quality is contributed by a few very good results averaged over 30 random configurations. This causes many of the results for which ENN appears superior to not measure as statistically significant. Conversely, instances for which the ECAM algorithm appears superior are typically significant. Of the 95 considered \(\theta\)-CAP instances with \(B \leq 50\%\) over 4 network structures, the ECAM algorithm outperforms the ENN algorithm with significance at a ratio of 39:4:52 (ECAM:ENN:neither).

Figure 5.11 presents a comparison between these algorithms over the complete \((\theta_G, B)\)-parameter space for which has a higher average testing fitness on the network nn.i0. The significance ratio when considering the full range of budget values is 17:23:60 and is nearly divided by the \(B = 50\%\) line of instances. The ECAM algorithm consistently shows statistically significant improvements versus the ENN algorithm at both the low and high ends of \(\theta_G\) values for instances with \(B < 50\%.\) However, the AM biased initialization of the ECAM algorithm clearly loses its advantage for
Figure 5.11: Comparison of superior average testing fitnesses over the $(\theta_G, B)$-parameter space.

instances with $B > 50\%$. Although the ECAM algorithm is capable of generating high quality results over large budget instances (see Figure 5.9), it does so from a disadvantaged initial condition, causing the standard ENN to show significantly better results for the same duration of training.

There is no clear advantage to either algorithm across all instances, it is thus more reasonable to discuss under which conditions the use of either algorithm is appropriate. From the above results there is a clear distinction among $\theta$-CAP instances for which either algorithm outperforms the other. Given interest in the complete parameter space one could construct a hyper-heuristic for selection of the appropriate variant based primarily on the budget parameter. However, as discussed in Section 5.2.8, the primary interest in this work is towards the more difficult instances (low budget, low consensus threshold). Over the restricted space of instances with $B < 50\%$, the ECAM algorithm appears to be superior.
What qualitative differences exist between the ENN and ECAM algorithms that could account for the observed quantitative results?

The qualitative differences between the ENN and ECAM algorithms derive from the differences of their initialization and structures. The initial state of the ECAM algorithm is similar by definition to the CAM algorithm, while the initial state of the ENN algorithm can only be described as random. Accordingly, it is observed that the population of the ECAM algorithm consistently begins with a higher average fitness than that of the ENN for low budget instances \((B \leq 50\%)\), and a lower average fitness for the high budget instances. The speed of convergence and relative final fitness, however, do not appear to be dependent on the initial conditions. These are instead determined to be a factor of the properties of the instance under consideration, as is explained below.

A visual assessment of the neurocontrollers in simulation revealed the two algorithms evolved different forms of control. It was observed that when possible, the ENN algorithm evolved to maintain fixed output signals at different regions of the network, which prevents \(\theta\)-consensus in the network through the creation of opposingly converged subnetworks. On the other hand, the Anti-Majority foundation of the ECAM algorithm leads it to a rapid flickering behaviour that, when successful, maintains balance in the social network by injecting deliberate noise-like information into its otherwise converging diffusion.

These behavioural differences between the ENN and ECAM algorithms can be measured in terms of their intra- and inter-neuron output standard deviations as presented in Figure 5.12. The intra-neuron output standard deviation is the observed standard deviation of all of the outputs of a single neuron in the output layer of the elitism neural network over the course of a simulation. This value is averaged over all output nodes, and then over 30 separate executions to produce a point in Figure 5.12. The inter-neuron output standard deviation is calculated as the standard deviation over all output layer neurons of the same neural network. As such, a control system which outputs a cleverly constructed fixed signal across the network will have a very low intra-neuron standard deviation (because each given node will have very little change), and a relatively high inter-neuron standard deviation (owing to the fact
that different states will be fixed for different neurons). Conversely, a neurocontroller which exhibits a flickering behaviour will have a large variance within each node, but all nodes will behave approximately the same (from this measure). This is precisely the observed behaviour in Figure 5.12.

The foregoing results indicate a clear distinction in evolved strategy brought about by the bias (and lack thereof) in the structure and initial conditions of the two considered evolutionary neurocontrollers. The fixed output strategy of the standard ENN is considerably more effective than the ECAM strategy when it is successful. However, even a single solution that successfully manages to stabilize the social network within the given threshold is inconsistent in its ability to do so. This inconsistency is likely a result of the reliance of the control system on the stochastic process of convergence among the uncontrolled local subsections in the network prior to reaching a steady state. This may account for the inferior performance of the ENN at low budget and low threshold values (relative to ECAM), since greater control and/or more forgiving thresholds are needed to balance under the ENN strategy. The inconsistency of the ENN near the simulation step limit may also be attributable to this fixed output
strategy, as compared to the more consistent responsive behaviour of the ECAM.

**What is the relative efficiency between the ENN and ECAM algorithms?**

In addition to its superior results, the biased structure of the ECAM algorithm requires significantly less memory. Whereas the ENN algorithm uses two fully connected layers of axons requiring storage for $O(V \cdot V_C + V_C^2)$ values per individual, the ECAM must train only a single layer requiring only $O(V_C^2)$. This smaller network structure consequently requires fewer calculations to determine a control signal from a set of inputs. This means that an ECAM control system requires less execution time to balance a network to the same simulation fitness as the standard ENN control system. A comparison of the execution times of the ENN and ECAM algorithms, over the range of $\theta_G = \{0, 10, ..., 90\}$, $B = 30\%$ problem instances, is presented in Figure 5.13. Simulation times were measured in nanoseconds from the beginning of simulation until termination, and do not include the set-up and configuration steps of the network and control systems. From the figure there is a clear distinction in the linear trend lines supporting the above claim, with results for the simulation step limit requiring up to twice as much execution time for the ENN control system as for the ECAM.

**5.3.4 Landscape Analysis**

The remaining experiments of this chapter aim to explain the results observed in the previous experiments using information theoretic measurements of the fitness landscape. First, the measures are considered according to their variance over the $(\theta_G, B)$-parameter surface to validate their use on $\theta$-CAP instances and determine the characteristic differences that are observable with known changes in difficulty. Next, the results of the landscape measures using ENN and ECAM representations for the random walk are compared with interest toward discerning whether the search space of either algorithm differs from the other, and to provide a characteristic description of their differences.

The experiments performed consisted of 30 random walks, each of 1000 steps. Thirty random configurations were used for each problem instance for a total of 900000 simulations per result. The landscape measures are calculated independently
Figure 5.13: Comparison of simulation execution time versus run length for a range of $\theta_C$ values using $B = 30\%$.

for each walk then averaged over all walks to compute the results for each instance.

**Do the information theoretic landscape measures correlate with changes in $\theta$-CAP instance difficulty?**

The landscape measurements taken over the $(\theta_C, B)$-parameter space show consistent trends corroborated over all observed networks. The characteristic curves of the three landscape measures over a range of $\theta_C$ values for a fixed budget are shown in Figure 5.14. The specific curves shown are for $B = 10\%$; however, a qualitatively equivalent curve is observed for each budget. As the budgets increase the transition phase of the curve adjusts, inline with the cliff observed in the surface plots (see Figure 5.8).

Random walks for instances with $\theta_C = 0$, as with evolutionary searches of these instances, typically have simulation run lengths of only 1. The average fitness for random walks with this value (for all considered networks/budgets) is $\approx 1.05$ with a standard deviation of $\approx 0.2$. This implies that the typical surface of such a walk consists primarily of long plateaus. As such, the initial points of all three measures
describe very little variety, and stand out as distinct from the rest of the curves.

The information content (IC) characterizes the frequency and variety of transitions in the random walk (where larger numbers refer to all transitions being equally likely and smaller numbers refer to one or few transitions being more likely). The high initial IC value for $\theta_G = 10$ indicates a reasonable balance between all 6 transition types on average over the random walks of this instance. The IC is observed to decrease in line with the instance difficulty over both threshold and budget values. The curve downwards approaches a limit of 0.4 which corresponds to the equal likelihood of two (out of six) possible transitions with probability $\approx 0.3$. That is, the decrease in IC likely corresponds to the disappearance of plateaus from the random walks as the fitness space becomes less predictable. Note, that due to the geometric distribution of testing results for any given solution, a greater variety of fitnesses with a random solution (corresponding to a more rugged landscape) counter-intuitively implies a more fit region and/or an easier-to-solve instance.

The partial information content (PIC) measures ratio of transitions in the random
walk to the total length. Here, the PIC increases as the threshold and budget parameters increase (corresponding to decreased difficulty). This supports the findings of the IC measure as it indicates increased modality of the random walks. Interestingly, the PIC for all considered instances rises towards a plateau at 0.6, indicating that at most two thirds of any walk are spent transitioning between increasing and decreasing fitness, while the remaining third are so-called ‘smooth’ edges. There is no clear evidence to suggest why this apparent limit exists.

The density basin information (DBI) measures the flat (non-transitioning) regions of the random walks. The DBI values remain relatively consistent throughout all considered instances between 0.52 and 0.54, accounting for the remaining one third as an approximately equal number of flat-increasing or flat-decreasing steps.

For all instances there is a sharp decrease in all three measurement values when the simulations of the random walks plateau at the simulation limit. However, certain budget levels show a slight increase in IC and DBI values prior to this owing to the increased variety from the presence of occasional simulation-limit plateaus in the walk.

**Is there a difference in difficulty posed by the search spaces of the ENN and ECAM algorithms comparatively?**

A comparison of the landscape results from the ENN and ECAM algorithms offers few distinguishing characteristics between the fitness space explored by the two algorithms. Although the algorithms have distinct structures, including different dimensionality of solutions and initial search space biases, landscapes using either algorithm reveal the same trends relative to changes in the \( \theta_G \) and \( B \) parameters as described above.

Table 5.5 presents sample landscape data for a cross-section of the parameter space fixed at \( B = 20\% \) for the network instance discussed in the previous section. The results are similar in nature to all other diffusive instances considered. A slight decrease in IC measure for the majority of ECAM results supports the previous observation that the ENN algorithm is less consistent in its results. The IC measure of the ECAM algorithm is lower than that of the ENN in 30 of 40 considered instances by an average amount of 0.0068. More low-valued plateaus would contribute to a
Table 5.5: Sample results comparing the landscape measures of ENN and ECAM structured random walks (values are rounded).

<table>
<thead>
<tr>
<th>$\theta_G$</th>
<th>ENN B=20%</th>
<th>ECAM B=20%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg. Fit.</td>
<td>St.Dev.Fit.</td>
</tr>
<tr>
<td>10</td>
<td>2.2</td>
<td>1.95</td>
</tr>
<tr>
<td>20</td>
<td>6.2</td>
<td>6.78</td>
</tr>
<tr>
<td>30</td>
<td>16.7</td>
<td>18.61</td>
</tr>
<tr>
<td>40</td>
<td>43.1</td>
<td>49.95</td>
</tr>
<tr>
<td>50</td>
<td>125.4</td>
<td>162.38</td>
</tr>
<tr>
<td>60</td>
<td>568.3</td>
<td>970.85</td>
</tr>
</tbody>
</table>

The similarity of landscape measures between ENN and ECAM implies that the differences between the two algorithms cannot immediately be attributed to an easier search space for one over the other. This is surprising given the evidence of significant differences in the fitness of the evolved solutions. Similar difficulty in landscape measurements of ENN fitness spaces was noted in [114], where any single solution had a non-uniform distribution of potential fitnesses. To determine the degree to
Figure 5.15: Time versus fitness of a single random walk of 1000 steps using multiple evaluations per step. a) 1 evaluation, b) 30 evaluations, c) 100 evaluations.

which the landscape measures are mislead by the geometric fitness distribution of $\theta$-CAP solutions, walks were performed using repeated evaluation of solutions.

A single evaluation of a solution will fall within approximately 2 standard deviations of the mean solution fitness (approximately 200% of the mean) with 95% confidence. By comparison, evaluating a solution 100 times provides an observed average fitness that is within approximately 20% of the true mean fitness with 95% confidence [82]. While greater accuracy is preferred, in order to obtain a sample mean within 5% of the true mean requires over 1500 evaluations per solution. An illustration of the smoothing effect of multiple evaluations over the same landscape is presented in Figure 5.15.

Figure 5.16: Comparison of landscape measures using single evaluation versus 100-evaluation random walks. (nn.j0.b10)
A single evaluation per step creates a highly rugged landscape with results rapidly oscillating from the best observed to near zero in a matter of one or two steps (mutations). However, with repeat evaluations per solution the best and worst results are tempered towards a landscape with observable features and relative continuity between fitnesses of steps. The changes in landscape measurements resulting from additional evaluations can be seen in Figure 5.16. In general, the resampled measurements are indicative of more smooth and consistent walks. The 100-evaluation walks exhibit lower IC and higher PIC in the low ranges of $\theta_G$ compared to the single evaluation walks. This indicates that the multiple evaluation walks are more sensitive to fitness variations in the low-lying plateaued regions of difficult instances. A smaller value of PIC (indicating fewer transitions in the walk), is observed for the wider threshold instances. This combined with lower IC values across all instances, highlights the decreased volatility of the sampled fitness space. This is further supported by the uniformly higher DBI, lower standard deviations of fitness, and lower average fitness changes per step.

Using this more accurate view of the true fitness landscape, small differences in the measures between the ENN and ECAM algorithms appear. Overall, the IC and PIC measures are typically larger for the ECAM algorithm while the DBI is
correspondingly smaller. An increase in PIC and corresponding decrease in DBI is exhibited by the ECAM algorithm at large threshold values. These slight changes, indicating a more erratic walk by the ECAM algorithm (despite its improved fitness), are likely attributable to the increased mutation rate used throughout. Ultimately, the similarity of landscape measurements exhibited by these two algorithms discounts the notion that the smaller search space of the ECAM algorithm provides an easier landscape to traverse, suggesting that the improved fitness results from improved starting conditions.

5.4 Summary

This chapter presented the results and analysis of the behavioural component of the $\theta$-CAP. Variations among instance parameters and controller algorithms were examined for their relative effects on the performance of the control system. Two variants of evolutionary neurocontroller were determined to have the greatest fitness across the greatest coverage of the $\theta_\ell, B$-parameter space. A fitness landscape analysis was applied to corroborate the observed results, and demonstrated the shifting difficulty
of the problem in relation to the changes in instance parameters. The information theoretic measures used therein indicated very little distinction in terms of the search-ability of the fitness space using either neurocontroller variant.

A summary of conditions for the selection of behavioural algorithm is presented in Figure 5.18. In brief, the behavioural heuristics present strengths and weaknesses relative to the instance parameters and their training requirements. The LO heuristic was both fast and optimal for diffusionless instances. It also presented an effective baseline of results (as compared to the RC) on diffusive instances. The AM heuristic, while fast and effective suffered in performance above the $B=50\%$ instances. This weakness was overcome by the ECAM algorithm at the cost of a substantial training phase. However, the ENN algorithm achieved better fitnesses than ECAM on the $B>50\%$ instances. The analysis of these algorithms in this chapter was performed using a random configuration exclusively on a limited subset of network structures. The next chapter will expand on these results using a range of heuristic configuration algorithms and additional graph types.
Chapter 6

Configuration Control

“Every living being is an engine geared to the wheelwork of the universe. Though seemingly affected only by its immediate surrounding, the sphere of external influence extends to infinite distance.”

Nikola Tesla, How Cosmic Forces Shape Our Destinies

6.1 Introduction

The previous chapter considered the study of the $\theta$-CAP behavioural subproblem independent of the configuration subproblem. This chapter explores the dependent nature of the complete $\theta$-CAP. The study of this is structured as follows. Section 6.2 discusses the motivation for the steps taken in this chapter including the relevant motivating literature for the configuration heuristics chosen. The specific configuration heuristics are detailed in Section 6.3, followed by the experimental setup used in studying them. Initial experiments considering the validity and relative quality of configuration heuristics are performed on a single network structure in Section 6.4.1. Further experimentation, in Section 6.4.2, then considers the extent to which the observed results are consistent across network type variations. Section 6.4.3 examines the predictive accuracy of potentially optimal selection properties. Following this, in Section 6.4.4, the dependency of the observed trends among configurations and network are relative to changes in the behavioural heuristic.

84
6.2 Motivation

If all individuals in a network are considered equal (or similarly if we know equally little about each), then their importance is determined by their position in the social structure. This is known as the individual’s social capital [13, 14]. The identification of an accurate measure of social capital can “improve the efficiency of society by facilitating coordinated action.”[96]. To this same end, a well-constructed control system configuration facilitates the guidance of the network through appropriate dissemination of control signals. As such, determination of a desirability measure that prescribes unequal importance to individuals is a critical part in the optimization of social network control.

Several NCPs exist in which there is a search for an optimal configuration independent of any behavioural dimension (see Section 3.3). Solutions to these problems offer insight to the configuration component and are considered in this chapter, but lacking from them is the effect of the configuration on a dependent behavioural search. The purpose of experimentation in this chapter is to elucidate the dependencies inherent in the complete $\theta$-CAP; namely, the interdependence of the behavioural and configuration components, and the dependence of the configuration on the network structure.

6.2.1 Desirability Measures

The desirability of a node indicates the node’s value for selection, for instance, to be added to a controller’s configuration. The optimal desirability measures are not known for the $\theta$-CAP; however, this thesis studies a number of related measures from the relevant literature.

From Influence Maximization

For the Influence Maximization Problem (IMP) [25, 57], a set of controlled nodes is required which maximizes the spread of a given node-state throughout the network. In [57, 58], extensive simulations are performed to estimate the marginal gain in
influence diffusion across the network relative to selecting each node. A greedy hill-climbing algorithm is then employed to approximate the optimal configuration to within a constant factor \( (1 - 1/e) \approx 63\% \). In [69], to handle the case of non-unit costs per node, the marginal-gain desirability measure is extended to greedily select the node that maximizes the gain/cost ratio. Both versions of this measure, however, require the marginal-gain to be known and assumed constant. In [27] it is shown that the optimal desirability measure for the IMP using the VM is equal to the degree of a node, under the condition that all nodes have equal cost, providing a very simple selection mechanism. In [37] the performance of Genetic Algorithms, Differential Evolution, and Particle Swarm Optimization are considered as methods for optimizing the configuration for the IMP, without the need for an explicit desirability measure.

**From Targeted Immunization**

In the Immunization Problem as in the IMP, a greedy degree-based selection heuristic is employed in [90] to great effect. An alternative strategy known as acquaintance immunization was proposed in [19], in which a node is selected uniformly at random from the network, and then one of its neighbouring nodes is selected for the immunized subset. This is an approximation to a greedy degree heuristic that requires no prior knowledge of the graph.

**From Structural Controllability**

In relation to the structural controllability problem [75], it shown in [76] that the minimum number of required control nodes is 1 if a perfect matching exists on the graph. Otherwise, the minimum number and location of the control sites can be determined efficiently by computing a directed maximum matching on the graph, and then applying control to any unmatched nodes. A heuristic for determining a directed maximum matching is defined in [16], and is reproduced here in Algorithm 3. While the proofs regarding the completeness of structural controllability are specific to the linear time-invariant dynamics considered in the respective papers, this method may still offer a distributed selection mechanism that is practical for the \( \theta \)-CAP.
Algorithm 3 Pseudo-code of the heuristic approach to a directed maximum matching, adapted from [16].

**Require:** \( G = (V, E) \) is a directed graph

1: **procedure** HEURISTICDMM(\( G \))

2: \( V_C \leftarrow \{\} \)

3: while \( V \neq \{\} \) do

4: \( v^* \leftarrow \arg\max_{v \in V}(d(v)) \)

5: \( V_C \leftarrow V_C \cup \{v^*\} \)

6: repeat

7: \( v' \leftarrow \arg\max_{v \in N_{v^*}}(d(v)) \)

8: \( N_{v^*} \leftarrow v' \quad \triangleright \) remove all other edges from \( v^* \)

9: \( V \leftarrow V \setminus \{v^*\} \quad \triangleright \) remove the selected node and all inbound edges

10: \( v^* \leftarrow v' \)

11: until \( d(v^*) = 0 \)

12: **return** \( V_C \)

From Social Network Analysis

Node importance has also been studied on the basis of a node’s ability to mediate information flow between clusters [13, 14]. There, a bridge node is defined as a node which joins two otherwise disconnected or loosely connected clusters. It is argued that bridge nodes have the greatest social capital, due to their positional influence brokering new information between clusters. This property, known as brokerage, is calculated as the inverse of the network constraint measure:

\[
\text{Con}_v = \sum_{u \in V} \text{con}_{vj}, \quad v \neq u \quad (6.1)
\]

\[
\text{con}_{vu} = (p_{vu} + \sum_{q \in V} p_{vq}p_{qu})^2, \quad v \neq q \neq u \quad (6.2)
\]

\[
p_{vu} = \frac{w_{vu}}{\sum_{q \in V} w_{vq}} \quad (6.3)
\]

where \( p_{vu} \) is the proportional dependency of node \( v \) on node \( u \), and \( w_{vu} \) measures the strength of contact between two individuals. So \( \text{con}_{vu} \) is the amount of direct and indirect influence of node \( u \) on node \( v \), and \( \text{Con}_v \) measures the extent to which
information exchange with node \( v \) is concentrated in a single group.

**From Intelligent Control**

It should be emphasized here that the optimal desirability measure for the \( \theta \)-CAP is unknown. As such an intelligent search metaheuristic such as EA may be applicable to the problem. EAs have been employed to determine the minimum number and placement of actuators and/or sensors on a number of control systems. Krishnakumar et al. [66] employ a GA to simultaneously search for optimal count, placement, and controller gain using a multiple-convergence niching approach. Cook and Crossley [21] and Rogers [101] use GA to minimize the number of actuators and optimize their placement given required functionality constraints of simulated aircraft. Masoum et al. [80], Onoda and Hanawa [87], Rao et al. [97], and Yao et al. [132] each employ GA algorithms to determine optimal placement of a specified number of sensors or actuators in various of control systems ranging from power distribution networks to photovoltaic arrays. Representation encodings vary depending on the objectives of the problem. Commonly used are either a bitstring representing placement or not at each possible location, or an integer string of a fixed number of sensors/actuators representing which location each is placed.

6.3 Design

It is evident that all nodes are not equally important in network control problems. The foregoing discussion of literature related to social network control and analysis in the previous section shows this. (The fact is confirmed for the \( \theta \)-CAP in Section 6.4.1.) Motivated by the observations and results from this literature, Section 6.3.1 first defines the set of configuration heuristics employed in the experiments to come. Following this, the experimental parameters and evaluation heuristics to be used are defined in Sections 6.3.2 and 6.3.3 respectively.
Figure 6.1: Example configurations from each of the configuration heuristics. All configurations use the same fixed budget (8 here, assuming $cost(v) := d_{out}(v)$, including implicit self-loops). The evolved configuration is omitted due the unsupervised nature of its search.

6.3.1 Configuration Heuristics

The following section details the motivations and hypothesized outcomes of the eight configuration heuristics employed in the experiments of this chapter. Each selects a controlled set according to differing measures of node desirability. Examples of each can be seen in Figure 6.1. All of the heuristics select nodes in their specific ordering until the budget is exhausted or until there are no more items, unless otherwise specified.

**Random (rand.)**

The Random configuration was employed effectively for the $\theta$-CAP in the previous chapter. Random selection approximates a uniform distribution of nodes. This has a high potential for effectiveness in this problem, in that a uniform distribution constitutes an uninformed maximization of the spread of control signals prior to any
diffusion. The Random configuration considers the distribution of the controlled set to be more important than properties of the individual nodes. Due to the algorithmic process of selecting nodes randomly until the budget is exhausted, the Random configuration is expected to exhibit a slight bias towards cheaper nodes. A comparison between Random and other configurations that are based on non-uniform selection desirability thus aims to discover what, if any, node properties make them more desirable relative to the $\theta$-CAP objective.

**Degree (deg.)**

Greedy selection based on node degree was shown to be conditionally optimal for the IMP [27]. Although the equal cost condition is not met in this problem, and the goal-state differs from that of the IMP, the degree of individual nodes is considered to be a naïve measure of a node’s influence in a network [11, 46]. Since all nodes share state information with their entire neighbourhood at each timestep, the amount of information flowing through a given node is directly proportional to its degree. That is, the degree of a node is an approximation of the expected spread of that node’s state in the next timestep. Greedily selecting the most highly connected individuals can thus be considered a means of maximizing the expected rate of diffusion of the control signal. Given, however, that the goal of the $\theta$-CAP is balance of node states within the network, and that a selected node may only be given a single state at any given timestep, this method may actually increase the difficulty of control by over-representing a single state.

**Cheapest (cheap.)**

Since the cost of selecting a given node here is equal to its degree, greedily selecting the cheapest nodes is equivalent to the inverse of the previous configuration heuristic. If greedily selecting for degree is optimal for the IMP, which has a goal of consensus in the network, then the inverse heuristic has potential to be effective for the $\theta$-CAP which has a goal of preventing consensus. Ultimately, this method considers whether quantity is more important than quality for the controlled set under the circumstances of this experiment. It should be noted that controlled sets of equal
cost have equal sized neighbourhoods, so a larger controlled set means a greater geographical distribution, not a greater amount of influence. Such a distribution in this case, however, would be targeted exclusively towards nodes with low degree centrality. While the large quantity of nodes is expected to give the control system an advantage in controlling an above-average size of the network for a given budget, the lack of centrality of the selected nodes is expected to decrease the likelihood of diffusion of the control signal throughout the network.

**Acquaintance (acq.)**

The Acquaintance configuration heuristic [19] works by choosing a random node from the network and then adding to the controlled set a single random neighbour of that chosen node. This consolidates the concepts of selecting for distribution and individual desirability. As such, this heuristic is halfway between the Random and Degree heuristics in terms of process, and is expected to consistently perform between these two heuristics in terms of solution quality.

**Brokerage (broke.)**

The Brokerage configuration heuristic biases selection towards network bridges by greedily selecting nodes according to the smallest values of the network constraint measure [13, 14] (see Equation 6.1). This configuration selects nodes independently according to a measure of their location, which is different from the above measures selecting for the total set distribution. Brokerage could be a highly effective strategy in support of a divide-and-conquer control behaviour which could separate converged subpopulations by preventing distribution through controlled nodes in key network bridge locations. This heuristic is highly dependent on the existence and proportion of these network bridges, and so it is expected to exhibit large variance in quality over different network types.
Matching (match.)

Selection of nodes according to the structural controllability principle [75], selects unmatched nodes in the order determined via the directed maximum matching algorithm (see Algorithm 3). The algorithm is biased towards higher degree nodes, suggesting that the heuristic may perform similarly when compared with the Degree configuration heuristic. However, the Matching heuristic considers the distribution of the controlled set to an extent. Between selections of individual nodes, all of the nodes along a given path are removed from consideration. While this does not prevent the occurrence of neighbouring nodes in the controlled set, it does reduce its likelihood. It should be noted, that in instances for which the budget is greater than the cost of all unmatched nodes, additional nodes are selected uniformly at random until the budget is exhausted.

Farness (far.)

The simple intuition elucidated in [14] that “diffusion requires channels and an interval of time,” implies that selection of nodes based on their individual desirability is less likely to be effective than selecting to minimize the diffusion length for the set as a whole. Motivated by this observation, this chapter presents the use of a set farness measure for determining the nodes in a controlled set as dependent upon one another. The Farness configuration heuristic is described in Algorithm 4, in which line 5 uses the set farness measure to determine the next node for selection. Given knowledge of the all-pairs shortest path\(^1\) information for the graph, the process adds nodes to the initially random controlled set by greedily selecting nodes with the largest shortest path to the current set.

This is hypothesized to be the ideal heuristic for the $\theta$-CAP configuration in that it intentionally maximizes the spread of the control signal throughout the network (or equivalently minimizes the redundancy of received signals). Multiple random variations to this configuration exist per experiment due to the random selection of the first node; however, it is thought that the wide distribution of control signals throughout the social network will give consistently high performing results regardless.

\(^1\)Via Dijkstra’s [23] or the Floyd-Warshall algorithm [30].
Algorithm 4 Pseudo-code of the Farness configuration heuristic.

Require: \( \text{Dist} \) is the matrix of all-pairs shortest paths of \( V \)

Require: \( V \) is the set of nodes in the social network graph

Require: \( B \) is the maximum budget

1: procedure \text{SetFarness}(\text{Dist}, V, B)
2: \( V_C \leftarrow \{V[\text{Uni}[0, |V|]]\} \) \hspace{1cm} \triangleright \text{random first node}
3: \( V_U \leftarrow V \setminus V_C \) \hspace{1cm} \triangleright \text{copy of all uncontrolled nodes}
4: \textbf{while } \text{cost}(V_C) < B \textbf{ do}
5: \hspace{1cm} far \leftarrow (v_i : \text{argmin}_i(Dist_{ij}) \geq \text{argmin}_i(Dist_{ik}) \forall k \in V_U, \forall i \in V_C)
6: \hspace{1cm} V_C \leftarrow V_C \cup \{\text{far}\}
7: \hspace{1cm} V_U \leftarrow V_U \setminus \text{far}
8: \hspace{1cm} \text{return } V_C

of the specific nodes selected.

Evolved (evo.)

The Evolved configuration metaheuristic is a means of optimizing the configuration without requiring explicit knowledge of the optimal underlying desirability measure. To accomplish this, the ECAM control system (see Section 5.2 and Figure 5.3) is extended to simultaneously evolve the configuration component along side the previous behavioural evolution. This variation of the ECAM algorithm uses the same evaluation, selection, and termination settings as the standard version. The remainder of this section describes the changes to population, and variation settings that distinguish this modified \textit{structural} ECAM procedure.

\textbf{Population:} Figure 6.2 depicts the genotypic and corresponding phenotypic representations of a sample structural ECAM control system. A control system’s behaviour and configuration are simultaneously evolved together in the encoded genotypic representation. This encoding consists of the \( V \times V \) behavioural weight matrix as used in the standard ECAM algorithm, with an appended identity bit string to determine the configuration. This bit string consists of \( |V| \) loci, corresponding to whether a given node in the social network is controlled or not (in the example, loci
Figure 6.2: An individual in the configuration variant of an ECAM control system. The genotypic and phenotypic representations (left top and left bottom respectively) of an example individual corresponds to a single configuration and a control system behaviour together. A sample network with 2 controlled nodes (right), is influenced by the control system (left). The genotype representation (left bottom) encodes a single configuration/behaviour combination. Weight values in grey are presently inactive. The corresponding phenotype representation (left top) depicts the decoded neural network. Dashed lines correspond to inactive configuration bits, shown here for exposition but are not included in the actual phenotype.

0 and 3 constitute the controlled set).

As with the standard ECAM algorithm, the structure of the phenotypic neural network is determined in two stages. First, the hidden and output layers are each populated with one neuron per node in the controlled set. Here, this is determined by the identity bitstring. The hidden-to-output layer is fully connected with axons deriving their weights from the behavioural matrix. Second, the input layer is populated with one neuron per unique node in the neighbourhood of the controlled set (from the social network). From the example, the controlled node 0 has neighbours 0 (self-loop), 1, and 2, and so the first three neurons in the input layer correspond to these three nodes. These three neurons are connected to the node in the hidden
layer that corresponds to the controlled node 0. The controlled node 3 has neighbours 1 and 3, and so the unique node 3 is added to the input layer, and axons are connected to the neurons corresponding to nodes 1 and 3. As with the standard ECAM algorithm, the weights on the input-to-hidden layer axons are fixed at 1.

In order for the representation to accommodate the full range of potential configurations, the behavioural matrix of the EA individual contains enough information to instantiate a neural network for the control of every node in the social network; however, the configuration gene specifies exactly which subset of the weight matrix is used. Note that there is a unique value for every potentialaxon in the hidden-to-output layer of the neural network, meaning that there is a one-to-one mapping from genotype to phenotype. The phenotype representation in Figure 6.2, shows the actual and potential networks used. The configuration bits of the solution directly specify which nodes are controlled in Figure 6.2 and thus where the output of the neurocontroller is directed.

**Variation:** To preserve the validity of the evolved solutions, this thesis introduces a cost-sensitive bit-flip mutation operator. The process uses a persistent ‘gene bank’ to keep track of unused budget in the configuration bit string. Active loci are selected uniformly at random according to the mutation rate, are deactivated, and their costs are credited to the bank. Loci are then reactivated at random until the budget is as exhausted as possible. This process is described in Algorithm 5.

Following the structural mutation, a mutation of the ECAM weight matrix is performed using a modified version of the mutate-nodes operator from [85]. There, a parameterized percentage of neurons are selected uniformly at random (from the hidden layer exclusively in the case of ECAM) and all outbound axons are adjusted by a small random value in the range [-0.5,0.5). In order to prevent the large (mostly inactive) weight matrix from causing ineffective weight shifts, the mutation adjustments are limited to the active loci only. In order to accommodate the increase in expressive power per solution, the search space is increased in size from $O\left(\frac{B \cdot |V|^d \mu}{d \mu}^2\right)$ to $O\left(|V|^2\right)$, where $d_{\mu}$ is the average degree of nodes in the network.

The identity bit string representation can potentially encode all possible configurations. The use of the EA metaheuristic to search this configuration space implies a
Algorithm 5  Pseudo-code of the cost-sensitive bit-flip mutation operator for the structural ECAM algorithm.

Require: \( \text{Id} \) is a \(|V|\) identification array

Require: \( \text{credit} \leftarrow B - \sum_i \left( \text{cost}[i] \times \text{Id}[i] \right) \)

Require: \( \text{rate} \) is the mutation rate

1: procedure \text{BitFlip}(\text{Id}, \text{credit}, \text{rate})
2: for \(0 \ldots \lfloor \text{rate} \cdot |V| \rfloor\) do
3: \( r \leftarrow (\text{Uni}(0, |V|) : \text{Id}[r] = 1) \)
4: \( \text{credit} \leftarrow \text{credit} + \text{cost}[r] \)
5: \( \text{Id}[r] \leftarrow 0 \)
6: while \( \text{credit} > \min_i (\text{cost}[i]) \) do
7: \( r \leftarrow (\text{Uni}(0, |V|) : \text{Id}[r] = 0) \)
8: if \( \text{credit} > \text{cost}[r] \) then
9: \( \text{credit} \leftarrow \text{credit} - \text{cost}[r] \)
10: \( \text{Id}[r] \leftarrow 1 \)
11: return \( \text{Id} \)

reduced bias as compared to the prescribed configuration heuristics described above. The resulting configurations are expected to be reasonable approximations of the optimal configuration. The specific configurations evolved will be compared in terms of solution distance to those obtained by the heuristics above. This should give evidence towards the network properties that are being implicitly selected for by this unsupervised search procedure. The use of a cost-sensitive mutation precludes the requirement for repairing solutions during the phenotype conversion, which would typically impose some form of bias. However, the mutation does introduce a bias towards the selection of cheap nodes over time. Thus, it is expected that the evolved and cheapest configuration heuristics will have similar solution properties.

6.3.2 Experimental Parameters

In Chapter 4 a class of problem instances known as the diffusionless instances (see Section 4.3.6) were identified as those instances for which the consensus status of the
network could be dictated by the state of the controlled set alone. In such cases, the 
location of controlled nodes need not matter, only the quantity and state. As this 
chapter is interested in the exposition of the optimal configuration heuristic, focus 
is narrowed to the diffusive instances only. Furthermore, for the sake of expediting 
eperimentation, instances for which \( B > 50\% \) or the control systems considered 
thus far commonly reach the simulation step limit are not considered. In short, this 
chapter considers the more difficult problem instances identified in Chapter 5.

The majority of experimental parameters remain the same as the previous chapter. 
Default degree parameters are fixed per network type as follows: nn \( d=5 \), sw \( d=4 \), er 
\( d=5 \), pa \( d=1 \) or 2. These values were selected to maintain a relatively similar number 
of edges between network types, \(|E_{(er,pa.d2)}| = 500 \), \(|E_{(nn,sw)}| = 600 \), with the exception of pa.d1 \(|E_{pa.d1}| = 300 \) which was selected for its structural properties. Three 
seeded random structures are considered per network type. Graph instantiations are 
generated in such a way that they are guaranteed to be strongly connected.

The behavioural component of the control system utilizes the ECAM algorithm 
alongside all configurations. For each configuration, an independent ECAM system is 
trained and tested unless otherwise specified. The mutate nodes operator is used to 
train the weights of the ECAM algorithm. The mutation was tuned experimentally 
per configuration heuristic; however, the results found that statistically significant 
changes in mutation occurred in only a small minority of comparisons. No rate 
appeared uniformly superior in all cases. Therefore, for simplicity, all experimentation 
reported here is done with a mutation rate of 0.3, or 30\% of the controlled node set 
size. The Evolved configuration has an additional mutation rate of 0.01 for its identity 
bitstring.

6.3.3 Comparison Metrics

This section briefly defines a number of metrics used in the comparison of results. 
The results presented for each of these measurements are averaged over 30 random 
seeds where applicable.

The average intra-configuration path length (avg_icpl) is the size of the average path length over all pairs of nodes in a given configuration. This is a measure
of the distribution of the configurations resulting from each selection heuristic. It is
defined as follows:

$$\text{avg}_{icpl}(V_{Ch}) = \frac{\sum_{v,u \in V_{Ch}} (pl(v, u)|v \neq u)}{(|V_{Ch}|^2 - |V_{Ch}|)}$$  (6.4)

where $V_{Ch}$ is the set of nodes selected for the control system configuration via heuristic $h$, and $pl(v, u)$ is the length of the shortest path between nodes $v$ and $u$.

The decay centrality [46] measures the importance of a node relative to its
distance to all other nodes in the network (see Section 2.4). This value is averaged
over the entire configuration as follows:

$$\text{avg}_{decay}(V_{Ch}) = \frac{\sum_{v \in V_{Ch}} \sum_{u \in V} (\xi pl(v, u))}{|V_{Ch}|}$$  (6.5)

where $0 < \xi < 1$ is a decay parameter set here to 0.5.

To compare the similarity between two configurations the rectilinear distance
(a.k.a. manhattan distance) is used as follows:

$$\text{diff}(V_{Ch_1}, V_{Ch_2}) = \sum_{i=0}^{[V]} |\text{Id}_{h_1,i} - \text{Id}_{h_2,i}|$$  (6.6)

where $\text{Id}_h \in \{0, 1\}^*$ is the identity vector in which $\text{Id}_{h,i} = 1$ iff ($v_i \in V_{Ch}$) and $\text{Id}_{h,i} = 0$
otherwise.

Lastly, the relative quality of control systems over ranges of instances is ascertained
using a rank sum analysis. For each instance under consideration, the average testing
fitness for a given control system is ranked among its peers, with the worst receiving
a rank of 0 and the best receiving a rank of $n - 1$, where $n$ is the number of items
being compared. Ranks for a given algorithm are then summed over all considered
instances (e.g., all $(\theta_G, B)$ combinations) to determine the rank sum.

### 6.4 Results

The results of this section are divided into a progression of questions that catego-
rize and narrate the exploration of the dependent (behaviour and configuration) $\theta$-
CAP problem space. Four main sections study a comparison of various configuration
heuristics, the dependency of configurations on network structure, the attribution of
configuration properties to successful control, and the dependency of behaviour and configuration respectively.

6.4.1 Comparative Analysis

Are some configurations objectively better than others?

Prior to ascertaining the relative ranks of specific configuration styles, it needs to be determined that there is, in fact, a potential ranking of configurations. A brief experiment was performed to validate the intuitive assumption that some configurations are definitively better suited than others for solving specific instances of the $\theta$-CAP. To observe this, 10 independent Random configurations were compared, over 30 behaviourally Random simulation test sets of 100 simulations per set.

The best and worst performing configurations show statistically significant (t-test with $\alpha = 0.05$) differences in their overall performance on several given instances (nn.i0.b10.t60: $p = 6.995 \times 10^{-5}$, nn.i0.b10.t50: $p = 1.174 \times 10^{-5}$, nn.i0.b10.t40: $p = 0.025$). However, it cannot be said that one configuration is innately better for all problem instances (even of the same network structure). The same two configurations for a different instance (nn.i0.b10.t20) shows the best is outperformed by the worst, but not by a significant margin ($p = 0.336$). The remaining experiments in this section are thus not intended to discover the single best configuration, but rather the desirable properties that a heuristic measure could use to select well-performing configurations with high expectation.

What is the observed ranking among the considered configuration heuristics for the $\theta$-CAP?

The first set of experiments compares the configuration heuristics over various parametrized instances of a fixed network structure (nn.n100.d5.i0). Experimentation over various network types is presented in Section 6.4.2.

The results of the first comparison in terms of the mean fitness per instance are illustrated in Figure 6.3 and described in Table 6.1. From these it is clear that the ranking order of configuration heuristics is not universal over all problem instances. Heuristics that are distinctly superior for one set of instance parameters (e.g., Farness
Figure 6.3: Comparison of configuration heuristics by parametrized instances of the mn.n100.d5.i0 network structure.

on b30.t30 may be among the worst performing for a different set (e.g., b10.t60). Despite this apparent inconsistency, however, several trends emerge.

The difficulty of the problem instance (as measured by the $B$ and $\theta_G$ parameters) impacts the distinction between heuristics. This can be most readily observed in terms of the frequency of statistically significant differences in Table 6.1. As the problem difficulty decreases (larger budget and/or threshold), the ordering becomes more consistent and the frequency of statistically significant differences of means tends to increase. A rank sum analysis yields the following ordering of heuristics over all considered instances:

far. > evo. > acq. > rand. > broke. > cheap. > deg. > match.

The ordering gives an indication that configuration based upon the distribution of the controlled set (far, acq, rand: avg. rank = 5.33) is ultimately more effective than targeted individual desirability measures (broke, cheap, deg, match: avg. rank = 1.75).

**What configuration properties account for their relative qualities?**

To address this broad question, this section considers the properties of the average configurations resulting from each heuristic in terms of their similarity to one another, average intra-configuration path length, and decay centrality. Similarity among configurations of well-performing heuristics indicates a shared property of effectiveness in their respective desirability measures. The $\text{avg}_{i_cpl}$ and decay centrality measures
Table 6.1: Sample results comparing the configuration heuristic performance. Results are ordered by average fitness, significant differences are marked with a >.

<table>
<thead>
<tr>
<th>Instance (B%, θG%)</th>
<th>Ordering</th>
</tr>
</thead>
<tbody>
<tr>
<td>(10, 20)</td>
<td>far, cheap, broke, evo, rand, acq, match, deg</td>
</tr>
<tr>
<td>(10, 30)</td>
<td>far, broke &gt; deg, rand, evo, cheap, acq &gt; match</td>
</tr>
<tr>
<td>(10, 40)</td>
<td>broke &gt; rand &gt; far &gt; acq, deg, evo &gt; cheap, match</td>
</tr>
<tr>
<td>(10, 50)</td>
<td>acq &gt; broke &gt; rand &gt; far &gt; evo &gt; deg, cheap &gt; match</td>
</tr>
<tr>
<td>(10, 60)</td>
<td>evo &gt; acq, rand &gt; cheap, broke, far &gt; match &gt; deg</td>
</tr>
<tr>
<td>(20, 20)</td>
<td>far &gt; evo, acq, rand, deg, cheap &gt; match &gt; broke</td>
</tr>
<tr>
<td>(20, 30)</td>
<td>far &gt; evo &gt; acq, rand &gt; deg &gt; cheap, match &gt; broke</td>
</tr>
<tr>
<td>(20, 40)</td>
<td>far &gt; evo &gt; acq, rand &gt; cheap &gt; deg, match &gt; broke</td>
</tr>
<tr>
<td>(30, 20)</td>
<td>far &gt; evo, acq, broke, deg &gt; rand, match, cheap</td>
</tr>
<tr>
<td>(30, 30)</td>
<td>far &gt; evo &gt; acq, deg, rand, match &gt; broke, cheap</td>
</tr>
<tr>
<td>(40, 20)</td>
<td>far &gt; evo, acq, rand, cheap, broke &gt; match &gt; deg</td>
</tr>
<tr>
<td>(40, 30)</td>
<td>far &gt; cheap &gt; evo &gt; rand, acq &gt; broke &gt; match &gt; deg</td>
</tr>
<tr>
<td>(50, 20)</td>
<td>far &gt; cheap &gt; rand, acq, evo &gt; match, broke &gt; deg</td>
</tr>
</tbody>
</table>

are applied here in response to the previous section’s indication that the distribution of a configuration correlates with its effectiveness. Figures 6.4 and 6.5 present the configuration distance (see Equation 6.6) between each pair of heuristics for the smallest and largest budgets respectively. Smaller values in these figures demonstrate the similarity of the average configurations resulting from the use of any two heuristics. Figure 6.6 presents the avg_icpl and decay centrality comparison for the various configurations. The remainder of this section examines how these measures apply to each heuristic in turn.

The clear outlier in terms of overall average rank is the set Farness heuristic, which corroborates the hypothesis that a maximally distributed controlled set improves the potential dissemination of the control signal. It should be noted that Farness was the only configuration heuristic to reach the simulation limit on the set of considered instances. The heuristic that is most similar to the Farness varies from the Evolved to the Cheapest as the budget increases.

The Acquaintance and Random configurations outperform the other simple heuristics, and achieve similar ranks to each other. The Acquaintance heuristic was
expected to obtain results between those of Random and Degree, but was observed to perform better than both. It is proposed that the reason for this is that the Acquaintance configuration heuristic is theoretically less likely to select multiple nodes from the same cluster than the greedy Degree heuristic, and more likely to select influential nodes than the Random heuristic. To confirm this, consider the comparison of set distribution and decay centrality by heuristic presented in Figure 6.6. The Acquaintance heuristic was observed to have an avg icpl that did fall between the Degree and Random heuristics on average, but was consistently close to the Random heuristic. (e.g., for $B = 40\%$, $\text{avg icpl}(V_{C_{deg}}) = 7.83$, $\text{avg icpl}(V_{C_{acq}}) = 9.08$, $\text{avg icpl}(V_{C_{rand}}) = 9.10$). The distance comparison consistently shows Acquaintance and Random as having the most similar configurations to one another over all budget values (for $B = [10, 50]\%$, $\text{diff}(V_{C_{acq}}, V_{C_{rand}}) = [6.1, 10.5]$), while having relatively large dissimilarities to the more targeted heuristics ($\text{diff}(V_{C_{acq}}, V_{C_{deg}}) = [14.8, 44.2]$). This suggests that the selection of neighbouring nodes in the Acquaintance heuristic does not skew the distribution far from uniform on the considered network.

The Matching heuristic has similar distribution, set size, and decay centrality to the Acquaintance and Random heuristics, but performs worst of all of the configurations on average. The Matching heuristic was expected to perform slightly better than the Degree due to its potential to select a distributed set of nodes while biasing selection towards those with high degree. In Table 6.1, it is observed that as the budget increases the Matching improves its relative standing among the configuration heuristics. Note, that the size of the maximum matching is the same over all budgets on a given network structure, and excess budget is used to select nodes randomly. As a consequence, the Matching moves in similarity from being near the Degree and Brokerage measures towards the Random and Acquaintance measures as the budget increases.

When fewer nodes are under control, selecting for individual importance appears to increase in effectiveness. The Degree and Brokerage configurations show higher ranking among the heuristics in the 20% and 10% budgets respectively, and the distribution based configurations are outperformed over several of the 10% budget instances. As expected, the Degree and Brokerage heuristics consistently selected
sets with the shortest avg.icpl and highest decay centrality compared to all of the other configurations. Overall the results indicate that targeted selection heuristics such as these are less effective than distribution-based selection. It should be noted, that the poor relative performance of these two heuristics may be also attributable to the smaller size of their selected sets resulting from selecting the most expensive nodes (e.g., with $B = 40\%$, $|V_{C_{deg}}|/|V| = 30\%$, $|V_{C_{broke}}|/|V| = 31\%)$.

The Cheapest heuristic stands out as inconsistent among the considered heuristics. It has properties similar to those of the Farness and Evolved heuristics, but is one of the lowest performing in terms of overall rank. The Cheapest heuristic has the lowest centrality, the largest avg.icpl, seconded by the Evolved and Farness heuristics respectively. The fact that it performs relatively poorly supports the hypothesis that primarily controlling nodes in the periphery of the social network impedes the diffusion of control signals. Thus, despite the Farness heuristic achieving the best observed results, selection for a configuration that maximizes the average path length among the controlled set is not exclusively the optimal selection property. The difference being that the Cheapest heuristic has potential to select multiple nodes in isolated regions of the network, whereas the Farness configuration deliberately spaces out its
configuration.

The Cheapest heuristic does appear to improve in the higher budget instances. This is a result of selection of nodes with below average cost allowing the control system to acquire above average sized portions of the network (e.g., with $B = 50\%$, $|V_{\text{cheap}}|/|V| = 61\%$), thus approaching the diffusionless threshold with narrower constraints than any other heuristic. If the configuration were to exceed this threshold, quantity would outweigh quality in the selection of nodes for the configuration. However, in the considered instances the diffusionless threshold is not passed, and the Farness heuristic remains the best performing.

The Evolved configuration metaheuristic is typically among the top performing, thus confirming expectation of its effectiveness. The implicit desirability function appears to select for a configuration that is similar (in terms of avg_icpl and decay centrality) to those of the Farness heuristic, and achieves an average rank that is second only to the Farness heuristic. The Evolved configurations have slightly larger size and slightly lower centrality on average, suggesting that the aforementioned bias towards cheaper nodes has affected the fitness of evolved configurations. In terms of configuration distance, the Evolved metaheuristic appears to fall halfway between
Figure 6.6: Similarity between developed configurations in terms of their decay centrality (Eq. 6.5) and avg._icpl (Eq. 6.4) measures for the instance nn.n100.d5.i0 averaged over budgets in the range $B = [10 - 50]\%$.

the Random and Cheapest solutions, showing smallest distances to them over the range of instances: $\text{diff}(V_{\text{evo}}, V_{\text{cheap}}) = [14, 30]$, $\text{diff}(V_{\text{evo}}, V_{\text{rand}}) = [14, 23]$, and a somewhat larger distance between them, $\text{diff}(V_{\text{cheap}}, V_{\text{rand}}) = [22, 51]$.

The weakness of the Evolved configurations, relative to the Farness heuristic, is potentially attributable to the increased size of the search space (configurations and behaviour simultaneously) that is explored in equal sampling time to the other configurations which only require a search of the behaviour space. Testing of the Evolved metaheuristic with a doubled population size improved the results, but still did not exceed the Farness heuristic. By contrast, however, Evolved does not require a prior knowledge of the network structure in order to determine its well-performing configurations.
Figure 6.7: Comparison of the considered random graph types in terms of their average path length and average local clustering coefficient.

**Summary:** This section has demonstrated the properties of configurations that result in relatively better fitnesses. In general configurations with lower centrality and higher distribution (avg.icpl) contribute to better performing control systems. Instances of lower budget appear to increase the relative effectiveness of targeted configuration heuristics, particularly when those heuristics select with some regard to distribution (e.g., acq. > deg.).

### 6.4.2 Network Structure Dependency

This section extends the results of the previous section in order to determine their consistency across various network structures. Prior to considering the effect of the networks on the control system configurations, however, a brief analysis of the network structures themselves and their uncontrolled behaviour in the $\theta$-CAP is presented.

**Is there a correlation between network properties and diffusion speed?**

Five network types (NN, SW, PA.d1, PA.d2, ER) were defined in the background chapter (see Section 2.5), with specific instantiations described in the design section.
Table 6.2: Average uncontrolled run length, path length, and local clustering coefficient using $\theta_G = 90\%$.

<table>
<thead>
<tr>
<th></th>
<th>ER</th>
<th>NN</th>
<th>PA.d1</th>
<th>PA.d2</th>
<th>SW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run Length</td>
<td>45.04</td>
<td>157.52</td>
<td>157.96</td>
<td>35.73</td>
<td>96.88</td>
</tr>
<tr>
<td>Path Length</td>
<td>3.00</td>
<td>6.72</td>
<td>5.54</td>
<td>3.09</td>
<td>5.12</td>
</tr>
<tr>
<td>Local Clustering</td>
<td>0.038</td>
<td>0.528</td>
<td>0</td>
<td>0.1</td>
<td>0.4</td>
</tr>
</tbody>
</table>

of this chapter (see Section 6.3.2). The network structures here are considered in terms of their average path length (over all pairs of nodes) and their average local clustering coefficient [46] in order to determine the effect that these properties have on the uncontrolled convergence of the network state. Figure 6.7 depicts their comparison along these two dimensions. It is expected that longer path lengths will correspond to longer convergence times and thus easier instances for the control system. To a lesser degree, the local clustering coefficient may correspond to the potential of locally converged opposing subgraphs, which may naturally oppose consensus. However, a high degree of clustering with low average path length, could correspond to a network of high closure resulting in rapid convergence and increased difficulty of control.

The results presented in Table 6.2 provide support for the intuition that the average path length correlates with the speed of dissemination of information, with shorter paths requiring fewer timesteps to spread. Interestingly, a higher average local clustering coefficient also appears to increase the speed of convergence. To observe this, compare the run length to consensus between the ER and PA.d2 types as well as between NN and PA.d1. In both cases the network types with lower clustering have a larger convergence time, despite similar path lengths.

**Are configurations’ relative quality consistent across network types?**

Three instantiations of each of the five random graph types were considered. For each such network structure, 13 instance parameter combinations were tested from the ranges $b[10-50]$. The results, in terms of simulation length, were compared between configurations for each instance resulting in a rank from 0 to 7, with 7 given to the configuration that performed the best for the given instance.

The total sum of configuration ranks over all instances is presented in Figure 6.8.
Figure 6.8: Global rankings of each configuration heuristic over the space of all considered instances (7 ranks x 15 networks x 13 parameter settings = 1365 max ranking).

There, it can be seen that the Farness and Evolved configurations perform best as previously observed, achieving approximately 21% and 18% of the cumulative total rank respectively. The Cheapest heuristic stands out as a distinct shift from the previous results, rising from 3rd worst to 3rd best overall (≈ 14%). The Acquaintance heuristic demonstrates poorer performance (≈ 11%) than the mid-level Random heuristic (≈ 13%). Unlike the previously observed results, the total performance of Acquaintance resides between that of Random and Degree as originally anticipated. The Matching (≈ 10%), Brokerage (≈ 7%), and Degree (≈ 7%) heuristics remain poor performing overall. An analysis of the causes of and degree to which this ranking remains consistent, is discussed below.

A more precise look at the performance of each configuration can be determined using the rank sum comparison of Figure 6.9 and Table 6.3, in which each network structure is considered independently. Each section of a single column in the figure corresponds to the proportion of the total rank obtained by each configuration for that instance. From these it can be seen that, while the Cheapest heuristic is relatively poor performing on the NN (average proportion ≈ 7%) and SW (≈ 4%)
Table 6.3: Rank sum orderings of configuration heuristics per each network structure. Rows with * indicate aggregate rank sums over all 3 example networks of each type. Note: > does not denote statistical significance as these orderings are aggregated over multiple instances per network.

<table>
<thead>
<tr>
<th>Network</th>
<th>Ordering</th>
</tr>
</thead>
<tbody>
<tr>
<td>nn.i0</td>
<td>far &gt; evo &gt; acq &gt; rand &gt; broke &gt; cheap &gt; deg &gt; match</td>
</tr>
<tr>
<td>nn.i2</td>
<td>far &gt; acq &gt; match &gt; rand &gt; evo &gt; deg &gt; broke &gt; cheap</td>
</tr>
<tr>
<td>nn.i4</td>
<td>far &gt; rand &gt; evo &gt; acq &gt; match &gt; cheap &gt; broke &gt; deg</td>
</tr>
<tr>
<td>nn.i*</td>
<td>far &gt; evo &gt; rand &gt; acq &gt; match &gt; broke &gt; cheap &gt; deg</td>
</tr>
<tr>
<td>pa.d1.i0</td>
<td>cheap &gt; evo &gt; far &gt; rand &gt; broke &gt; deg &gt; acq &gt; match</td>
</tr>
<tr>
<td>pa.d1.i1</td>
<td>cheap &gt; evo &gt; far &gt; deg &gt; rand &gt; broke &gt; acq &gt; match</td>
</tr>
<tr>
<td>pa.d1.i2</td>
<td>cheap &gt; evo &gt; far &gt; deg &gt; rand &gt; match &gt; acq</td>
</tr>
<tr>
<td>pa.d1.i*</td>
<td>cheap &gt; evo &gt; far &gt; deg &gt; rand &gt; broke &gt; acq &gt; match</td>
</tr>
<tr>
<td>pa.d2.i0</td>
<td>far &gt; cheap &gt; evo &gt; rand &gt; match &gt; acq &gt; deg &gt; broke</td>
</tr>
<tr>
<td>pa.d2.i1</td>
<td>cheap &gt; far &gt; evo &gt; rand &gt; match &gt; acq &gt; broke &gt; deg</td>
</tr>
<tr>
<td>pa.d2.i2</td>
<td>cheap &gt; far &gt; evo &gt; rand &gt; match &gt; acq &gt; broke &gt; deg</td>
</tr>
<tr>
<td>pa.d2.i*</td>
<td>cheap &gt; far &gt; evo &gt; rand &gt; match &gt; acq &gt; deg &gt; broke</td>
</tr>
<tr>
<td>sw.i0</td>
<td>far &gt; evo &gt; acq &gt; rand &gt; match &gt; deg &gt; broke &gt; cheap</td>
</tr>
<tr>
<td>sw.i1</td>
<td>far &gt; match &gt; evo &gt; acq &gt; rand &gt; cheap &gt; broke &gt; deg</td>
</tr>
<tr>
<td>sw.i2</td>
<td>far &gt; evo &gt; cheap &gt; broke &gt; deg &gt; rand &gt; acq &gt; match &gt; cheap</td>
</tr>
<tr>
<td>sw.i*</td>
<td>far &gt; evo &gt; match &gt; acq &gt; rand &gt; broke &gt; deg &gt; cheap</td>
</tr>
<tr>
<td>er.i2</td>
<td>far &gt; evo &gt; rand &gt; cheap &gt; acq &gt; match &gt; deg &gt; broke</td>
</tr>
<tr>
<td>er.i4</td>
<td>far &gt; evo &gt; rand &gt; match &gt; acq &gt; cheap &gt; deg &gt; broke</td>
</tr>
<tr>
<td>er.i8</td>
<td>evo &gt; cheap &gt; far &gt; rand &gt; acq &gt; broke &gt; deg &gt; match</td>
</tr>
<tr>
<td>er.i*</td>
<td>far &gt; evo &gt; rand &gt; cheap &gt; acq &gt; match &gt; deg &gt; broke</td>
</tr>
<tr>
<td>global</td>
<td>far &gt; evo &gt; cheap &gt; rand &gt; acq &gt; match &gt; deg &gt; broke</td>
</tr>
</tbody>
</table>

Network types, it is substantially more effective on the PA (≈ 23% for both degree parameters) and to a lesser degree the ER (≈ 13%) networks. Given that, as evidence suggests, the optimal configuration is one that maximizes the distribution throughout the network, a high local clustering coefficient, as seen in the NN and SW networks could lead the Cheapest configuration heuristic astray by excluding low degree nodes from the high degree cliques, weakening the ability of control signals to penetrate the network. The PA and ER type networks, however, distribute the cheapest nodes more evenly throughout the network giving the Cheapest heuristic’s configurations an
Figure 6.9: Comparison of the rank sum share obtained by configuration heuristics over various network structures. Each network instance corresponds to 13 instance parameter variations in the range: b[10-50],t[20-60].

implicit advantage (both quantity and distribution). The Farness and Evolved configurations perform consistently well across all considered network types. Observed deficits in performance for both of these in Figure 6.9 are only in proportion to the improved performance of the Cheapest heuristic.

The Brokerage configuration heuristic performs its best on the PA.d1 networks (average proportion $\approx 10\%$) and its worst on the PA.d2 networks ($\approx 2\%$). This is because most nodes form bridges in the PA.d1 network (removing any node with degree higher than 2 causes a disconnect in the network), while in the PA.d2 no nodes are bridges (removing any single node will not cause a disconnect). In general, Brokerage performs best on networks with large average path length, owing to an increased likelihood of bridge nodes as the network is more loosely connected. The Degree and Brokerage perform similarly across all network types indicating that high degree nodes are commonly selected by the Brokerage heuristic. This is supported by the distance measurements shown in Figures 6.4 and 6.5.
How do the instance parameters affect the configuration heuristics?

To observe how the budget and threshold parameters affect the quality of configurations, Figure 6.10 presents a similar rank sum analysis in which the results per configuration are grouped by instance parameter combination. Here, each item in the graph corresponds to the normalized sum of ranks over 15 network structures using the same instance parameters and configuration heuristic.

With the smallest, most difficult, values of the parameters (b10.t20), all configurations achieve approximately equal portions of the total rank (range:[9%, 15%]). However, as the difficulty decreases, the configuration heuristic performance becomes more distinguished (b40.t30, [3%, 25%]) as was previously observed in Section 6.4.1. The Evolved metaheuristic improves its portion of the rank as the threshold parameter increases, suggesting that the evolution performs better when the individual solutions have a greater chance of success. Given the geometric fitness distribution of the problem space, poor quality solutions are less likely to be mistaken as good (and vice versa) on instances which allow the evolving population to more easily develop high fitness (and thus more distinguished) fitnesses. The Cheapest heuristic increases its
portion of the rank as the budget increases, supporting the observation from Section 6.4.1 that Cheapest gains an advantage in terms of quantity as the budget increases. The Degree and Brokerage heuristics decrease their portion of the rank as the budget increases, likely due to a higher relative amount of clustering among selected nodes as compared to the heuristics which generate more-distributed configurations. Matching again improves its overall portion at higher budget, owing to the use of a random heuristic for excess budget. Random is, not surprisingly, relatively consistent across all parameter combinations. Farness improves its proportion of ranks as the results become more distinguished, similar to the previously observed performance in Table 6.1, suggesting the status of Farness as the overall best performing configuration is robust. This robustness is further emphasized by the global ranking of farness (see Table 6.3).

6.4.3 Optimal Property Verification

What selection features contribute to optimality?

Despite inconsistency among the performance of individual configurations, it has been shown that a control system with a well-selected configuration has an advantage (see Tables 6.1 and 6.3). This section considers the ranking of properties of configurations in terms of their consistency with the ranking of the resulting control system. The accuracy of a given property’s prediction is measured as the difference between the observed fitness ranking and the ranking of configurations according to the given property. It should be reiterated here that the final solution quality is dependent on the behaviour of the control system in addition to its configuration, so the predictive accuracy is not expected to reach 100%.

Three potential properties are proposed to be evaluated for their predictive accuracy. The first property, temporal advantage, is derived from the intuition that the Farness configuration heuristic greedily approximates a set that has the shortest path to all nodes in the network for a given budget. This provides a given control system with the shortest diffusion time for a control signal to reach all nodes in the network. Specifically, the temporal advantage \( (\text{temp}) \) is measured as the average closeness over
Table 6.4: Predictive accuracy of candidate configuration properties across network types.

<table>
<thead>
<tr>
<th></th>
<th>ER</th>
<th>NN</th>
<th>PA.d1</th>
<th>PA.d2</th>
<th>SW</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>temp.</td>
<td>78.10%</td>
<td>79.53%</td>
<td>66.02%</td>
<td>63.09%</td>
<td>87.79%</td>
<td>74.9%</td>
</tr>
<tr>
<td>(</td>
<td>V_C</td>
<td>)</td>
<td>81.44%</td>
<td>73.96%</td>
<td>78.88%</td>
<td>90.53%</td>
</tr>
<tr>
<td>avg.icpl</td>
<td>81.20%</td>
<td>74.58%</td>
<td>77.87%</td>
<td>90.44%</td>
<td>76.08%</td>
<td>80.0%</td>
</tr>
</tbody>
</table>

all nodes not in the controlled set to any node in the controlled set:

\[
temp(V_{Ch}) = \frac{\sum_{u \in V} max_{v \in V_{Ch}} (1/pl(u,v))}{|V| - |V_{Ch}|}
\] (6.7)

The second candidate property is simply the size of the controlled set (\(|V_C|\)). The third is the avg.icpl measure defined in Equation 6.4 and shown in Figure 6.6. Table 6.4 presents the results of ranking the configurations according to the three candidate properties and comparing the predicted ranks to the actual ranks determined via the simulated results of the previous sections.

It is observed that, while each of the properties are good indicators of predictive ranking, no single property gives a superior predictive ranking accuracy over all network types. The avg.icpl marginally outperforms the set-size on average, with both properties exhibiting greater accuracy on networks for which the Cheapest configuration heuristic has improved performance. Recall from Figure 6.6 that configurations selected by the Cheapest heuristic commonly have the highest avg.icpl measure. The temporal advantage property, which achieves the lowest accuracy overall, provides improved accuracy for the remaining network types. In fact it is more apt at distinguishing the relative quality of the Farness and Cheapest heuristics, while suffering in predictive accuracy among the known poorer-performing configuration heuristics. The average predictive accuracy of temporal advantage for Cheapest and Farness configurations exclusively is 76.3% and 91.1% respectively, compared to (59.5%,74.0%) for set size, and (67.8%,74.3%) for avg.icpl. These results suggest that despite the inherent dependency of the \(\theta\)-CAP component subproblems, the quality of a given configuration can be approximated (relative to another) using one or more of these properties. It is conceivable that a weighted sum of these predictors, potentially in conjunction with network information such as the average path length, could provide
a more robust prediction. This, however, is left for future study.

6.4.4 Behaviour-Configuration Dependency

To assess the reliance of the behavioural component on the underlying configuration, this section briefly examines a variety of combinations of these two components. It was shown in the previous sections that a change in the configuration component while using a fixed ECAM behaviour metaheuristic had significant impact on the performance of the resulting control system. The Farness heuristic was demonstrated to produce the overall best performing configurations, with the Cheapest heuristic demonstrating improved performance under favourable network and/or parameter conditions. The ranking of heuristics was shown to vary with changes in the instance parameters, with the distinction between heuristics diminishing as the budget was decreased. This section considers the degree to which these observed trends are consistent given alternative behavioural implementations.

Are the observed configuration trends consistent across behavioural heuristics?

The observations of this section are evaluated for potential correlation from a sample of seven instances from the massive parameter space (configurations x behaviours x $B \times \theta_G \times$ networks). Specifically, the considered networks consist of a single network instance from each of the five generators with $B = 30\%$, $\theta_G = 30\%$, as well as two additional instances from two different NN-type networks (i0 and i1) with $B = 10\%$, $\theta_G = 50\%$. A sample of results is presented in Table 6.5.

It is clear that the ranking of heuristics varies from instance to instance. This was also the case with the ECAM-based experiments of the previous sections. A rank sum comparison across all considered instances is presented in Figure 6.11. Ranked configuration orderings per behavioural implementation are presented in Table 6.6.

The Farness configuration heuristic is not uniformly superior across all behaviour implementations. It does, however, have the highest rank sum on the considered instances for the ECAM and AM heuristics. These combinations are typically among the highest observed fitnesses of all control systems across each considered instance. It
Table 6.5: Run Length results of comparing combinations of behaviour and configuration components. For all presented results $B = 30\%, \theta_G = 30\%$.

<table>
<thead>
<tr>
<th></th>
<th>acq.</th>
<th>broke.</th>
<th>cheap.</th>
<th>degree</th>
<th>far.</th>
<th>match.</th>
<th>rand.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>5.58</td>
<td>6.73</td>
<td>6.55</td>
<td>6.15</td>
<td>5.26</td>
<td>5.01</td>
<td>6.48</td>
</tr>
<tr>
<td>AM</td>
<td>591.78</td>
<td>271.64</td>
<td>109.51</td>
<td>293.91</td>
<td>11405.58</td>
<td>236.62</td>
<td>351.13</td>
</tr>
<tr>
<td>ECAM</td>
<td>2836.66</td>
<td>1253.72</td>
<td>1090.86</td>
<td>2108.22</td>
<td>203163.30</td>
<td>1277.74</td>
<td>1798.99</td>
</tr>
<tr>
<td>ENN</td>
<td>2177.70</td>
<td>45246.52</td>
<td>597.34</td>
<td>29928.39</td>
<td>116.90</td>
<td>971.79</td>
<td>9669.81</td>
</tr>
<tr>
<td>Local</td>
<td>30.58</td>
<td>29.77</td>
<td>41.10</td>
<td>23.00</td>
<td>28.00</td>
<td>29.51</td>
<td>27.84</td>
</tr>
<tr>
<td>Rand.</td>
<td>21.57</td>
<td>20.36</td>
<td>23.44</td>
<td>19.85</td>
<td>20.42</td>
<td>21.05</td>
<td>20.46</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>acq.</th>
<th>broke.</th>
<th>cheap.</th>
<th>degree</th>
<th>far.</th>
<th>match.</th>
<th>rand.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>4.65</td>
<td>6.88</td>
<td>3.45</td>
<td>7.22</td>
<td>3.35</td>
<td>3.68</td>
<td>4.47</td>
</tr>
<tr>
<td>AM</td>
<td>62.55</td>
<td>23.52</td>
<td>20027.59</td>
<td>25.66</td>
<td>43879.54</td>
<td>138.99</td>
<td>132.06</td>
</tr>
<tr>
<td>ECAM</td>
<td>84.24</td>
<td>22.43</td>
<td>58796.67</td>
<td>28.52</td>
<td>53773.67</td>
<td>254.64</td>
<td>166.81</td>
</tr>
<tr>
<td>ENN</td>
<td>11.70</td>
<td>8.54</td>
<td>323.79</td>
<td>8.55</td>
<td>44.42</td>
<td>18.83</td>
<td>16.36</td>
</tr>
<tr>
<td>Local</td>
<td>11.33</td>
<td>8.99</td>
<td>110.81</td>
<td>7.56</td>
<td>32.27</td>
<td>18.38</td>
<td>14.99</td>
</tr>
<tr>
<td>Rand.</td>
<td>10.30</td>
<td>7.42</td>
<td>36.42</td>
<td>7.80</td>
<td>19.74</td>
<td>13.72</td>
<td>12.48</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>acq.</th>
<th>broke.</th>
<th>cheap.</th>
<th>degree</th>
<th>far.</th>
<th>match.</th>
<th>rand.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>4.27</td>
<td>6.83</td>
<td>8.50</td>
<td>7.10</td>
<td>7.15</td>
<td>4.06</td>
<td>7.56</td>
</tr>
<tr>
<td>AM</td>
<td>423.55</td>
<td>45.93</td>
<td>13.78</td>
<td>27.96</td>
<td>238021.12</td>
<td>1621.14</td>
<td>313.61</td>
</tr>
<tr>
<td>ECAM</td>
<td>731.07</td>
<td>63.11</td>
<td>27.69</td>
<td>78.78</td>
<td>358521.15</td>
<td>2371.07</td>
<td>747.71</td>
</tr>
<tr>
<td>ENN</td>
<td>77.54</td>
<td>71.10</td>
<td>24.59</td>
<td>40.11</td>
<td>199.26</td>
<td>100.46</td>
<td>75.25</td>
</tr>
<tr>
<td>Local</td>
<td>33.00</td>
<td>28.99</td>
<td>22.15</td>
<td>22.85</td>
<td>46.29</td>
<td>37.64</td>
<td>32.49</td>
</tr>
<tr>
<td>Rand.</td>
<td>23.22</td>
<td>23.51</td>
<td>16.76</td>
<td>19.17</td>
<td>29.10</td>
<td>24.29</td>
<td>22.16</td>
</tr>
</tbody>
</table>

is probable that the Farness heuristic is beneficial to the anti-majority behaviours for the same reason that large budgets were shown to cause problems for these behaviours (see Section 5.3.2). Separating the controlled set reduces the likelihood of controlled node neighbourhoods overlapping, which likely reduces the occurrence of destructive tail-chasing behaviour.

The Degree configuration heuristic is typically the worst performing for each behaviour, with the peculiar exception of ANN. For this poor performing behaviour, the Degree configurations were seen to have the highest ranks over all instances, while Farness was shown to have the lowest.

For the remaining behaviours, Farness falls within the upper half of configuration ranks. The Locally optimal behaviour heuristic commonly performs best using
the Cheapest heuristic. This is reasonable owing to the unobservant nature of this behaviour maximizing the size of its locally optimized contribution to the global network state. Similarly, the Random behaviour (which approximates a locally optimal heuristic), does best with Cheapest as well.

The Cheapest configuration is observed to perform well on the preferential attachment networks for all behaviours, corroborating observations of the previous section. Similarly, in instances of low budget, in which Farness was previously observed to perform less well and Acquaintance was observed to perform better, the same is seen here over the majority of behaviours ($\text{rank}_\mu(\text{acq}, B = 10\%) = 4.6$, $\text{rank}_\mu(\text{far}, B = 10\%) = 2.7$, $\text{rank}_\mu(\text{acq}, B = 30\%) = 2$, $\text{rank}_\mu(\text{far}, B = 30\%) = 4.5$).

The ENN Brokerage control system performs well relative to other configurations on the geographically distributed Nearest Neighbour type networks. These networks of high path length and local clustering exhibit large subnetworks with clear choke point nodes, supporting the ENN’s divisive strategy observed in Section 5.3.3.
Table 6.6: Rank sum orderings of configuration heuristics per behavioural heuristic. Each behavioural row is aggregated over multiple instances of varying network structure and instance parameters.

<table>
<thead>
<tr>
<th>Behaviour</th>
<th>Ordering</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>deg &gt; broke &gt; rand &gt; acq &gt; cheap &gt; match &gt; far</td>
</tr>
<tr>
<td>AM</td>
<td>far &gt; acq &gt; rand = match &gt; cheap &gt; broke &gt; deg</td>
</tr>
<tr>
<td>ECAM</td>
<td>far &gt; acq &gt; rand &gt; match &gt; cheap &gt; deg &gt; broke</td>
</tr>
<tr>
<td>ENN</td>
<td>rand &gt; acq &gt; far &gt; match = cheap = broke &gt; deg</td>
</tr>
<tr>
<td>LO</td>
<td>cheap &gt; far = acq &gt; match &gt; rand &gt; broke &gt; deg</td>
</tr>
<tr>
<td>RC</td>
<td>cheap &gt; far &gt; match &gt; rand &gt; acq &gt; broke &gt; deg</td>
</tr>
<tr>
<td>Total</td>
<td>far &gt; cheap = acq &gt; rand &gt; match &gt; broke &gt; deg</td>
</tr>
</tbody>
</table>

6.5 Summary

This chapter studied the complete $\theta$-CAP, requiring dependent solutions to both the behavioural and configuration subproblems. Several potential configuration heuristics were proposed in Section 6.3, and subsequently compared throughout Section 6.4.

In Section 6.4.1 the proposed configuration heuristics were compared in terms of the best solution fitness found via the training of an ECAM behavioural algorithm (see Chapter 5). The results indicated that configurations which select for a distribution of nodes performed better than those which select for individually desirable nodes. The Farness configuration heuristic was demonstrated to be the most highly effective heuristic when considered across all ranges of the $\theta$-CAP parameter space. However, for more constrained instances (those with small values of $\theta_C$ and $B$), it was found that the selection of nodes with high degree, without completely disregarding distribution, led to more effective control systems.

Experiments involving variations in the underlying structure of the social network, in Section 6.4.2, indicated significant variation of the results under certain circumstances. Selection of the largest possible configuration by choosing the cheapest nodes was found to provide an effective control set for the ECAM algorithm provided that the cheapest nodes were well distributed throughout the network as in the scale free networks considered. An attempt was made in Section 6.4.3 to determine the property of a given configuration which lends itself to effective control. It was found there
that the distribution of the controlled set was the most likely considered indicator of configuration fitness. However, the temporal advantage measure (Equation 6.7) was more capable at distinguishing between the Farness and Cheapest configurations.

Finally, the dependency of the observed results across various behavioural heuristics was tested in Section 6.4.4. There it was revealed that, although the ECAM_Farness configuration was the superior combination among all those considered, the optimality of a given configuration varied with the behaviour applied to it.

The next chapter will explore the degree to which these observations can be extended to related real-valued problems in the NCP/θ-CAP domain.
Chapter 7

Real Value Voter Model

“You got folks who are looking for facts that reinforce their existing point of view, as opposed to having a common conversation.... How do we join together in a common conversation around something other than the superbowl?”

Barak Obama, 2015

7.1 Introduction

The preceding chapters presented the results of applying various behavioural and configuration heuristics to the $\theta$-CAP on a number of network structures. This chapter explores the robustness of the observed results via testing the extensibility of the proposed heuristics to real-valued variants of the $\theta$-CAP. The motivation for the chosen problems is outlined in Section 7.2. Section 7.3 introduces two similar but distinct NCPs, in the formal NCP layout structure as proposed in Chapter 3. These problems differ by their objective function exclusively, and so are presented simultaneously for brevity of exposition. Section 7.3.1 presents these objective functions. A real-valued variant of the Voter Model (see Section 4.3.3) is proposed in Section 7.3.3. Section 7.3.5 describes the conversion of previously considered heuristics to the real-valued domain, as well as a locally optimal heuristic proposed for this problem. Finally, the degree to which the previously considered heuristics apply to this new domain is studied in the results of Section 7.4.
7.2 Motivation

Group polarization is the phenomenon of social influence driving individual opinions towards decreased diversity and increased extremism [31, 108]. Racism, political parties, juries, and other such opinion-based behaviours all experience ideological amplification as a result of group deliberation [108, 119]. Forms of group polarization are even attributed a partial causal role in terrorist extremism [35, 65, 108, 119].

Anonymous deliberation and group membership within online social networks has been found to exacerbate the phenomenon of group polarization [36, 55, 118]. Indeed, even simply having the ability to seek out like-minded people causes individuals to become more certain of their views and thus more extreme [108]. This is similarly an issue in the form of seeking like-minded news outlets via the personalized opt-in structure of modern news dissemination [95]. It was proposed in [108] that the application of external administrators to information exchange might produce shifts in opinion other than intensification. However, most modern recommender systems, such as those that offer news article or group suggestions, are based on similarity and overlap metrics [48, 135]. This serves to censor diversifying information, and thus move opinions towards their cluster means, thereby intensifying group polarization [135].

This chapter studies the training and use of an automated control system that is capable of mitigating extremism in a social network. To this end the $\theta$-Extremism Avoidance Problem ($\theta$-EAP) is introduced herein as an extension to the $\theta$-CAP defined in Chapter 4. This chapter aims to understand the degree to which $\theta$-CAP results extend to similar NCPs.

7.3 Problem Definition

7.3.1 Objective Functions

Two distinct but similar problems are studied in tandem throughout this chapter: the Real-valued $\theta$-Consensus Avoidance Problem, and the $\theta$-Extremism Avoidance Problem. They differ by only a small ordering of operations in their respective objective functions. Both problems consider the balance of real-valued states as they diffuse
through the social network; however, the former considers balancing of extreme states while the latter is concerned exclusively with reducing the overall magnitude of states in the network. Both problems remain constrained by a limited budget on the size of their configuration (see Equation 4.6).

**Real-valued $\theta$-Consensus Avoidance Problem**

The Real-valued $\theta$-Consensus Avoidance Problem ($\theta$-CAP$_{RV}$) is defined here as a straight forward extension of the discrete valued $\theta$-CAP of Chapter 4. Given that states in the Real-valued Voter Model (RVM, see Section 7.3.3) range from $[-1.0, 1.0]$, the failure condition of this problem occurs only if a $\theta_G$-sized portion of the network converges:

\[
U_{\theta\text{CAP}_{RV}}(t) = \begin{cases} 
1, & \text{if } \frac{|\sum_{v \in V} s(v,t)|}{|V|} < \theta_G \\
0, & \text{otherwise}
\end{cases}
\]  

(7.1)

This utility function considers the balance of opinions in the network while still allowing balanced amounts of extreme opinions. Note, a control system capable of success relative to this utility function would not be considered a remedy to bipartisan extremism; in fact, it may be seen as a potential cause. The primary purpose of this utility function is to explore the effect of the RVM while maintaining a similar objective as the previously explored $\theta$-CAP.

**$\theta$-Extremism Avoidance Problem**

The second utility function takes into consideration the total extremism of the network. This defines a similar, but distinct, problem called the $\theta$-Extremism Avoidance Problem ($\theta$-EAP). The utility function of the $\theta$-EAP distinguishes it as having the goal of minimizing the total extremism in the network.

\[
U_{\theta\text{EAP}}(t) = \begin{cases} 
1, & \text{if } \frac{\sum_{v \in V} |s(v,t)|}{|V|} < \theta_G \\
0, & \text{otherwise}
\end{cases}
\]  

(7.2)

Note this equation differs from Equation 7.1 by the order of application of the absolute value operation, making it equal to the average magnitude of opinion (in either direction).
7.3.2 Network Structure

For brevity of exposition, this chapter considers the $\theta$-CAP$_{RV}$ and $\theta$-EAP on the nn.i.o network structure unless otherwise specified, as in Chapter 5.

7.3.3 Real-valued Voter Model

In order to extend the applicability of the standard Voter Model (VM), this chapter defines the Real-valued Voter Model (RVM). The RVM allows for intermediary state values (opinions) between the two extreme values, $\mathbb{D}_{v_{RV,M}} = [-1.0, 1.0]$. The diffusion process occurs in a manner similar to that of the standard VM. In both models, a given node chooses a random node from its neighbourhood with uniform probability. In the RVM, instead of completely adopting that opinion, the state of the node shifts some small amount in the direction of the selected neighbour’s state. The size of the shift, known as the step size, is an experimental parameter of the RVM in Section 7.4. It should be noted that, although the individual opinions shift by a fixed global step size, the state values are not limited to these discretized step values. Initial state values are uniformly random in range $[-1.0, 1.0]$, and the control system may deliver signals of any value within that range.

In relation to the definition of the standard VM (see Equations 4.4 and 4.5), the RVM is defined as follows:

$$Sh(v) := s(v, t)$$ (7.3)

$$L(v, I) := s(v, t) + (\text{step} \times \text{sign})$$ (7.4)

where the value of $s(v, t)$ is bounded to the range of $\mathbb{D}_{v_{RV,M}}$, step is the step size parameter, and

$$\text{sign} = \begin{cases} 
-1 \text{ with probability } \frac{|\{s(u,t)<s(v,t)|(w,s(u,t),u)\in I(v,t)\}|}{|I(v,t)|} \\
1 \text{ with probability } \frac{|\{s(u,t)>s(v,t)|(w,s(u,t),u)\in I(v,t)\}|}{|I(v,t)|} \\
0 \text{ otherwise }
\end{cases}$$ (7.5)
The defines a diffusion process in which the flow of information from one individual to another drives the other towards the state of the first. An example of a single timestep transition in the RVM is presented in Figure 7.1. Note, owing to small differences in the initial states, interactions between any two individuals $u$ and $v$ with $|s(u, t) - s(v, t)| < step$, may result in ideological intensification; that is, the state resulting from such an interaction may have a greater magnitude than either of the initial interacting states. This can be seen in the bottom right node in Figure 7.1. Thus, the total magnitude of the network state may increase if either, individuals with large initial state magnitudes diffuse their opinion prolifically, or if individuals of similar opinion cause intensification among each other. The problem of preventing such intensification is studied throughout the remainder of this chapter.

7.3.4 Control System Configuration

The study of the problems at hand is done using the far., rand., and deg. configurations defined in as Section 6.3. The aim is to determine if the relative ranking observed among these configurations in Section 6.4.1 (far. > rand. > deg.) is consistent on the problems considered here.
7.3.5 Control System Behaviour

The behavioural heuristics employed in this chapter are primarily selections from those defined in Chapter 5. The aim here is to determine the robustness of applicability and relative fitness of these heuristics as explored in Sections 5.3 and 6.4.

The two behaviour heuristics considered for robustness are ECAM and Random. A third heuristic proposed for the problem at hand is described below. Here, the ECAM and Random behaviours are adapted to match the real valued domain of the problem, $D_{RV_M}$. This was simply achieved by removing the rounding to discrete states that was used on the outputs of either algorithm. Additionally, the output ranges of both algorithms was scaled to $D_\sigma = [-1, 1]$ as in $D_{v_{RV_M}}$.

Locally optimal heuristic

A locally optimal control system behaviour is one which is optimal on the state utility function over the set of controlled nodes ($V_C$). This is in contrast to a globally optimal behaviour which employs the best possible signals towards optimizing the state of the entire network ($V$). Given the utility function of the $\theta$-EAP (Equation 7.2), to be locally optimal is to minimize $\sum_{v \in V_C} |s(v, t)|$, or equivalently to minimize $|s(v, t)|$ for each $v \in V_C$. Therefore, a locally optimal control system is one which dictates the state of its controlled nodes to minimize $|[-1, 1]|$. That is, $\sigma(C, v, t) = 0, \forall v \in V_C, \forall t$. This simple behaviour is referred to as the **Zero** heuristic throughout this chapter.

The Zero heuristic is locally optimal for the $\theta$-CAP$_{RV}$ as well. In this case, there exist many optimal network states; namely any state in which the individual node states sum to Zero. This is clearly the case among the controlled set ($V_C$) using the Zero heuristic.

7.3.6 Discussion

Locally optimal is globally optimal

In the $\theta$-EAP, the total network state is comprised of the sum of absolute values of every individual node state. Let a decrement interaction between two nodes be one which lowers the total value of this sum. Given the interaction behaviour of
the RVM (Equations 7.3 through 7.5), a single interaction between a listening node $v$ and a sharing neighbour node $u$ leads to a decrement in the network state iff $|s(u,t)| < |s(v,t)|$. Given a constant step size, the Zero heuristic maximizes the likelihood of decrement interaction within its neighbourhood ($0 \leq |s(v,t)|, \forall v \in N_{V_C}$). Furthermore, through any decrement interactions with its neighbouring nodes, the Zero heuristic is increasing the probability that those nodes have lower states than their remaining neighbours and, by extension, the probability that they will be involved in decrement interactions with them, and so on. Ultimately, because there is a unidirectional optimization (single known optimum point), and because the controlled nodes do not listen to social interactions, a controlled set that exclusively outputs the optimum state will gradually drive the network state towards the global optimum if the set has sufficient influence to outweigh the network’s natural diffusion.

**$\theta$-EAP conditionally resembles the IMP**

When the Zero heuristic is implemented as the behavioural component in the $\theta$-EAP, what remains to be solved of the problem is the configuration component. That is, the goal of the problem becomes maximizing the influence of the control system so as to spread the fixed optimal output. This mirrors the definition of the Influence Maximization Problem (IMP) [57, 58] in which a predefined single-state behaviour is assumed of the control system and the search objective is for an optimal configuration. Given this, consideration is paid to the effectiveness of the Zero_Degree control system on the $\theta$-EAP.

The $\theta$-EAP under this restatement still remains distinct from the standard IMP in a number of ways. Firstly, the $\theta$-EAP is defined in a real-valued problem space whereas the IMP has been studied exclusively as discrete valued (typically $\{0,1\}$). Secondly, the standard IMP is typically considered on progressive diffusion models, meaning that once a state becomes active (changes state a single time) it cannot change back. The $\theta$-EAP is non-progressive as the RVM allows the repeated exchange of opinions between nodes. Finally, owing in part to the progressive definition, the stopping criteria of the IMP is considered to be the time in which diffusion stops occurring. The $\theta$-EAP retains its usage of the $\theta_G$ threshold parameter as the failure
condition.

7.4 Experimental Results

The experimentation of this section is broken down into 3 parts. First, the uncontrolled convergence behaviour of the RVM is explored in Section 7.4.1. Next, Section 7.4.2 examines the ability of a control system to adapt signals to the utility function of the $\theta$-CAP$_{RV}$. Following this, the solutions will be extended to mitigating bipartisan extremism by solving instances of the $\theta$-EAP in Section 7.4.3.

7.4.1 Uncontrolled RVM Behaviour

The study of the diffusion model is presented here in terms of its behaviour when not under the influence of an external control system. First, the effect of the step size parameter on the behaviour of the RVM is examined. Next, a monte carlo simulation is employed to characterize the speed and consistency of terminal states in the diffusion process of the RVM.

How should the step size parameter be set?

The step parameter for the RVM (see Equation 7.4) was tuned via experimentation in the range 0.01 to 1.0 with 100 tests per step value, and it was found that a tradeoff occurs. The comparison is presented in Table 7.1. Step sizes at the smallest extreme (e.g., 0.01) slowed the diffusion of opinion between individuals, resulting in very slow progression of the network state. Alternatively, larger step sizes (e.g., 0.2 and up) also demonstrated relatively long average run lengths, but only when given a large enough failure threshold ($\theta_G$). With large step sizes, the network state may oscillate rapidly, often quickly exceeding threshold boundary (particularly if $\theta_G$ is small). However, to increase the observed average run length, small numbers of simulations converge to a single state within the threshold boundary, thus remaining without failure until the simulation limit.

The default step size for the RVM was decided via experimental tuning to be a shift of 0.1 per step. This represents the mid point between the two characteristics of
Table 7.1: Average run lengths of various step size parameters using $U_{\theta_{\text{CAP}}\text{RV}}$.

<table>
<thead>
<tr>
<th>$\theta_G$ (%)</th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>6751.71</td>
<td>78.48</td>
<td>6.75</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>28885.89</td>
<td>628.77</td>
<td>78.48</td>
<td>6.75</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>58028.29</td>
<td>1960.49</td>
<td>310.02</td>
<td>30.92</td>
<td>7.5</td>
<td>1.2</td>
<td>1</td>
</tr>
<tr>
<td>40</td>
<td>102711.68</td>
<td>4011.12</td>
<td>588.88</td>
<td>86.79</td>
<td>28.84</td>
<td>9.72</td>
<td>2.06</td>
</tr>
<tr>
<td>50</td>
<td>162891.87</td>
<td>7023.90</td>
<td>1376.07</td>
<td>5251.18</td>
<td>5104.95</td>
<td>43.69</td>
<td>5012.85</td>
</tr>
<tr>
<td>60</td>
<td>244764.45</td>
<td>10373.90</td>
<td>2628.63</td>
<td>60682.21</td>
<td>45582.54</td>
<td>15223.39</td>
<td>188525.69</td>
</tr>
</tbody>
</table>

the previously described tradeoff, as indicated by the shortest time to failure given a sufficient $\theta_G$ value. It should be noted that this trend in the tradeoff is consistent using either $U_{\theta_{\text{CAP}}\text{RV}}$ or $U_{\theta_{\text{EAP}}}$, as well as over various network structures. A theoretical investigation for the determination of an optimally challenging step size is left for future work.

Does the RVM converge to a failure state?

Recall from Chapter 4 that the standard VM will converge to a failure state in $O(|V|^5)$ time [27]. A comparison between the three versions of VM utility function is presented here in terms of the probability of the diffusion model progressing to a failure state (consensus). In order to expedite experimentation, a simulation that reached the time limit of $5 \times 10^5$ steps was considered to not reach the failure state. The networks considered are all of size $|V| = 100$, so the probabilities observed below can be considered the probability of reaching a failure state in $O(|V|^3)$ time. The probabilities here are generated from 100000 uncontrolled test simulations each.

The results shown in Table 7.2 demonstrate that the standard VM converges to consensus in 100% of considered instances. The failure probability of the RVM varied slightly by utility function, with $U_{\theta_{\text{CAP}}\text{RV}}$ failing less often than $U_{\theta_{\text{EAP}}}$. This is expected since both utility functions follow the same diffusion pattern, but a balanced state of high extremism is not considered a failure state in $U_{\theta_{\text{CAP}}\text{RV}}$. In the uncontrolled RVM, all observed stable non-failure states occur when all nodes in the network converge to a state value below the given threshold (e.g., $\theta_G = 90\%$, and
Table 7.2: Probabilities of reaching a failure state prior to the time limit and the average simulation duration with no control system.

<table>
<thead>
<tr>
<th>$\theta_G$</th>
<th>VM</th>
<th>RVM $U_{\theta_{CAP_{RV}}}$</th>
<th>RVM $U_{\theta_{EAP}}$</th>
<th>VM</th>
<th>RVM $U_{\theta_{CAP_{RV}}}$</th>
<th>RVM $U_{\theta_{EAP}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>80.16</td>
<td>7.09</td>
</tr>
<tr>
<td>20</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>3.8</td>
<td>301.27</td>
<td>79.77</td>
</tr>
<tr>
<td>30</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>10.21</td>
<td>670.29</td>
<td>269.26</td>
</tr>
<tr>
<td>40</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>23.84</td>
<td>1173.75</td>
<td>636.92</td>
</tr>
<tr>
<td>50</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>53.52</td>
<td>2362.00</td>
<td>1336.99</td>
</tr>
<tr>
<td>60</td>
<td>1.0</td>
<td>0.9993</td>
<td>0.9997</td>
<td>95.63</td>
<td>4493.80</td>
<td>3228.16</td>
</tr>
<tr>
<td>70</td>
<td>1.0</td>
<td>0.9786</td>
<td>0.9790</td>
<td>126.51</td>
<td>1.78x10^4</td>
<td>1.69x10^4</td>
</tr>
<tr>
<td>80</td>
<td>1.0</td>
<td>0.6824</td>
<td>0.6831</td>
<td>204.12</td>
<td>1.67x10^5</td>
<td>1.66x10^5</td>
</tr>
<tr>
<td>90</td>
<td>1.0</td>
<td>0.2767</td>
<td>0.2865</td>
<td>255.28</td>
<td>3.67x10^5</td>
<td>3.62x10^5</td>
</tr>
</tbody>
</table>

$s(v, t*) = 0.8$, $\forall v \in V$. Thus, under $U_{\theta_{EAP}}$ the diffusion of states reaches a failure state more often and sooner than under $U_{\theta_{CAP_{RV}}}$. Accordingly, the average run lengths of the uncontrolled RVM are longer under $U_{\theta_{CAP_{RV}}}$.

7.4.2 Real-valued $\theta$-Consensus Avoidance Problem

In Chapter 6 the ECAM_Farness algorithm combination was shown effective at solving a wide range of instances of the standard $\theta$-CAP (Equation 4.2). This section extends the application of this algorithm to the $\theta$-CAP$_{RV}$ (Equation 7.1). For comparison, random variants of the behaviour and configuration component, as well as the locally optimal Zero heuristic are employed. Owing to the results of the previous section the range of the consensus threshold ($\theta_G$) is limited here to $[10, 60]$. This ensures that all considered instances would have an extremely high likelihood ($\geq 99.9\%$) of converging to failure if uncontrolled, so the effect of the control system can be measured explicitly. Initial results of this section further motivate the parameter scoping of the budget ($B$) to the range $[1, 10]$. 
Do behaviour heuristics which perform well on the $\theta$-CAP perform as well on the $\theta$-CAP$_{RV}$? 

The results of the previous section indicate that the $\theta$-CAP$_{RV}$ acts as a slower progressing version of the standard $\theta$-CAP. The results here show that this simplifies the problem in terms of extending simulation run length, allowing for greater durations of controlled state over the same parameter space. The increased run lengths of the $\theta$-CAP$_{RV}$ are readily apparent, for example, on the instance with the narrowest previously considered constraints (b10.t10) average fitness $f_{\theta_{CAP}}(ECAM\_Farness)=2.284$, $f_{\theta_{CAP}_{RV}}(ECAM\_Farness)=15818.682$, significant with a threshold of $\alpha=0.01$. To this end, the RVM experiments are compared across the narrower range of budgets $[1,10]\%$.

To determine if the relative performance of the ECAM\_Farness algorithm persists on the $\theta$-CAP$_{RV}$, a comparison is performed against a behaviourally random control system. In Chapter 5, the Random heuristic was shown to be a semi-competent implementation of the behaviour component that was outperformed across the parameter space by the locally optimal and ECAM algorithms. For the $\theta$-CAP$_{RV}$, ECAM\_Farness outperforms the Random\_Farness control system with a significance ratio of 49:0:11 (ECAM:Random:neither). The ECAM\_Farness algorithm is capable of reaching the simulation limit with a 20% narrower consensus threshold ($\theta_{G}$) than required by the Random\_Farness algorithm on the $B=10\%$ instances. A comparison of fitnesses is illustrated in Figure 7.2. Similarly, the ECAM\_Farness outperforms the locally optimal Zero\_Farness with a significance ratio of 23:0:37 (ECAM:Zero:neither) across the considered set of diffusive instances. This shows that the ECAM algorithm, which was developed for the $\theta$-CAP is effectively extensible to the $\theta$-CAP$_{RV}$.

Are $\theta$-CAP configuration heuristics effective on the $\theta$-CAP$_{RV}$?

The Farness heuristic was typically found to select the best performing configurations over the majority of instances of the $\theta$-CAP in Chapter 6, with greater performance exhibited given larger budgets. To determine if this carries over to the $\theta$-CAP$_{RV}$, consider the comparisons illustrated in Figure 7.3. There it can be readily seen that control systems employing the Farness heuristic routinely outperform their Random
configuration counterparts. Considering the combined comparison between Farness and Random configurations using both Zero and Random behaviours, the significance ratio is 42:1:77 (Farness:Random:neither). Note, the Random configuration heuristic was among the middle quality heuristics considered previously. In comparisons among heuristics on the $\theta$-CAP, the Degree configuration heuristic was among the worst performing. This trend is consistent for the $\theta$-CAP$_{RV}$ as well with significance ratios 62:11:47 (Farness:Degree:neither), and 46:14:60 (Random:Degree:neither). This consistent ranking of configurations further highlights the extensibility of $\theta$-CAP solutions to the $\theta$-CAP$_{RV}$.
7.4.3 $\theta$-Extremism Avoidance Problem

This section extends the application of the ECAM_Farness algorithm to the $\theta$-Extremism Avoidance Problem ($\theta$-EAP). This section is structured similar to the previous section with additional consideration given to the comparison between the $\theta$-EAP and $\theta$-CAP<sub>RV</sub> problems.

How does the challenge of the $\theta$-EAP compare to the $\theta$-CAP<sub>RV</sub>?

When compared to the $\theta$-CAP<sub>RV</sub>, the $\theta$-EAP is intuitively a more difficult problem. Many states in which extreme opinions would cancel each other out in the $\theta$-CAP<sub>RV</sub> instead magnify the network state in the $\theta$-EAP. Thus a single optimal point and more restricted set of allowable states exist for this problem, given the same instance parameters. A comparison of the fitnesses resulting from the application of the ECAM_Farness control system to the two problems is presented in Figure 7.4. The domination of fitnesses (40:0:20, $\theta$-CAP<sub>RV</sub>:$\theta$-EAP:neither) gives a clear indication of the differing challenge posed by these two problems given the same control system algorithms. The relative change in uncontrolled run lengths observed in Table 7.2 correlate in this instance with the controlled run lengths, supporting the hypothesis that $\theta$-EAP is a more difficult optimization problem than the $\theta$-CAP<sub>RV</sub>.
Are the developed control systems applicable to the $\theta$-EAP?

To consider the extensibility of the developed control systems to the $\theta$-EAP, a comparison of behavioural fitnesses between the ECAM_Farness and Random_Farness algorithms is first presented in Figure 7.5. The clear domination of fitnesses therein (54:0:6, ECAM:Random:neither) is consistent with the expected relative performance of the ECAM algorithm over Random on the related problems considered thus far.

![Log difference comparison of fitnesses for the ECAM_Farness and Random_Farness algorithms on the $\theta$-EAP, across the \texttt{nn.ij0.b[1-10].t[10-60]} parameter space.](image)

Figure 7.5: Log difference comparison of fitnesses for the ECAM_Farness and Random_Farness algorithms on the $\theta$-EAP, across the \texttt{nn.ij0.b[1-10].t[10-60]} parameter space.

When compared to the globally optimal Zero behaviour (also using the Farness configuration), the ECAM algorithm is outperformed as expected. The significance ratio of 0:29:31 (ECAM:Zero:neither) demonstrates the degree to which the ECAM algorithm approximates the optimal solution of the Zero heuristic. This comparison is depicted in Figure 7.6. A clear pattern illustrates the growth in superiority of the Zero heuristic as the budget parameter increases given the narrowest values of $\theta_G$ ([10%,20%]), beyond which both algorithms approach the simulation step limit.

Comparisons of the configuration heuristics on the $\theta$-EAP yield no further surprises. The ordering remains consistent with the previously observed results; specifically, Farness outperforms Random (43:11:66, Farness:Random:neither), and Random outperforms Degree (47:27:46, Random:Degree:neither). Interestingly, the Zero_Degree
algorithm was outperformed by the Zero_Random algorithm, indicating that the degree heuristic remains sub-optimal under the conditions of this homeostasis type problem (36:2:22, Zero_Random:Zero_Degree:neither).

7.5 Summary

This chapter explored two real-valued extension problems of the $\theta$-CAP, with the aim of determining the robustness of solutions developed in previous chapters to changes in the problem space itself. In particular, the ECAM and Random behaviour heuristics were compared in combination with the Farness, Random, and Degree configuration heuristics, and it was determined that their relative performance remained consistent with that observed in Chapters 5 and 6. The Zero heuristic was employed as a locally optimal heuristic in order to give a high-fitness target of comparison. The top performing ECAM_Farness algorithm from experimentation on the $\theta$-CAP demonstrated superior results to the Zero heuristic on the $\theta$-CAP$_{RV}$ where it was locally optimal. Furthermore, it demonstrated competitive results on the $\theta$-EAP, on which the Zero heuristic is globally optimal. The results indicate, firstly, the feasibility of an automated control system to mitigate bipartisan extremism via recommendation, and secondly, the robustness of the algorithms developed for the $\theta$-CAP on related
NCP instances.
Chapter 8

Conclusion

“All nature is so full, that that district produces the greatest variety which is the most examined.”

Gilbert White, *The Natural History of Selbourne* (1768)

The goal of this dissertation was to determine if diffusing opinions in social networks could be deliberately controlled, and further, the conditions under which this is feasible. To this end, this thesis considered the question: Can targeted external influence protect a social network from catastrophe? A formalization of the Network Control Problem was presented and the properties of several NCP instances worthy of study were then highlighted, including the novel $\theta$-Consensus Avoidance Problem. The $\theta$-CAP was demonstrated to be an effective testbed NCP instance requiring a search for both a configuration and a behavioural mapping in the development of an optimal control system in order to protect a network from convergence to a consensus of opinion. With the intent of motivating future study in this area, this chapter summarizes the contributions of this work, the increase in understanding that results from these contributions, and insights into future research directions.

8.1 Summary of Results

The first contribution of this thesis was the formalization of the Network Control Problem, as a component-based description of the commonalities among a subset of the existing problems in the literature. This description provides an anchor for comparison, analysis, and discussion of the diverse range of problems that the NCP
description encompasses. To further support this aim, a unified diffusion model description as a pair of behaviours with prespecified application was presented alongside the NCP definition, allowing for potentially universal interoperability of agent behaviours between or within NCP instances. The practical value of these formalisms, is that new problems can be defined within this space without the need for a complete redefinition of all required components. Similarities among problems can be examined for their support of shared solutions, and differences between isolated descriptions can be narrowed to specific discrepancies.

One such deficiency among the existing NCP instances was the lack of optimization of the required behaviour component. To address this issue, this dissertation introduced the $\theta$-Consensus Avoidance Problem ($\theta$-CAP) as the first known complete NCP instance. Motivated by the well-known pole-balancing problem from the field of linear control, the $\theta$-CAP requires the balancing of states through a largely unobserved network of stochastically interacting agents. A successful solution requires solving the dependent optimization of both the subset of agents to influence, and the content of the time series of states with which to do so. This supports the development of advanced social network control systems by providing an accessible test problem on which to consider their value.

An explorative study of the parameter space of the $\theta$-CAP demonstrated the tunable difficulty of the problem. This is a highly desirable property of any valuable testbed problem such that the competence of applied solutions can be considered with greater depth than a simple pass-or-fail basis. This study revealed features of the problem space such as the diffusionless threshold, exponential growth in fitness over parameter ranges, and geometric distribution of solution fitnesses. Such features dictate the adjustment of approach taken by the practitioner given their intended study (i.e., a general purpose control system is faced with different challenges than one applied to specific ranges of parameters).

A comparative analysis of the $\theta$-CAP problem space in Chapters 5 and 6 considered the relative performance of several heuristic and metaheuristic solutions to the behavioural and configuration subproblems respectively. Solutions for the behavioural component focussed on the application of neurocontrollers. Evolutionary solutions to
the optimization of neurocontroller parameters for this problem demonstrated the
optimizable search space of the problem, and discovered variability among the strate-
gies that were learned. A comparison between the standard Evolutionary Neural
Network (ENN) and the novel Evolutionary Connectionist Anti-Majority (ECAM)
variant demonstrated the applicability of the $\theta$-CAP to the development and study
of new improved optimization techniques. ECAM, which was designed as a middle
ground between the ENN and AM algorithms, provides benefits from both. It achieves
better quality results in less time on the more difficult small budget instances. While
the standard ENN attains consistently better results for instances with large bud-
ggets, the ECAM algorithm overcomes the weakness of the Anti-Majority heuristic to
demonstrate extended durations of control for these instances as well. The standard
ENN evolved a fixed-output convergence to two or more opposed subsections of the
network of size equal enough to prevent exceeding the $\theta_G$ threshold. ECAM, on the
other hand, evolved to inject noise-like signals in order to prevent convergence among
the highly communicative agents of the social network.

A landscape analysis corroborated observed trends in the search space including
the variable difficulty associated with ranges of instance parameter values. A compa-
ration between the two evolutionary neurocontrol algorithms revealed unexpected
similarity in the nature of the fitness landscapes traversed. Slight changes in the
observed values, and increased average fitness over the majority of random walks,
suggests that both networks search on the same fitness space, with the ECAM simply
biased to a promising region thereof.

Consideration of the configuration subproblem adopted heuristic strategies from
across the varied fields of related node-selection problems. Given the ECAM be-
haviour implementation, the optimal property of configuration was to select a con-
trolled set that maximizes the initial distribution of its signal throughout the network.
The Farness configuration heuristic greedily approximates this property and achieves
the best observed performance over the set of considered instances, performing better
even than an extended evolutionary search for the same component. Exceptions to
the use of Farness occur under a number of parameter settings. Selecting for the
cheapest nodes maximizes quantity, which becomes more important when the size of
the configuration exceeds the diffusionless threshold. The use of the Farness heuristic on an instance with a small budget value can result in a configuration that is solely at the periphery of the network, resulting in poor performance. The remedy to this is using a configuration heuristic that is more uniformly distributed to include more individually influential nodes towards the core of the network. The Farness heuristic was determined to be most effective in conjunction with ECAM/Anti-Majority behaviours, but was not consistently optimal across all behaviours. The performance of the Cheapest heuristic achieves a noteworthy improvement on the networks of low local clustering, in which low degree nodes have uniform distribution throughout the network. The consistency of trends was considered across network types, behaviours, and utility functions with varying results.

The experimental analysis of solutions to the $\theta$-CAP and the related $\theta$-CAP$_{RV}$ and $\theta$-EAP, demonstrate the effectiveness and extensibility of the $\theta$-CAP as an intelligent algorithms testbed in the field of network control problems. However, the developed understandings are not just in terms of a testbed problem, but also have potential real world application. The consensus avoidance metaphor applies broadly to real world scenarios as discussed in the introduction of this dissertation. As such the observed results can inform the development of social propaganda in situations with balance or diversity maintenance goals. As an example, consider the problem of news recommendation with the objective of maintaining an inclusive discourse among users. Given that real world networks commonly exhibit scale free properties, insights from the results of this thesis suggest that minimum cost control should leverage a maximized distribution of cheap nodes, as opposed to very few highly influential nodes. Simulation of a learning algorithm could then inform of the parameters required to coordinate the shifting network state via these nodes.
8.2 Future Research

Research proceeding from the work done here should be of three main directions: 1) additional benchmarking of improved intelligent algorithms on the $\theta$-CAP, 2) greater depth and extension of $\theta$-CAP analysis, and 3) development towards more general NCP instances.

8.2.1 Additional breadth of intelligent algorithms for the $\theta$-CAP

The algorithms employed in this dissertation scratch the surface of all possible control system implementations. While effective intelligent control was demonstrated against several baseline heuristics, improved benchmarks are likely achievable through more advanced techniques. From the field of neurocontrol, recurrent neural network controllers [71, 122, 131] offer a means of determining control signals from a memorized time series of states as opposed to the single state reactionary networks developed here. Coevolution of neural network structure and axon weights [32, 116, 134] may provide the evolution a means of biasing the search space (as with ECAM above), without the bias inherent in human design. Adaptive critic networks [72, 73, 94] empower the standard neurocontroller with a sense of consequence of its own actions by evaluating partial states instead of only the final failure state. This would likely improve the robustness of the created neurocontrollers by providing a means of online training. The vast fields of stochastic control theory [9] and intelligent control [53, 79] offer many potential solutions with high degrees of relevance: Fuzzy modelling and its hybrids [47, 93, 125], Bayesian control [39], expert systems [70], and analytical forms of adaptive control such as recursive least squares [4, 109], and so on. Additional approaches from existing influence diffusion literature, such as the marginal gain analysis of configurations [57, 69], may also yield improvements in the $\theta$-CAP. As a testbed problem, the $\theta$-CAP is expected to support the development of a wide range of optimization and control algorithms.

The results observed indicate variability among solutions relative to problem parameters, suggesting that a more robust approach would be an ensemble of approaches. Configuration heuristics showed a particular sensitivity to changes in the network structure, and may benefit from the deliberate choice of heuristic based on
the specific instance under consideration. Alternatively, a linear combination of configuration heuristic desirability measures is hypothesized to discern greater prediction accuracy for a given configuration, and may also improve the measure of individual node selection desirability.

The Farness heuristic requires a priori knowledge of the network in the form of the all-pairs shortest path lengths. While calculation of this is negligible compared to the simulation-based fitness evaluation of the evolutionary algorithm, the requirement does preclude its use in situations where that information is not known or prohibitive to compute. A temporal response hillclimber is envisioned such that the Farness heuristic can be approximately constructed through simulation without any a priori knowledge of the network structure. Therein control signals sent to the network are marked, such that when they are sensed in the neighbourhood of other controlled nodes, an estimate of time/distance can be ascertained and incorporated into a dynamically adjusting control system configuration.

8.2.2 Greater depth of $\theta$-CAP analysis

The $\theta$-CAP problem space itself can be expanded to direct it towards new challenges that are potentially more relevant to real world scenarios. Indirect control, as described in Chapter 3, considers the problem of influence diffusion without the form of coercion implied in the direct control version. Instead individuals seeking to influence the network must interact with no special privileges (beyond collusion within a centralized control system) over the natural diffusion model. Going further, multiple independent distributed controllers (using, for example, multi-agent neurocontrollers [22, 106]) more closely model the problem of social network control via grass roots mobilization. These may operate in isolation or with communication depending on the objectives to be considered.

Additional depth in the diffusion models may also bring more realism. Selections from literature beyond the Voter Model (e.g. Linear Threshold, Independent Cascade [57], Infection and Rumour models [11]) offer greater depth and variety to considerations of related NCPs. Similarly, the controllability of dynamic networks should be considered, in which nodes adjust their neighbourhoods in response to the
information flowing through the network. Existing models of trust and reputation among networked agents [104, 105] can readily be incorporated into the DM formalism defined in Section 3.2.3 as part of the adjustment phase.

In relation to the problem of modelling bipartisan extremism, the real world scenario involves ideological intensification resulting directly from individual disagreement. This leaves populations susceptible to misinformation propaganda, and subsequently fuels the bipartisan extremism division [67, 88]. A diffusion model that captures this opposing ideological intensification would better serve the study and mitigation of this pressing issue.

The VM (and RVM) may be further unrealistic in that, when faced with an opposing view, the individual reconsiders their position (whereas in real life, opposition most often leads to entrenchment [108]). A diffusion model in which individuals develop a sense of social identity (via similarity measure [115], for example) associated with a particular opinion or group, could pose such interesting challenges to the control system.

8.2.3 Towards the general NCP

From an opposing point of view, using the trained controller as an automated opponent, the defined problems can be used to develop diffusion models of less gullible individuals. Discerning individuals capable of validating information [1, 12, 64], as well as those capable of adjusting their neighbourhood relative to trust and reputation measures [104, 105] could shift the focus of the problem from balancing opinions to detecting and avoiding external influences that foment disagreement to disrupt social progress.

One final problem of interest to real world situations is the inequality problem. There the diffusion model represents economic investment, and real-valued states represent quantified wealth. A similar homeostatic goal is considered as preventing run-away wealth inequality within the networked marketplace, with the automated control system tasked with balancing this process via targeted investment. Such a model could be employed to determine the expected outcomes of uncontrolled diffusion under various investment styles, and a control policy could be developed to
maintain the smooth functioning of such a simulated network economy. As such, this problem could contribute to the growing body of research in the area of Cognitive and Behavioural Economics, owing to the causal role of opinion on behaviour.
Appendix A

Ramped Evolution

This appendix presents a ramped difficulty evolution, in which the population begins by initially evolving to solve a problem instance with a high $\theta_G$ (=90%). The termination criteria of the standard EA are extended to include a single individual in the population reaching the simulation step limit (500000 steps). As with the standard evolution scheme an upper limit of 100 generations is used. Once the evolution is complete, the difficulty of the problem is increased by lowering the value of $\theta_G$ by 10% and the evolution is begun again. The population is evolved continuously through each level, so the population that reached the limit condition (step- or generation-) for one value of $\theta_G$ is the initial population for the next. This process is described in Algorithm 6.

Initial testing of the ENN algorithm considered using independent evolutions for each combination of the problem instance parameters $B$ and $\theta_G$. There, poor results are observed for low consensus threshold values where the problem difficulty is greater. Two potential causes for this are: a) There is a limit to how narrow the constraints to the problem can be before the solution space is unexploitable, in that no solution is better than any other, or b) The execution time is too short to allow for the sufficient training of better solutions to these instances. The ramped evolutionary scheme is considered in order to distinguish between these two cases. Using this scheme, each evolved population will have been given considerably more training time prior to the final testing for the lower values of $\theta_G$.

Does a ramped difficulty evolution improve the quality of the resulting neurocontroller?

The ramped evolution scheme is compared against the standard variant over budgets from 10% to 40% for both ENN and ECAM algorithms, and are presented in Figures
Algorithm 6 Pseudo-code of the Ramped evolution ENN for the $\theta$-CAP.

Require: $instance$ is a tuple $(\theta_G, B, G)$

1: procedure RAMPEDENN($instance$, $maxGen$)
2:  $instance.\theta_G \leftarrow 0.9$
3:  $Pop \leftarrow$ InitPop($instance$)
4:  while $instance.\theta_G \geq 0$ do
5:      $Pop \leftarrow$ TrainENN($instance$, $maxGen$, $Pop$)
6:      Report($Pop$)
7:  $instance.\theta_G \leftarrow instance.\theta_G - 0.1$

A.1 and A.2 respectively. The best solution found per parameter set throughout either evolution is tested over 100 independent simulations. A Wilcoxon Signed-Rank test paired by control system configuration seeds with a 99% confidence interval, was used to determine if the observed improvements in test results were significant in terms of improved solution quality. Results showing statistically significant improvement over their counterpart are indicated as dots along the $x$-axis in Figures A.1 and A.2. Dots below or above the $x$-axis indicate instances for which the ramped evolution performs significantly better or worse respectively on average. There it can be readily seen that the ramped evolutionary solutions exhibit improved results as compared to their independently evolved counterparts for low values of $\theta_G$. A lack of improvement over the $\theta_G = 0$ instances suggests that this level of constraint may be too severe to solve effectively with low budget values, however the improved performance over the 10-30 range suggests that the earlier hypothesis of additional time requirements is a feasible one. Notably, the ramped evolution fails to outperform the standard evolution at higher $\theta_G$ values. This trend is presumed due to the single-limit early stopping condition used in the ramped evolution, as compared to the early stopping occurring after five consecutive limits in the standard evolution. This was done in order to hasten the execution time of the ramped evolution over the set of all levels of $\theta_G$, as well as to avoid the potential of overfitting the population prior to resuming evolution for an altered problem.

Overall the ramped evolution outperforms the standard evolution more consistently as measured by the frequency of instances for which statistically significant
improved results are obtained. For the ENN algorithm ramped evolution outperforms standard at a ratio of 10:7:19 (ramped : standard : neither) over the 36 considered instances. For ECAM the ratio is 14:7:15. Many of the instances for which the ramped approach is outperformed could likely be improved if the more lengthy standard early stopping criteria are observed (at a loss of speed), since it is the only practical difference between the populations at high values of $\theta_G$. The results of this comparison show the effectiveness of the ramped evolution in training control systems that are capable of preventing convergence for greater time-spans on average for low values of $\theta_G$. The diminished performance at high levels of $\theta_G$ are seen as acceptable losses that do not prevent the development of higher quality solutions for the lower, more difficult values.

**Why does the ramped approach outperform the standard evolution?**

Interestingly, the weights of the final population for low consensus threshold perform poorly when tested on higher and thus easier values. For example, the best weights evolved for the $\theta_G = 20\%$ portion of a ramped evolution achieve much lower testing
results on the $\theta_G = 70\%$ instance than the weights evolved for that instance specifically. This implies that the intuition which prompted the use of ramped evolution, namely that lowering $\theta_G$ simply creates a narrower constraint around the same optimal areas of the solution space, is unsupported. This is further corroborated by Figure A.3, which shows a typical learning curve for a ramped evolution. The sharp decreases in fitness correspond to decreases in $\theta_G$, followed by increases in fitness where possible, which indicates convergence towards new local optima. Despite this pattern, the quality of solutions is still improved for lower values of $\theta_G$ as compared to the standard evolutionary scheme.

If selection pressure is too high in independent standard executions of small $\theta_G$ values, leading to comparatively poor results (premature convergence to local sub-optima) then the benefit of ramped evolution may come from pre-conditioning the population into potentially fit regions along the low-pressure fitness surface within large $\theta_G$ instances prior to narrowing the constraint. This would explain both the shape of the convergence curve, as well as the improved testing results versus standard
evolution, but not the ineptitude of the best solutions of narrow problems when applied to wider ones. It was considered that the improved testing results may have simply arose from the larger range of potential values for ANN weights given the additional generations of random shift diffusion from their initial $[-0.5, 0.5)$ range. However, testing of standard evolution with wider initial ranges demonstrated poor performance. If the highly fit regions of the solution space for larger values of $\theta_G$ are simply distinct from those of lower ones, then it remains unclear why the ramped evolution would provide superior results.

Figure A.3: Typical learning curve of a ramped evolution. Vertical lines indicate a decrease in $\theta_G$. 
Bibliography


