THE EVOLUTIONARY DYNAMICS OF BIOCHEMICAL NETWORKS IN FLUCTUATING ENVIRONMENTS

A DISSERTATION SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

2011

By
Robert John Platt
Chemical Engineering and Analytical Sciences
Contents

Abstract 11
Declaration 13
Copyright 15
Acknowledgements 17

1 General Introduction 19
   1.1 Genetic Architecture and the Interaction of Genes 19
       1.1.1 The Physical Relationship between Genes 20
       1.1.2 Epistasis: Genes can Interact at Multiple Levels 21
       1.1.3 Pleiotropy: Genes can affect Multiple Traits 23
       1.1.4 Evolvability 24
       1.1.5 Modularity 26
       1.1.6 Robustness and Neutral Mutations 30
       1.1.7 The Varying Environment 33
       1.1.8 Repeats 36
       1.1.9 Genes Interactions, Regulation and Lifestyle 37
   1.2 Laboratory Evolution in Fluctuating Environments 39
   1.3 Approaches to Simulating the Emergent Properties of Gene Interactions 40
       1.3.1 The Genetic Algorithm as Simulation of Evolution 41
       1.3.2 Are Genetic Algorithms the Right Modeling Tool? 43
       1.3.3 The Relationship between Simulated Genomes and Real Genomes 45
       1.3.4 Overview of the Thesis 47

(51200 words) 3
2 The Evolution of Robustness to Noise in Protein-Protein Oscillators 51
2.1 Introduction .......................................................... 52
2.2 Methods ............................................................... 54
  2.2.1 Representation of protein-protein interactions .................. 55
  2.2.2 A framework for evolving networks: sbolve ..................... 56
  2.2.3 The fitness function for evolving oscillators ................... 57
  2.2.4 The fitness function for robustness ............................ 60
  2.2.5 The pruning algorithm ........................................... 62
  2.2.6 Population size and mutational parameters ..................... 65
2.3 Results ............................................................... 67
  2.3.1 Robustness as a Selected Trait ................................ 67
  2.3.2 Network Complexity and Robustness ............................ 70
  2.3.3 Feedback loop topology was highly varied ...................... 72
  2.3.4 What makes an oscillator robust? .............................. 73
2.4 Discussion ............................................................ 81

3 Generalism and Genetic Architecture in Varying and Constant environments 85
3.1 Introduction .......................................................... 86
3.2 Methods ............................................................... 91
  3.2.1 Avida Software ..................................................... 91
  3.2.2 Treatments ......................................................... 91
  3.2.3 Populations ......................................................... 92
  3.2.4 Population Statistics ............................................. 93
  3.2.5 Genotype-Phenotype Mapping .................................... 94
  3.2.6 Determination of Modularity ................................... 94
  3.2.7 Evolvability ....................................................... 97
    3.2.7.1 Measuring Evolvability .................................... 97
    3.2.7.2 Testing the Link between Modularity and Evolvability 98
3.3 Results ............................................................... 98
  3.3.1 Treatments can be partitioned by Specialism/Generalism ...... 98
  3.3.2 Phenotypic Diversity ............................................ 106
  3.3.3 Genetic Architecture: Epistasis ................................ 107
  3.3.4 Genetic Architecture: Mutational Robustness .................. 107
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.5 Modular Pleiotropy</td>
<td>110</td>
</tr>
<tr>
<td>3.3.6 Evolvability with Changing Environment</td>
<td>112</td>
</tr>
<tr>
<td>3.3.7 Putting it All Together: Genetic Architecture and Evolvability</td>
<td>117</td>
</tr>
<tr>
<td>3.4 Discussion</td>
<td>120</td>
</tr>
<tr>
<td>4 Contingency Loci, Housekeeping Genes and the <em>E. coli</em> Core Genome</td>
<td>125</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>126</td>
</tr>
<tr>
<td>4.2 Methods</td>
<td>130</td>
</tr>
<tr>
<td>4.2.1 <em>E. coli</em> Sequences</td>
<td>130</td>
</tr>
<tr>
<td>4.2.2 Detection of Simple Sequence Repeats</td>
<td>131</td>
</tr>
<tr>
<td>4.2.3 Orthology of Genes and Functional Annotation</td>
<td>132</td>
</tr>
<tr>
<td>4.2.4 Variability of Genes</td>
<td>133</td>
</tr>
<tr>
<td>4.3 Results</td>
<td>134</td>
</tr>
<tr>
<td>4.3.1 GC Content</td>
<td>134</td>
</tr>
<tr>
<td>4.3.2 Repeats across the Genome and Ecotypes</td>
<td>136</td>
</tr>
<tr>
<td>4.3.3 Repeats and the Genetic Variability of Genes</td>
<td>137</td>
</tr>
<tr>
<td>4.3.4 Repeats and Core Genes</td>
<td>140</td>
</tr>
<tr>
<td>4.3.5 Repeats and Functional Categorization</td>
<td>142</td>
</tr>
<tr>
<td>4.3.6 Repeats and Essentiality for Cell Growth</td>
<td>149</td>
</tr>
<tr>
<td>4.4 Discussion</td>
<td>151</td>
</tr>
<tr>
<td>5 Experimental evolution of <em>E. coli</em> in a varying environment</td>
<td>155</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>156</td>
</tr>
<tr>
<td>5.2 Materials and Methods</td>
<td>160</td>
</tr>
<tr>
<td>5.2.1 Strains</td>
<td>160</td>
</tr>
<tr>
<td>5.2.2 Treatments</td>
<td>161</td>
</tr>
<tr>
<td>5.2.3 Media and Growth Conditions</td>
<td>162</td>
</tr>
<tr>
<td>5.2.4 Selection of the carbon sources</td>
<td>164</td>
</tr>
<tr>
<td>5.2.5 Fitness Assays</td>
<td>172</td>
</tr>
<tr>
<td>5.2.6 Determination of Growth Rate, Lag and Stationary Phase Times</td>
<td>173</td>
</tr>
<tr>
<td>5.3 Results</td>
<td>174</td>
</tr>
<tr>
<td>5.3.1 Fitness of Controls Relative to Ancestor</td>
<td>174</td>
</tr>
<tr>
<td>5.3.2 Relative Fitness in All Treatments</td>
<td>179</td>
</tr>
<tr>
<td>5.3.3 Analysis of Growth Curves</td>
<td>182</td>
</tr>
<tr>
<td>5.4 Discussion</td>
<td>186</td>
</tr>
</tbody>
</table>
A Methodology: The Application of Genetic Algorithms in Evolutionary Systems Biology
A.1 Introduction ................................................. 196
A.2 Methods ......................................................... 201
   A.2.1 Choosing the Model of Selection ......................... 201
   A.2.2 Genome Representation .................................... 203
   A.2.3 Systematic Bias in Mutations .............................. 206
   A.2.4 The Fitness Function ....................................... 208
   A.2.5 Calculation of Mutational Robustness .................... 209
A.3 Results ......................................................... 210
   A.3.1 Comparison of Truncation and Tournament Selection .... 210
   A.3.2 Confirmation of Unbiased Mutations ....................... 213
   A.3.3 Genome Size and Elitism .................................... 213
   A.3.4 Genome Size is Under Selection ........................... 217
A.4 Discussion ...................................................... 220
B Kinetics for Two Simple Robust Oscillators ....................... 225
   B.1 Kinetics of a robust two-protein oscillator ................. 225
   B.2 Kinetics of a robust three-protein oscillator ............... 226
C Avida Configuration ............................................. 229
Bibliography ....................................................... 246
List of Tables

2.1 ANOVA of Network size ........................................ 72
3.1 The NOR Boolean logic function ............................... 89
3.2 Avida resources and their value for the two environments ...... 92
3.3 Post-hoc Tukey test of mutational robustness of fitness ........ 110
3.4 Post-hoc pairwise testing of changes in fitness after 1000 updates .... 116
3.5 Analysis of Covariance for evolvability .......................... 118
4.1 Sequenced E. coli strains and their ecotypes ..................... 135
4.2 ANOVA of genetic variability .................................... 140
4.3 Cross tabulation of core/non-core genes are repeat significance .. 141
4.4 Contingency table of effects between repeat coverage and gene essentiality 150
5.1 ANOVA of relative fitness in control populations ............... 176
5.2 Post-hoc analysis of interaction between treatment and competition environment ........................................ 177
5.3 ANOVA of competition and treatment environments ............ 180
5.4 Post-hoc testing of interactions between environment and treatment .. 182
A.1 The NAND Boolean logic function ............................... 203
A.2 Fitness functions used for the model of network evolution in our study 209
A.3 Tukey post-hoc tests for various approaches to analysing genome size .. 216
# List of Figures

2.2.1 Steady state solution instead of oscillator ........................................ 58
2.3.1 Robustness of both amplitude and period of the evolved oscillators .... 68
2.3.2 Network properties compared between non-robust and robust oscillators 69
2.3.3 Correlation between generations and genome size ............................. 71
2.3.4 Examples of some of the simplest oscillators evolved ........................ 75
2.3.5 Relationship between initial concentration, amplitude and period ........ 76
2.3.6 Scaling of the Jacobian ....................................................................... 78
2.3.7 The order of magnitude difference between kinetic parameters ........ 80

3.2.1 The parabolic relationship found between our measure of epistasis and  
    functional modularity ........................................................................... 100
3.2.2 Relationship between modularity and epistasis .................................. 101
3.3.1 Number of resources utilized ............................................................. 102
3.3.2 Coding size of genotype, and correlation with tasks performed ........ 103
3.3.3 Genome size ...................................................................................... 104
3.3.4 Specialism in digital organisms ......................................................... 105
3.3.5 Phenotypic diversity .......................................................................... 106
3.3.6 Density of epistatic interactions in dominant organisms ................. 108
3.3.7 Average neutrality of point mutations for instructions coding for traits 109
3.3.8 Modularity of traits in dominant genotypes ....................................... 111
3.3.9 Fitness before and after the environment was perturbed by switching  
    populations ......................................................................................... 113
3.3.10 Evolvability (changes in fitness after 1000 updates) ........................ 114
3.3.11 Evolvability, tasks performed, and interaction with treatment .......... 121
4.3.1 Over-representation of repeats by ecotype ........................................ 136
4.3.2 Ecotype and proportion of genome which is non-core. Error bars show
95% confidence intervals. .................................................. 138
4.3.3 Variability of core and non-core genes .......................... 139
4.3.4 Functional categories organized by descending proportion of genes that
are core ................................................................. 143
4.3.5 Functional categories organized and over-represented repeat coverage
(core) ........................................................................ 144
4.3.6 Functional categories organized and over-represented repeat coverage
(upstream) ................................................................. 145
4.3.7 Fisher exact tests of effects between core and non-core and over- and
under-represented repeats ................................................. 146
4.3.8 The relationship between core/non-core and repeats across functional
categories ................................................................. 148
5.2.1 Contamination checking plate used during competition assaying ...... 163
5.2.2 Minimum flux change between E. coli carbon sources ............... 168
5.2.3 Number of reactions changing between E. coli carbon sources ...... 169
5.2.4 A hierarchical clustering of the shortlisted carbon sources ......... 170
5.2.5 An example of a fitted growth curve. ................................ 175
5.3.1 Responses to lactose and glycerol compared ....................... 178
5.3.2 Relative fitness of treatments in lactose and glycerol ............... 181
5.3.3 Growth in glycerol: effect on lag, exponential and stationary phase .... 183
5.3.4 Growth rates of each treatment compared ....................... 184
5.3.5 Lag times of each treatment compared ............................ 185
A.2.1 An example genotype, phenotype and the emerging network ......... 207
A.3.1 Comparison of genetic algorithm selection models .................. 211
A.3.2 Mutations are not biased towards increasing or decreasing genome size . 212
A.3.3 Elitism tends to lower the number of generations required .......... 214
A.3.4 Genome size, measured in various ways .......................... 215
A.3.5 Robustness to mutations ............................................. 219
Abstract

University of Manchester

Robert John Platt

Doctor of Philosophy

The Systems Biology and Evolutionary Dynamics of Biochemical Networks in Fluctuating Environments

30 September 2010

Typically, systems biology focuses on the form and function of networks of biochemical interactions. Questions inevitably arise as to the evolutionary origin of those networks' properties. Such questions are of interest to a growing number of systems biologists, and several groups have published studies shown how varying environments can affect network topology and lead to increased evolvability. For decades, evolutionary biologists have also investigated the evolution of evolvability and its relationship to the interactions between genotype and phenotype. While the perspectives of systems and evolutionary biologists sometimes differ, their interests in patterns of interactions and evolvability have much in common. This thesis attempts to bring together the perspectives of systems and evolutionary theory to investigate the evolutionary effects of fluctuating environments. Chapter 1 introduces the necessary themes, terminology and literature from these fields. Chapter 2 explores how rapid environmental fluctuations, or "noise", affects network size and robustness. In Chapter 3, we use the Avida platform to investigate the relationship between genetic architecture, fluctuating environments and population biology. Chapter 4 examines contingency loci as a physical basis for evolvability, while chapter 5 presents a 500-generation laboratory evolution experiment which exposes E. coli to varying environments. The final discussion, concludes that the evolution of generalism can lead to genetic architectures which confer evolvability, which may arise in rapidly fluctuating environments as a by-product of generalism rather than as a selected trait.
Declaration

No portion of the work referred to in this dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://www.campus.manchester.ac.uk/medialibrary/policies/intellectual-property.pdf), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University’s policy on presentation of Theses.
I would like above all to thank my supervisors Daniel Rozen, Nils Blüthgen and Hans Westerhoff for all their support, ideas and encouragement throughout the doctorate. I am also indebted to Dominique Schneider at the Institut Jean Roget for providing the 20,000 generation *E. coli* strain, as well as Charles Ofria from Michigan State University and Claus Wilke from University of Texas, Austin for their helpful answers to our questions about Avida. Long sessions of coffee with Keiran Smallbone, in which he explained his geometric approach to FBA, was invaluable for my work in finding the minimal flux difference between carbon sources. Also thanks to Chris Knight for his suggestions for the analysis of growth curves. Thanks also go to my collaborator David Springate, since there would be one less chapter were it not for his dedication, brains and tea. I wish also to acknowledge the insights and hot beverages I gained from other members of my group: Daniel Engelmaier, Andres Arce, Paul Johnston, and Ben Evans. Emma Sherratt, too, gave me useful advice on phylogenetics.

*I dedicate this thesis to Sheena.*
Chapter 1

General Introduction

1.1 Genetic Architecture and the Interaction of Genes

Genetics is a field of science rooted in a particle model of heredity, in which the units of inheritance – genes – are discrete. In their idealized form, genes have a one-to-one correspondence with traits and can be inherited independently. Of course, in reality genes interact in order to generate expression of a given phenotype and they are often not inherited independently of each other. This was understood very early on in the history of genetics. The term epistasis, for the masking of the traits of one gene by another, was first coined by Bateson around 1908 (Bateson, 1908, 1909). But the masking of traits is just one example of how two genes can affect the phenotype in an interdependent manner, and epistasis has gained a wider meaning for many kinds of gene interactions (Phillips, 1998). In terms of the interdependent inheritance of genes, it was Bateson again who showed with Punnett in 1910 that combinations of genes for flower colour and pollen shape could be inherited with different frequencies, resulting
in phenotypes in ratios different from what was expected from the Mendelian model of inheritance (Bateson and Punnett, 1911). This is an example of genetic linkage, where genes physically closer together on the chromosome have less chance of undergoing recombination. Epistasis and genetic linkage demonstrate that genes can be interrelated both in terms of how they affect the phenotype and how they are inherited. The structure of the complex relationships that can exist between genotype and phenotype is collectively referred to as the genetic architecture, and is widely studied both theoretically and experimentally (Hansen and Wagner, 2001; Mackay, 2009). In this Introduction to my thesis, I will review and discuss some of the mechanisms by which genes can be dependent upon each other in more detail, introducing several critical concepts and developing towards the questions addressed in this thesis.

1.1.1 The Physical Relationship between Genes

There are many ways in which genes can be interdependent through their sequence position. In the extreme case, genes can overlap, sharing base pairs of DNA (or RNA for some viruses). Overlapping is possible not just on the anti-sense strand, but on a single strand because the codon sequence can be in a different reading frame for each gene (Miyata and Yasunaga, 1978). The overlapping of genes is most notable in viruses, which have a high rate of mutation per base pair and need to compress the genetic information into as few base pairs as possible, so as to reduce the overall mutation rate for the genome (Belshaw et al., 2007). Overlapping genes are also present in bacteria – and there are even examples in higher eukaryotes – but in bacteria and higher organisms overlapping has a regulatory purpose, with the overlapping genes being related in function and expression (Fahey et al., 2002; Johnson and Chisholm, 2004). This means that overlapping genes can interact both in terms of their evolutionary dynamics and their regulatory dynamics. They are thus of interest to the systems biologist.
1.1. GENETIC ARCHITECTURE AND THE INTERACTION OF GENES

as well as the evolutionary biologist, e.g. Krakauer (2000), which combines both the evolutionary and regulatory dynamics of overlapping genes into a single treatment. He gives mathematical arguments as to which forms of gene expression kinetics can favour overlap. Specifically, gene expression feedback (a systems property) with a high level of activation (a biochemical property) favour the evolution of overlap. This study by Krakauer is an example of biochemistry, systems biology and evolutionary biology being brought together to understand a complex problem.

Where overlapping genes have a lower capacity to undergo mutations independently, genes which are close together on the same chromosome have a lower capacity to be disassociated by recombination due to the physical association of genes on the genome (genetic linkage).

1.1.2 Epistasis: Genes can Interact at Multiple Levels

From a systems biologist’s perspective, a gene interaction would be interpreted to mean the expression of one gene influencing the expression of another gene, for example through a transcription factor. Genetic interactions in an evolutionary context, known as epistasis, can be any interaction where the effect of one gene (or mutation) on the phenotype is masked, enhanced or altered by another. A genetic interaction in this more general context is different from a gene interaction in a “gene network” or regulatory network where it is not necessarily the case that one gene is modifying the expression of the other gene.

There is a classical example of epistasis which demonstrates this difference – the rare hh blood group or ‘Bombay phenotype’ (Seifinejad et al., 2010). The A and B genotypes express enzymes which modify the H antigen to produce the A and B antigens. However, if an individual is homozygous recessive for the H antigen, then they will not produce the A or B antigens – in spite of the A and B enzymes being
expressed by their genes. Effectively, the H, A and B genes have interacted entirely at the protein level, with the result that one gene has masked the phenotype of the other two. Epistasis at the protein level is sometimes referred to as physiological epistasis, as opposed to genetic epistasis.

In an evolutionary context, the causes and consequences of epistasis is an important area of research, because it underpins the mutational deterministic hypothesis, a prevailing theory to explain the evolution of sexual reproduction. A key assumption of this hypothesis is that the effects of mutations on fitness are not additive but that subsequent mutations are more deleterious (synergistic epistasis). The opposite is antagonistic epistasis, where each mutation is lessened in the effect. Understanding the interaction of cumulative mutations, and its relationship to recombination, is a key area of research into epistasis. Previous research has shown that synergistic interactions may be balanced by antagonistic interactions, which does not support the mutational deterministic hypothesis (Elena and Lenski, 1997). There is strong evidence that epistasis is widespread, but it does not necessarily buffer against deleterious mutations (Elena and Lenski, 2001; MacCarthy and Bergman, 2007).

The prevalence of epistasis suggests that it should not be of interest to just evolutionary biologists. Indeed, medical researchers are increasingly recognizing the role of epistasis in complex multi-factorial diseases, such as Alzheimer’s disease, e.g. Combarros et al. (2009). Multi-factorial diseases are those where a certain combination of alleles – and in many cases, environmental and lifestyle factors – leads to a pathological phenotype that would not occur in an individual with only one of those alleles or factors. Systems approaches to multi-factorial diseases are being actively developed, due to the ability to perform high-throughput screening and various -omics techniques (Buhimschi et al., 2008; Glocke et al., 2006; Novère, 2008). It may be that evolutionary and systems biology have important contributions to make to understanding these
1.1. GENETIC ARCHITECTURE AND THE INTERACTION OF GENES

diseases.

1.1.3 Pleiotropy: Genes can affect Multiple Traits

Pleiotropy is where a gene, or a mutation in a gene, affects more than one phenotypic trait. One classic example of pleiotropy is the disorder phenylketonuria (Penrose, 1951), where a mutation in the gene for one enzyme can have pathological effects in multiple traits, including white matter development, ketones in body odour and hypopigmentation due to the inability to correctly form melanin (Anderson and Leuzzi, 2010; Fistarol and Itin, 2010). Phenylketonuria also demonstrates how the observation of pleiotropy depends on the environment; dietary management at an early stage completely changes the outlook (Poultie and Wildgoose, 2010).

Pleiotropy is not a form of gene interaction in itself, given that its definition is inclusive of a gene being solely responsible for several traits. But this is often not the case. For example, the hypo-pigmentation in phenylketonuria masks other genes responsible for skin pigmentation. There is therefore both epistasis and pleiotropy acting simultaneously. Multiple genes are interacting in multiple traits in a disorder that arises from just one mutation in one gene.

From an evolutionary perspective, a mutation which gives rise to pleiotropic effects may be beneficial or detrimental in different traits or environments, potentially leading to conflicting directions of selection. A pleiotropic mutation which is beneficial in more than one trait is called synergistic, while a mutation which is beneficial for a target of selection but deleterious for other traits is called antagonistic. Ostrowski, Rozen and Lenski (Ostrowski et al., 2005) carried out a laboratory evolution experiment to determine the extent to which synergistic and antagonistic pleiotropy are found with beneficial mutations in Escherichia coli. Populations of E. coli were cultured by daily transfer into liquid glucose medium, until the first beneficial mutation was
found. The effect of that mutation on the mutant’s fitness in novel carbon sources was then determined, by competing it against the ancestral, wild type strain. Where the beneficial mutation in glucose was also beneficial in another resource environment, that was a case of synergistic pleiotropy. Where the mutation was beneficial in glucose but not in another resource environment, that was case of antagonistic pleiotropy.

They found that pleiotropic effects were common. The pleiotropy was largely synergistic rather than antagonistic in this case. Antagonistic pleiotropy was only statistically significant in its effect on relative fitness for one carbon source: melibiose, and that was for just 3 out of 27 mutants. As the paper points out, pleiotropy was measured for traits that were related through carbon metabolism; there could be additional, antagonistic pleiotropic effects for traits that were not measured. They put forward an interesting hypothetical example: a beneficial mutation might reduce the lag phase of \textit{E. coli} growth, at the expense of fitness during stationary phase. However, they note that this was not observed in an experiment by Vasi \textit{et al.} (Vasi et al., 1994), where \textit{E. coli} adapted under very similar conditions. Perhaps exposure of the evolving strain to both lag and stationary growth during the 24 hour period discouraged mutations which were antagonistic for the two phases. This demonstrates the extent to which pleiotropy depends upon evolutionary and environmental context.

1.1.4 Evolvability

Since pleiotropy and epistasis tie the adaptation of phenotypic traits together, they potentially restrict the extent to which those traits can adapt independently. They therefore affect the \textit{evolvability} of the organism – its ability to adaptively respond to a change in the environment. However, the potential effects of pleiotropy and epistasis on evolvability is a matter of context. Hansen (2003) argues that if there are multiple sources of pleiotropy which can affect traits differently, and can cancel each other out,
then pleiotropy can increase variability while still allowing the genome to adapt flexibly. This increases evolvability. But if all sources of pleiotropy for a pair of traits have a similar effect, then evolvability may be impaired.

Evolvability has a number of definitions (Jones et al., 2007; Pigliucci, 2008). It can be defined as the ability of an organism to generate novel, heritable phenotypic variation (Kirschner and Gerhart, 1998), although there are a number of other related meanings in the literature. It is also used in the context of heritable variation at the level of the population, such as standing genetic variation, or the rate at which a single trait adapts in an environment. Since all of these contribute to how effective organisms are at persisting in diverse environments, Pigliucci (2008) recommends acknowledging that evolvability comprises a broad spectrum of effects. While it would be impractical for in vivo studies to measure all forms of evolvability, it is possible to assess different aspects of evolvability in computational models of evolution. However, network approaches to evolvability, such as Kashtan et al. (2007) and Crombach and Hogeweg (2008), have focused on how the properties of a biochemical network may affect the adaptive rate of traits only of the fittest individual. There is therefore scope for a study which integrates the network approach with population biology.

Previous research suggests that evolvability is itself heritable, and is selected for over time under environmental fluctuations (Earl and Deem, 2004). This has also been the focal point of evolutionary studies in the systems biology community, particularly because both environmental uncertainty and evolvability have been linked to a property known as modularity, in simulations of evolving networks (Kashtan et al., 2007; Parter et al., 2008). Modularity is introduced and discussed next.
1.1.5 Modularity

Modularity refers to the extent to which a system is made up of independent, or partly independent, sub-components. In a modular architecture, interactions and relationships within a module (e.g. between genes) are stronger and more numerous than between modules. The most intuitive examples of modularity are communities in human (and animal) social networks, and indeed much of the theory that has arisen around modularity is in terms of finding community structure, e.g. the Newman and Girvan (NG) algorithm (Newman and Girvan, 2004). In spite of being an intuitively simple concept, modularity is very difficult to define precisely, as well as to calculate, because of the combinatorial explosion in the ways a network of individuals can be subdivided with respect to network size. There are many ways to subdivide a network or population into communities, particularly given that multiple levels of structure may exist. Consider for example human interactions, which have structures at the level of families, friends, work colleagues, villages, countries, and so on. What is the ‘correct’ way to subdivide the population into communities? Therefore many algorithms, such as the NG algorithm, produce a hierarchy of potential clusterings, and then calculate the quality $Q$ of each potential clustering by comparing the strength of interactions between members of the same community (which should be high) to the strength of interactions between members of other communities (which should be low). A poor partitioning of a network – one which does not reflect the structure of the network – will have a low quality score. The hard problem computationally is maximizing $Q$ given the large number of potential ways to subdivide a network, and there are many approaches to determining the clustering/partitioning with the highest quality (Lancichinetti and Fortunato, 2009). The most popular ‘benchmark’ algorithm in use is NG, and it is found in many systems biology studies, e.g. Kashtan and Alon (2005). In statistical studies such as
1.1. GENETIC ARCHITECTURE AND THE INTERACTION OF GENES

these, it is not the clustering itself but the quality of it which is of most interest. If a network has no structure, then no subdivision will exist with a high $Q$. Therefore, the maximum $Q$ is a measure of modularity.

In terms of genetic architecture, pleiotropy and epistasis create dependencies in the genome and so affect the structure of the genotype-phenotype mapping and its modularity. The intuitive view is that increasing modularity will increase evolvability, because it allows genes and the traits they affect to evolve more independently. But Hansen (2003), as we discussed above, suggests that pleiotropy can contribute to evolvability, even if it reduces modularity. Indeed, Hansen goes so far as to suggest that evolvability may be harder to achieve by modularity than by other means. Does this contradict the observation of modularity in bioinformatic analyses of metabolic and regulatory networks (Ma et al., 2004; Zhao et al., 2006; Samal et al., 2006)? Not necessarily, primarily because modularity may still be one of the sources of evolvability. The mathematical models presented by Hansen lead him to suggest that an ‘intermediate’ level of pleiotropy may maximize evolvability, suggesting that some modularity will still be observed.

We need to be careful because modularity is used in many different contexts in the literature and can be demarcated using many different criteria. Here is a rough list of the various contexts in which modularity is used:

1. Physical modules: protein complexes, molecular machines, enzymes which share a common substrate (Wilhelm et al., 2003),

2. Topological modules: clusters of nodes in a biochemical network which are more highly connected to each other than to the rest of the network (Schuster et al., 2002),
CHAPTER 1. GENERAL INTRODUCTION

3. **Functional modules**: biochemical molecules that are grouped by function (Wilhelm et al., 2003),

4. **Regulatory modules**: enzymes and processes which are co-regulated, also used in specific contexts such as Cis-regulatory modules (Nam et al., 2010),

5. **Evolutionary modules**: characteristics of an organism which do not, or cannot, adapt independently. Includes pleiotropic modularity (Wagner et al., 2007) which could also be physical in the form of overlapping genes or genetic linkage,

6. **Developmental modules**: morphological subunits in the developmental processes of an embryo or in the adult anatomy, such as the subdivisions of a *Drosophila* wing (Klingenberg, 2009),

7. **Motifs**: topological patterns which are overrepresented in the biochemical network (or between species) compared to what would be expected by chance, and therefore are presumed to be significant (Mangan et al., 2003; Ward and Thornton, 2007),

8. **Evolutionary motifs**: Motifs which occur because they are duplicated and re-used during evolution (Ward and Thornton, 2007).

This list neither attempts to be comprehensive nor capture all the subtleties of each case; behind each type of module is a whole field of study and debate. Nevertheless, there are strong reasons to expect overlap between the different types of modularity. This can best be demonstrated by example. Proteins that form a complex constitute a physical module, but since a protein complex has a function, these proteins are also part of a functional module. They are also part of an evolutionary module, because the sequence of each protein’s gene will have an effect on the physiological state of all the proteins (i.e. certain mutations will change the dynamics of complex formation). It
1.1. GENETIC ARCHITECTURE AND THE INTERACTION OF GENES

may even be an evolutionary motif of sorts; one proposed mechanism for the evolution of protein complexes is through gene duplication of homomers (Pereira-Leal et al., 2007).

A further demonstration can be found with ‘communication modules’: the two-component signaling systems in bacteria. The sensor and receiver in a two-component system are functionally and physically related, would not be able to evolve entirely independently without impairing function of the communication module, and their possible lateral transfer during evolution suggests an evolutionary motif (Parkinson and Kofoid, 1992).

And in both examples given above, the kinetics of the interaction between the proteins will be affected by the expression of the proteins. Thus, their expression should not be entirely independent, i.e. they should be co-regulated – perhaps by being in the same regulon or operon. Functional and physical relationships between proteins expressed on the same operon are often, if not always, observed (Zaslaver et al., 2006), not least due to recombination and the need to avoid breaking up groups of genes which are co-adapted (Martin et al., 2005). Thus one would expect the proteins to be part of the same regulatory module (though regulatory and evolutionary modules are not the same thing, as will be shown in the next section).

In summary, modularity at one level should correlate with modularity at other levels, and regulatory demands can contribute to the patterns of modularity observed in nature. Combine this with Hansen’s argument that evolvability and modularity do not have to go together, and we are led to the following questions. To what extent has modularity arisen simply to meet regulatory needs, and to what extent has it arisen as a form of evolvability? Is it possible that modularity largely arises not so as to increase evolvability but rather due to constraints on the system?
1.1.6 Robustness and Neutral Mutations

Another potential source of integration between components of a functional module, or the genotype and phenotype mapping, is robustness.

Just as modularity can have very many different meanings, so too can robustness. The first type we will consider is known as mutational robustness. In section 1.1.1 on page 20, we mentioned that RNA viruses use gene overlap to reduce the average mutational load per individual in the population (mutational load is dependent upon population in classical population genetics, although recent work has led to surprising contradictory results, e.g. Glémin (2010), where mutational robustness was independent of population size). Another way of reducing mutational load is for mutations to be neutral (van Nimwegen et al., 1999), on the basis of mutational load being in proportion to average mutation rate. It has been observed that naturally occurring proteins are very stable with respect to point mutations, while de novo designed proteins are not so robust (Taverna and Goldstein, 2002) – suggesting that mutational robustness has been selected for in proteins. Yet these studies concern the robustness of the protein structure, and we are more interested in the robustness of the interactions at the level of the biochemical network. Other work has focused on the robustness of the secondary structure of RNA, such as Wagner (2008).

A number of studies have looked at neutral mutations in the context of regulatory networks. In particular, computational studies have suggested that neutral mutations can improve evolvability in the long term – because neutral mutations are intermediaries through which to reach fitter networks – while, in the short term, they may make novel, beneficial mutations more unlikely (Ciliberti et al., 2007a; Wikls et al., 2008). This means that the effect mutational robustness has on evolvability is timescale dependent; robustness increases evolvability in the long term by relaxing selection so
1.1. GENETIC ARCHITECTURE AND THE INTERACTION OF GENES

as to accumulate novel genetic diversity, but in the short term a lower intensity of
selection reduces evolvability (Elena and Sanjuán, 2008). Mutational robustness also
contributes to survival under high mutation rates, sometimes referred to as “survival
of the flattest”. Survival of the flattest has been observed in digital organisms (Wilke
et al., 2001) and in studies using complementary experimental and theoretical models
of viroids (Codoñer et al., 2006; Elena et al., 2008).

Mutational robustness in biochemical networks could mean either robustness to
changes in topology (rewiring/adding/removing an interaction), to changes in kinetic
parameters or to changes in the concentrations of interacting metabolites or proteins.
Computer simulations to find possible arrangements of the segment polarity network
which reproduce the patterns observed in Drosophila led to many solutions which
were also highly robust to parameter perturbations (von Dassow et al., 2000; von
Dassow and Odell, 2002). Indeed, the simplest possible network was also very robust
to perturbations in both parameters and initial protein concentrations.

Robustness to initial concentrations has a different physiological interpretation:
*robustness to noise*. Stochasticity in gene expression can interfere with the organism’s
ability to regulate and respond to signals (Swain et al., 2002) and so regulatory systems
need to be robust to fluctuations in gene expression. However, the main source of noise
is transcription, rather than translation (Kollmann et al., 2005). In bacterial systems,
genes for the same functional module may be placed along one operon, so that the
transcriptional noise affects all proteins in a correlated way. So robustness to noise
is partly achieved through a modular regulatory arrangement. Such an arrangement
is found in the E. coli chemotaxis pathway (Kollmann et al., 2005) and the KaiC
circadian clock in the cyanobacterium Synechococcus elongatus (Clodong et al., 2007).
It was found, in these cases, that robustness to correlated noise was only achieved for
a small number of possible network designs – designs which agreed with experimental
CHAPTER 1. GENERAL INTRODUCTION

observation. At least in the case of the chemotactic system, robustness to uncorrelated noise was difficult to achieve.

Comparing the computational designs of these bacterial systems and the segmentation pathway of Drosophila, we see a striking difference. In one case, robustness to uncorrelated noise was difficult to achieve, whereas in the other case it was easily achieved for many different designs and even for the simplest design. This indicates that the constraints on robustness may depend upon the particulars of the system and the function that it is meant to achieve.

Robustness does not necessarily require greater complexity, or particular topologies – even to cope with uncorrelated noise or perturbations to individual parameters. It depends on the system in question. This is in disagreement with common views about robustness; that it requires greater complexity (Ciliberti et al., 2007b), and that is a topological feature (Kollmann et al., 2005; Clodong et al., 2007; Tsai et al., 2008). Indeed, Dassow and Odell, who authored the Drosophila study, were surprised by their result of finding simple, robust designs. Yet one can intuitively grasp why simpler networks can be more robust: a more complicated network has more proteins whose concentrations may adversely affect the kinetics of the system. This argument was put forward by Leclerc (2008). But I do not go as far as he does, and argue that robust networks will always favour a sparse design. The most robust design found for the chemotactic system was more complex than the others. Rather, I think that the topological requirements of robustness are problem-specific.

These findings also cast doubt on the claim, made by Ciliberti et al. (2007b), that there is a strong correlation between mutational robustness and noise robustness. This is based upon noise robustness to the perturbation in the expression of one gene – but we have seen in bacterial signaling and the KaiC circadian clock that robustness to correlated changes is sufficient in certain cases.
1.1. GENETIC ARCHITECTURE AND THE INTERACTION OF GENES

So, in some cases neither noise nor mutational robustness require a special topological solution. Robustness can sometimes be achieved without impacting on the integration between components: there is no simple or universal relationship between robustness and modularity. That said, these studies focus upon specific networks in specific organisms, and try to generalize from there. We try to address this in 2 on page 51, by evolving a wide range of possible protein networks as oscillators and examining their topology and robustness.

While functional modules might be arranged into operons so as to limit uncorrelated noise, operons can also be explained by the need to co-regulate the proteins in response to a signal, and the efficiency of co-expressed genes being on the same transcript (see 1.1.9 on page 37). Which of these explanations might be most important for the observed arrangement of operons remains an important question.

1.1.7 The Varying Environment

In living systems, the concept of a varying environment is clearly a relative one; all environments vary to some extent. However, in computer simulations, it is entirely possible to keep all properties of the environment entirely fixed, and compare it to variable environments. This allows us to study how the variability inherent in all environments affects evolutionary dynamics. Crombach and Hogeweg (2008) used a computer simulation of evolution with a simple representation of the genome as a sequence of genes and binding sites, representing a Boolean network in which the genes could either be expressed (on) or unexpressed (off). The environment consisted simply of a target pattern of gene expression, so that the fitness of individuals was greater when the correct genes were on, but lower if those genes were off or the incorrect genes were on. They then varied the environment seasonally, such that maximization of fitness was realized when gene expression patterns alternated at regular intervals
CHAPTER 1. GENERAL INTRODUCTION

throughout evolution.

The varying environment in their study led to the evolution of evolvability. The networks which were selected under a changing environment were able to respond to the change with just a few beneficial mutations over a shorter period. Surprisingly, they also found that the mutational robustness was unaffected by this improvement in evolvability. The basis for this evolvability was topological – the regulatory networks contained a hub with widespread effects on other genes. Just a few mutations near a hub will change the expression state of many genes. This seems trivial, because fitness in this study directly depended upon having many expression states change. Nevertheless, it would be impressive if a small number of mutations could lead to any phenotype in more biologically plausible or realistic scenarios; this would allow very rapid adaptation to a novel environment. Unfortunately, the paper does not demonstrate that the hub-based networks can rapidly adapt to entirely new goals. The idea remains promising and it has been suggested that a hub connecting modules in the *Escherichia coli* heat shock response system allows such evolvability (Kurata et al., 2006).

Other *in silico* simulations of biological evolution have also suggested that varying environments can “speed up” evolution and have consequences for modularity (Kashtan et al., 2007). Varying evolutionary goals increase the rate of convergence towards an optimal solution of a mathematically stated problem, but only if the varying goals share sub-goals. The sub-goals correspond to modules in the network itself (Kashtan and Alon, 2005), so that minimal ‘rewiring’ of the network is required when the goal changes. However, these studies required the goals of evolution to vary modularly, and the goal was phrased in terms of the expected phenotypic response. Given that the genotype directly represented the network in the phenotype (indeed, as far as I can tell the genotype could be written down from the phenotype) their conclusions
1.1. GENETIC ARCHITECTURE AND THE INTERACTION OF GENES

seem uncomfortably circular. In real organisms, the relationship between genotype and phenotype, i.e. genetic architecture, is much more complex.

The examination of the relationship between variable environments and modularity is promising but lacks broad support. Some studies support the view that variation in the environment leads to modularity (although causality has not been established). Parter et al. (2007) categorized hundreds of bacteria based upon the diversity of their natural habitats, and then compared those assessments to a measure of modularity of their metabolic network from the KEGG database. They found a positive correlation between diversity of habitat and modularity.

However, the claims that modularity arises due to varying environments have been disputed by others, such as Soyer (2007). Soyer shows that modularity can arise in a signaling network simply from the functional requirement of distinguishing between two input signals. Interestingly, Soyer points out that the modularity generated in this simulation depends upon the mutational parameters used (mutation rates for adding new interactions, removing existing interactions, and duplication). Soyer indicates, though, that it becomes more difficult for mutational processes to reach a solution with modularity as network size increases. In an extensive review of the various views and models of modularity, Wagner et al. (2007) concludes that both mutational bias and adaptive pressures may be needed to achieve modularity.

There is one other observation to be made, which leads to a key problem this thesis wishes to address. Soyer’s simulation had fitness based on the ability of the signaling network to distinguish between two signals. But there is only one reason why an organism wants to discriminate between two signals – to respond to relevant features of the environment. Now, the meaning of ‘varying environment’ used in the these papers is changing selection pressure over evolutionary time. The response in mind in this case is rewiring the network. But Soyer’s simulation can be re-interpreted as
evolving modularity to produce a highly discriminatory response still to environmental changes, but now within the organism’s lifetime. We could call one the regulatory timescale and the other the evolutionary timescale. The scoring Parter et al. use for the diversity of habitat does not discriminate environmental variability on both of these timescales – so the modularity they found could be due to either, or both, timescales.

As discussed earlier in the introduction, regulatory and evolutionary modularity are not equivalent. This is at least the case where the relationship between genotype and phenotype is not trivial. This raises two issues. First, the relationship between genotype and phenotype should not be trivial in models of evolution, so that the effects of varying environments on modularity and evolvability can be examined in the context of the interaction between genes and traits. Second, the effect of fluctuating environments on genetic architecture should be put in context with the fact that real organisms sense inputs and regulate their responses in their environment.

1.1.8 Repeats

The DNA sequence of genes can lead to differences between how and at what rates individual genes experience mutation and recombination, contributing to the physical architecture to the genome. For example, tandem repeats of nucleotides (e.g. ATATATATAT...) can cause slippage of DNA replication machinery, resulting in hypermutability (Moxon et al., 2006). Known as contingency loci, this allows localized genetic variation in traits where high levels of variability might provide a substantial advantage, such as in antigenic response. Such loci are believed to increase the evolvability of organisms which face unpredictable or fluctuating environments. We examine contingency loci as a physical form of evolvability in 4 on page 125. We will also revisit the issues of overlapping and linked genes later in this introduction, when considering approaches to simulating the evolution of genomes.
1.1. GENETIC ARCHITECTURE AND THE INTERACTION OF GENES

1.1.9 Genes Interactions, Regulation and Lifestyle

Physically and functionally related proteins need to be co-regulated. The relative concentrations of the proteins will affect the kinetics of their interaction, and therefore, these concentrations need to be co-ordinated so as to maintain function. But perhaps this is not quite satisfactory in itself. Couldn’t a biological system regulate each protein independently, so as to maximize modularity and evolvability?

One answer is that the logic for regulation needs to be provided somehow, and we expect this to be reflected in the organization of the regulatory network. To respond to a signal from the environment, an organism must often up-regulate or down-regulate entire functional systems, so the proteins in that system should be downstream of the same regulatory protein. Then again, the function’s response might need to integrate different signals, and so one would not expect perfectly distinct regulatory modules. There may be overlap and integration in the hierarchical structure of the global regulatory network, and the pattern of overlap will depend upon the physiological needs of the organism. Indeed, this has been demonstrated in *E. coli* in the computational integration of experimental data (Baldwin et al., 2005; Barrett et al., 2005).

Would modularity in the regulatory network lead to evolutionary modularity? Consider two genes which are downstream of the same regulatory protein, perhaps by being in an operon sharing a promoter. A mutation in a gene expressing the regulatory protein will affect the expression of the regulated genes, potential masking or enhancing the effects of mutations in these genes. This is epistasis, which was introduced in section 1.1.2, and epistasis may also be operating if the products of the downstream genes interact, which they often do in regulatory modules (e.g. by forming complexes), putting them in the same functional module. While functional, regulatory and evolutionary modules are not necessarily equivalent, we can expect the regulation of functional
However, as this thesis will show, genetic architecture and network topology may be affected by diverse, rapidly fluctuating or unpredictable environments even without taking regulation into consideration. Parter et al. (2007) found that the metabolic pathways of organisms with lifestyles associated with variable environments were more modular, but they also found that the metabolic maps were larger, too. However, in Parter’s paper, variable environments are effectively defined not as temporal fluctuations or spatial variation of environmental properties, but rather as the relative ecological ‘richness’ of the environments. For example, host associated bacteria were considered to have less variable environments than soil bacteria – a debatable classification, as host-associated bacteria can be subject to rapid reciprocal adaptation with their host (Schulte et al., 2010). They did also quantify environmental variability by how many transcription factors the organism has (on the basis that more dynamic environments require more regulatory mechanisms), but essentially, what they really showed is that more diverse environments were associated with generalists with larger metabolisms and larger regulatory systems, with the capability to metabolize a wider range of substrates. It’s not clear that the modularity they observed in the metabolic network was a direct consequence of the environmental ‘variability’. One concern is that they were examining modularity in the metabolism itself and not the regulatory network. They attempt to control for the size of metabolism and its effect on modularity by shrinking linear pathways and cycles, so that the network sizes were all the same, and then calculating modularity on these reduced networks. However, the resulting topology would surely still be determined by the fact that the metabolism was spread across many pathways, merely a larger subset of the potential pathways constrained by the chemistry of the substrates. This suggests a potential link between the lifestyle in unpredictable environments, generalism and modularity even without
regulation.

However, it is far from clear how these ideas would transfer from metabolism (which as I point out, is rather constrained biochemically) to the properties of the genetic architecture. Therefore, this thesis will attempt to address the question of how fluctuating environments affect genetic architecture and evolvability, in the context of the lifestyle of organisms which face unpredictable and variable environments. In the next two sections of this introduction, I will give an overview of the experimental and computational methods used in the thesis.

1.2 Laboratory Evolution in Fluctuating Environments

Both systems biology and evolutionary biology are multidisciplinary fields, incorporating theoretical approaches from mathematics and computer science with experimental work in the laboratory or field. This thesis therefore undertakes both theoretical and experimental approaches to examining genetic architecture in response to fluctuating environments.

Ideally, we would like to observe the evolution of evolvability and modularity under controlled conditions. This would be done by evolving populations of a bacterium (e.g. *E. coli* because of its short generation time) in a range of continuous and varying environments on different timescales. We would then determine the modularity of the regulatory network by measuring the transcriptome using gene deletion studies. Of course, such an experiment would be a very large, or impossible, undertaking.

Nevertheless, laboratory evolution in *E. coli* has been highly successful for examining the evolution of the genome and genetic architecture, both in the long-term experiments running for many thousands of generations (Elena et al., 1996; Papadopoulos et al., 1999; Cooper and Lenski, 2000; Rozen and Lenski, 2000; Cooper et al., 2003)
as well as for shorter periods of evolutionary time (Fong et al., 2005; Ostrowski et al., 2005).

We undertook a laboratory evolution experiment, comparing evolutionary changes to growth rate and fitness of *E. coli* grown for 500 generations in batch cultures with alternating resources. Treatments varied in terms of the carbon source included in the media (lactose or glycerol), including environments where the carbon source was the same every day, and environments which switched carbon sources every 1 or 10 days. We also included a control where both carbon sources were mixed together. The experiment was carried out with both the sequenced *E. coli* B strain REL606 Barrick et al. (2009), from which the long-term evolution experiment in Richard E. Lenski’s lab began, and a highly evolved 20,000 generation isolate from that experiment. In principle, the 20,000 generation strain should respond differently to the varying environment if it had lost its modularity and evolvability due to the relatively unchanging environment of its recent evolutionary past (this is based on the theoretical study of Parter et al. (2007) in which modularity rapidly decays when the environment becomes constant).

1.3 Approaches to Simulating the Emergent Properties of Gene Interactions

In addition to experiments, I use simulations of evolution to investigate genetic architecture in the context of fluctuating environments. There are several reasons for using simulations rather than theoretical mathematical models. Firstly, existing systems biology approaches have already used simulations – usually genetic algorithms as described below – to study evolution in varying environments (e.g. Kashtan et al.
1.3. APPROACHES TO SIMULATING THE EMERGENT PROPERTIES OF GENE INTERACTIONS

2007). We wish to compare our results to theirs as readily as possible. Secondly, and perhaps insurmountably, complex traits and genetic architectures are not always conducive to analytical modeling. For example, the differential equations which represent the kinetics of biochemical networks often cannot themselves be solved analytically, instead requiring numerical solving. If the phenotype represents such a network, as it does in chapter 2 of this thesis, then the evolution of those phenotypes must also be modeled numerically. This makes a simulation of the evolution of such networks a natural approach.

1.3.1 The Genetic Algorithm as Simulation of Evolution

Genetic algorithms were inspired by natural selection as a way to solve difficult mathematical or engineering problems. An example application is finding the combination of kinetic parameters in a model of glucose metabolism to best fit experimentally measured values (Morbiducci et al., 2005). Trying different combinations of parameters by hand is inefficient and one would probably miss a good solution. An obvious way to automate the process is to start with some initial guess, and have a computer adjust the parameter values until a good solution is found. However, this particular problem cannot be optimized in a linear way; trying to incrementally improve an existing solution to the problem in a deterministic manner (known as steepest ascent) will probably get stuck at a suboptimal solution. This can be understood by imagining a landscape of solutions, where the height of the landscape for each possible combination of parameters is proportional to how optimal the solution is. If the landscape is rugged, with hills and valleys, then we would reach a solution at the top of the nearest hill which, while better than our initial guess, may not the best in the whole landscape of solutions.
CHAPTER 1. GENERAL INTRODUCTION

This is familiar to researchers in evolutionary biology as the fitness landscape. Instead of the fitness landscape being a space of solutions to a problem, it is the space of genotypes or phenotypes of the organism. The height of the landscape is the fitness of the organism with the given genotype/phenotype.

Genetic algorithms apply the evolutionary process of hill climbing to genotypes which represent solutions to some problem at hand. Usually, the genome is formulated quite directly in terms of the problem domain. In the example of fitting parameters given above, the genome might be a set of numbers corresponding to the kinetic parameters. The phenotype is how the glucose metabolism model behaves with those parameters, and this behaviour would be compared to measured values to calculate the fitness associated with a given genome. The rest of the genetic algorithms is deceptively simple: start with a random population of such individuals, calculate their fitness, take the better ones and randomly change some of the parameters in their genome to create a new generation. Repeat until a satisfactory solution is found.

Of course, the reality is that there are a number of subtleties in the application of genetic algorithms. For example, genetic algorithms often result in low diversity with the result that it can be difficult to apply genetic algorithms to problems which must solve multiple criteria: in biological terms, a fitness trade-off between traits. Intuitively, one would expect an engineering tool inspired by natural selection to be able to respond well to conflicting demands on fitness, since examples of trade-offs abound in nature (e.g. the fitness cost of antibiotic resistance). However, it can be surprisingly difficult to design a fitness criterion which give desirable results. A whole class of relatively complex genetic algorithms exist to solve such issues, Multiple Objective Genetic Algorithms or MOGA (Fonseca and Fleming, 1993; Murata and Ishibuchi, 1995). These depart even further from biological evolution but are very effective. In MOGA, individuals are ranked; first, according to whether they ‘dominate’
other individuals (they are better at all objectives); second, so as to maintain solutions ranging from the best at one objective, to the best at the other objective, and in-between. Essentially, they force the population to form niches and maintain diversity.

These issues highlight the fact that genetic algorithms were designed to solve complex optimisation problems, not to mimic evolution. To deal with this, the first main chapter of this thesis is a methodology chapter, which identifies and explores some potential problems with transferring genetic algorithms back from the domain of problem-solving to the domain of modeling evolution in a meaningful way.

### 1.3.2 Are Genetic Algorithms the Right Modeling Tool?

There are numerous approaches to genetic algorithms (GA), differing most of all in how individuals are selected to reproduce. There are, of course, also approaches other than genetic algorithms for simulating evolution, such as the use of the Avida platform (Ofria and Wilke, 2004), which we will use and more fully describe in chapter 3. It is necessary to choose the tool which models selection appropriately for the research problem at hand.

One key question is how competition and ecology play their part in the model. Commonly, in evolutionary simulations such as GA, the population size is fixed or given an upper limit. The fitness of each individual is assessed independently (see A.2.4), and the only limited resource is space in the next generation for offspring. There is then only one niche: the population space. This is suitable if the research problem is insensitive to the simplification that there is only one homogeneous niche. If niche formation or co-evolution are to be part of the model, then basic genetic algorithms are unsuitable, since they determine each individual’s fitness independently, rather than as dependent upon each other; in GA, fitness is generally absolute rather than relative. There is therefore no capacity for the complex interaction between genes at the population
level. There are genetic algorithms which have been inspired by the concept of the ecological niche, but care is needed before choosing to use one of these. An example is crowding or fitness sharing (Horn et al., 1994), where individuals are assigned to niche sub-populations based upon a score of phenotypic difference, and their fitness is relative to how crowded the niche is. Individuals won’t compete for shared resources if the genetic algorithm assigns them to different sub-populations due to having different phenotypes. Of course, it is not that niching genetic algorithms are flawed but that they are mainly designed to generate many unique solutions to a complex problem, rather than model niches. In contrast, the *Avida* software platform (Ofria and Wilke, 2004), has been designed for evolutionary and ecological research and directly models resource utilization and community structure in a very general way – from cross-feeding and food webs (Johnson and Wilke, 2004) to spatial heterogeneity. Note that *Avida* is not based upon a genetic algorithm. There are no fixed generations. The large body of publications resulting from *Avida* (reflecting its large number of options) is testament to the fact that evolution can be effectively simulated without needing to use a genetic algorithm.

Therefore, the first step is to decide whether or not GA are the best way simulate evolution for a given problem. When studying resource competition, GA may lose much of the simplicity which makes them attractive. We chose the *Avida* platform over genetic algorithms for chapter 3 partly because we were considering the inclusion of resource competition in the model.

*Avida* has other benefits for modeling, by virtue of being a mature piece of software designed specifically for simulating evolution. These are not necessarily insurmountable in GA, but it would require an effort from the modeling community to develop a GA platform designed for evolutionary and systems biology. For example, *Avida* has a feature-rich analysis mode which can be used to determine the genetic architecture of
1.3. APPROACHES TO SIMULATING THE EMERGENT PROPERTIES OF GENE INTERACTIONS

the genotype-phenotype map. Of course, there is no reason why genetic architecture cannot be investigated with genetic algorithms, and the GA software used in chapter A has a similar system for deleting genes and examining their effect on the phenotype. For this reason the GA framework *sbevolve*, which is presented in chapter 2, was developed to provide a general tool for using GA in systems and evolutionary biology. It is intended that it will be released open source after that chapter’s initial publication. However, *Avida* can still do more than what is possible with even a purpose developed GA system.

1.3.3 The Relationship between Simulated Genomes and Real Genomes

When simulating the evolution of genetic architecture, it is necessary to decide how the gene – the unit of heredity – will be represented in the simulation. A direct approach would be for the digital organisms genetic information to be stored in an equivalent of DNA itself. The sequence can then be operated on with standard genetic algorithm procedures such as mutation, asexual recombination and sexual crossover. However, representing the resulting biochemistry would not be trivial. And unless great care was taken, the resemblance to DNA and RNA would be merely superficial. DNA does not represent the kinetics of biochemical networks directly, the kinetics are expressed through the chemistry of the protein and its three-dimensional, folded form. An attempt to formulate a realistic model of this would be going well beyond the scope of this project.

For this reason, simulations of evolution usually use a digital genome which can readily be interpreted to give the organism’s phenotype. In systems biology studies (e.g. Deckard and Sauro (2004); Kashtan et al. (2007); Crombach and Hogeweg
(2008)) the genome typically represents numerical or logical computations, which can be translated directly into a network of logical operations or perhaps a model of a biochemical network. We used a similar approach in chapter 2 to examine robustness in biochemical networks. However, one limitation with this approach is that the genotype and phenotype become essentially the same thing, if both are in the language of networks, i.e. genes directly refer to each other in the genome. As suggested earlier, the concept of modularly varying environments leading to modular genetic architecture is then somewhat circular, if not trivial. And it is in contrast to the genomes of real organisms, where the genotype is not a blueprint (Pigliucci, 2010), and gene networks emerge through a hierarchy of transcription and translation. While a trivial genotype-phenotype mapping might be convenient when applying GA to an optimizing problem, it is less relevant in modeling evolution and it is our intent is to move away from the trivial blueprint approach and explore the genetic architecture of digital organisms as a network of gene-trait interactions, rather than as an abstract network of directly interacting genes.

The GA developed in chapter A takes a step towards this goal by allowing the network to emerge from the genotype; genes do not directly refer to each other but they form a network by being able to alter the expression as well as be regulated by of a shared set of phenotype-level traits (the details are given more fully in that chapter). In this system, it is impossible to reverse engineer the genotype from the phenotypic network.

Non-trivial genetic architecture was another reason for using Avida in chapter 3 on page 85; the relationship between genotype and phenotype is very rich, and again it is impossible to reverse engineer the genotype. The same phenotype can be observed with a whole range of genetic architectures with different evolutionary dynamics.

Avida has been used to successfully examine the evolution of genetic architecture
1.3. APPROACHES TO SIMULATING THE EMERGENT PROPERTIES OF GENE INTERACTIONS

in previous studies. For example, Misevic et al. (2006) used digital organisms to simulate evolution with and without sexual reproduction (crossover of genetic information). With sexual reproduction, the genome was arranged such that genetic information for the same traits tended to be physically closer together; this was not as strong in the asexual populations. Moreover, there was less overlap of which genomic sites contributed to the same traits in the sexually-reproducing organisms. It is interesting to consider this overlap of sites in terms of overlapping genes, which I discussed at the beginning of this introduction. Physically overlapping genes are interdependent with respect to both mutation and crossover, and this is found in the Avida digital organisms. However, the sites in an Avida genome are the units of heredity, and so could be considered to be individual genes; in this sense there is no possibility of genes overlapping since sites cannot be subdivided. In this view of the Avida genome, Misevic et al. (2006) are observing physical modularity in the mapping between neighbouring genes and traits. The relatedness of neighbouring genes is genetic linkage. In other words, whereas we structurally distinguish between DNA sites and protein-coding genes in natural genomes, there is usually no clear distinction between sites and genes in digital organisms. This is true not just in Avida but genetic algorithms and simulations of evolution generally. This is because we allow each site to code for a functional component of the phenotype. Therefore, we will consider sites and genes to be interchangeable when discussing the simulated evolution of genetic architecture in this thesis.

1.3.4 Overview of the Thesis

This thesis will address the relationship between genetic architecture and fluctuating environments, taking into account the time scale of fluctuation and determining whether lifestyle and generalism is an important factor in genetic architecture and evolvability.
CHAPTER 1. GENERAL INTRODUCTION

Chronologically, we began with the work referred to in chapter 2 on page 51, which used a genetic algorithm to look at network topology and robustness to noise in protein networks. Noise is a very rapid environmental fluctuation. However, this revealed some questions about whether the genetic algorithm was biasing our results. Therefore, we tested various GA techniques to determine their validity as a method of simulating evolution. This resulted in appendix A on page 195.

Chapter 3 on page 85 takes the work forward and examines genetic architecture and evolvability in environments with resources fluctuating over a wide range of timescales. Critically, we also include a control which contains all the resources of the varying environments combined simultaneously. This allows us to test whether it is the fluctuations specifically, rather than the lifestyle associated with the wider range of resources, which explains differences in genetic architecture and evolvability.

Chapter 4 on page 125 addresses the physical aspect of evolvability which has linked tandem nucleotide repeat sequences with localized hyper-mutability at sites known as contingency loci. We examine contingency loci at the level of E. coli strains, looking at strain ecotype, gene conservation and gene function. The results are put in the context of E. coli’s lifestyle: its complex and diverse range of habitats which go far beyond being a host-associated commensal with opportunistic pathogenicity.

In chapter 5 on page 155, we undertake a laboratory evolution experiment to try to see whether an E. coli strain which has been adapting for 20,000 generations in a glucose batch culture (and therefore can be said to have adapted to a narrow range of environments) responds differently to varying environments than its wild type ancestor.

The thesis then concludes with a general discussion. This summarizes the effect of variable environments on genetic architecture and evolvability, in the context of generalism and lifestyle.

All chapters are the sole work of myself, with the exception that the work referred
to in chapter 3 is the result of a project in collaboration with David Springate. The vast majority of this project was a joint effort at each step, although notably David was responsible for calculating and analyzing phenotypic diversity and mutational robustness, while I was responsible for the determination of modularity and measures of evolvability. Nevertheless, all written materials and figures presented in chapter 3 were authored by me, as with other chapters.
Chapter 2

The Evolution of Robustness to Noise in Protein-Protein Oscillators

Abstract

Robustness to transcription-level noise correlated across protein signaling networks has been observed in the KaiC bacterial oscillator and the *E. coli* chemotaxis pathway, as well eukaryotic circadian clocks. These systems differ how readily networks which are robust to such noise can be designed by hand or discovered through computer optimisation. However, these previous studies have focused on hand-designed network models and networks with constraints on the network size, which limits the potential robust network structures found. To address this, we examined robustness in the simulated evolution of oscillating networks, based on protein kinetics, with no constraint on protein network size or complexity. Apart from one negative feedback loop, which is necessary for oscillations, feedback loops did not explain robustness. We found that robust networks tended to be more complex, but that this was not sufficient for robustness and that it was correlated with the time required to evolve the network.
in the simulation. Instead, fine-tuning of the kinetic equation parameters explained robustness to correlated noise.

2.1 Introduction

Robustness, which is the ability to retain a phenotype in the presence of a perturbation, can be used in a number of contexts. One is mutational robustness, which is the ability of organisms to maintain fitness in the presence of mutations during replication. A practical example of mutational robustness is that naturally evolved proteins are very stable with respect to site mutations, whereas de novo designed proteins are not robust in this way (Tavera and Goldstein, 2002). Mutational robustness in biochemical networks could mean either robustness to changes in topology (rewiring/adding/removing an interaction) or robustness to changes in kinetic parameters. Computer simulations to find possible arrangements of the segment polarity network which reproduce the developmental patterns observed in Drosophila larvae led to many solutions which were also highly robust to parameter perturbations. Indeed, the simplest possible network was also very robust to perturbations in both parameters and concentrations (von Dassow et al., 2000; von Dassow and Odell, 2002).

Robustness to fluctuations in concentrations of proteins or metabolites in a biochemical network can have a specific physiological interpretation: robustness to noise. Stochasticity in gene expression can interfere with the organism’s ability to regulate and respond to signals (Swain et al., 2002) and so regulatory systems need to be robust to fluctuations in gene expression. However, the main source of noise is transcription, rather than translation (Kollmann et al., 2005). In bacterial systems, genes for the same function may be placed along one operon, so that the transcriptional noise affects
the concentration of all proteins for that function a correlated way. Such an arrangement is found in the *E. coli* chemotaxis pathway (Kollmann et al., 2005) and the KaiC circadian clock in the cyanobacterium *Synechococcus elongatus* (Clodong et al., 2007). In these cases, it was found that robustness to correlated noise was only achieved for a small number of possible network designs - designs which agreed with experimental observation. Likewise, the *Neurospora* circadian clock is believed to be robust due to interlocking feedback loops (Cheng et al., 2001). These results show the complexity of biochemical networks in nature, and suggest that robustness to correlated noise requires specific network topologies.

However, these studies did not have an open-ended approach to modeling the networks. Network size was constrained and networks were designed manually to be have similarity to measured *in vivo* dynamics and known genes in the network. While this allows the properties of the networks found in nature to be investigated, for example that interlocking feedback loops contribute to the robustness *Neurospora* circadian clock, it cannot be concluded that such topological features are necessary and are the only way to be robust. There are many countless ways in which genes and proteins could interact to produce oscillatory or clock-like behaviour, and it would be impossible to try them all, but the open ended approach of simulating the evolution of such systems with no arbitrary limitations on the number of proteins and interactions, could yield new network topologies which are robust in unanticipated ways. Or the evolved robust networks may have the same ‘solutions’ as nature. We would then gain a new understanding of why the network topologies observed in nature exist as they do.

To explore robustness to correlated noise in an open-ended approach, we examined whether topology was an important aspect of robustness to correlated noise using a genetic algorithm with no constraint on network size, to see whether network size, complexity or topology were implicated in the evolved robustness. We did this for
CHAPTER 2. THE EVOLUTION OF ROBUSTNESS TO NOISE IN PROTEIN-PROTEIN OSCILLATORS

a new system, oscillators in signal transduction networks, as a step towards moving from the evolution of simple biochemical oscillators towards full eukaryotic circadian systems, such as those found in Drosophila and Neurospora. The circadian clocks are similar to those found in mammals and are examples of dynamic biochemical networks of wide interest (Dunlap, 2006).

We found that robust networks tended to be more complex, but that the greater complexity was correlated with the time required to evolve robustness, and that non-robust oscillators could also be complex. Rather, the fine tuning of the parameters of the kinetics was required to achieve correlated robustness. There appeared to be a minimal additional number of interactions in robust networks, but there was no strongly recurring network arrangement or ‘motif’ which was associated with robustness, and robustness did not require additional feedback loops. In summary, the kinetics of the signaling network were at least as important, if not more important, than network topology in determining robustness.

2.2 Methods

We evolved, in silico, two sets of networks: one which has a selection pressure to be robust to correlated noise, the other without that selection pressure. The evolved networks represent oscillating protein networks, similar in their kinetics to eukaryotic circadian clocks such as the one in Neurospora (Smolen et al., 2001). These networks have been observed to be complex in nature, containing positive and negative feedback loops and both protein level and transcriptional regulation. Note that we do not include time delayed transcriptional regulation as often done in Neurospora, but make the simplifying assumption that all interactions take the form of Michaelis-Menten kinetics without any time delays (the kinetics are described below). Time delays found
2.2. METHODS

in full circadian rhythms which can reset according to the length of the day/night variation, but are not necessary to produce the oscillatory behaviour itself. In this chapter, we will be focusing on the robustness of the oscillations to noise in the protein concentrations, rather than to variations in the day/night cycle, and so we will exclude time delays as they greatly complicate the solving of differential equations.

Rather than incorporate random noise into the model, which is computationally expensive, robustness was tested directly in the fitness function, by evaluating the fitness of the oscillator at different protein concentrations. Robustness was tested using a five-fold increase and decrease in the concentration of all proteins simultaneously. Thus, the oscillatory behaviour of the network is examined over an order of magnitude. This approach follows Clodong et al. (2007) who also used a five-fold increase to measure robustness of the KaiC oscillator.

2.2.1 Representation of protein-protein interactions

Networks were based upon the kinetics of phosphorylation/de-phosphorylation of proteins and their interaction. As well as being a realistic context for a biochemical oscillator, it has the advantage that the total concentration of phosphorylated and unphosphorylated proteins are conserved. This makes it easier to compare evolved networks, and to have a selection pressure for high amplitude without leading to unbounded solutions, since the largest possible amplitude is constrained by the total concentration of each protein pair.

The protein-protein interactions has the same form as given by Paladugu et al. Paladugu et al. (2006). In this representation, phosphorylation (and de-phosphorylation) can have allosteric interactions with an activator protein, an inhibitor protein, or both. Proteins cannot have multiple activators or multiple inhibitors, but a protein can be its own effector. The kinetics took a Michaelis-Menten form.
CHAPTER 2. THE EVOLUTION OF ROBUSTNESS TO NOISE IN PROTEIN-PROTEIN OSCILLATORS

This form of protein-protein interaction is akin to signaling pathways in higher eukaryotes, in which the active form of one protein may allosterically affect the phosphorylation of the another protein, but does not transfer its phosphate ion. This is in contrast to a two-component signaling system as found in bacteria such as *E. coli*. Parkinson and Kofoid (1992), in which the sensing component transfers the phosphate to the receiving component.

2.2.2 A framework for evolving networks: *sbevolve*

One of the main problems for evolving biochemical networks is deciding how to code the interactions and kinetics on which the genetic algorithm will operate. Most evolutionary strategies and genetic algorithms are written for genomes represented by sequences of one type of data, such as binary digits or floating point numbers. However, biochemical networks consist of a hybrid mixture of kinetic parameters and information on connections. Kinetic parameters are continuous, whereas the presence or absence of an interaction between two members of a network is discrete. (François and Hakim, 2004; Deckard and Sauro, 2004; Paladugu et al., 2006) used biochemical networks as the genotype itself, with mutations operating on them directly. There were several mutational operators. A similar approach was taken for our simulations. *sbevolve*, a genetic algorithm framework developed specifically to model biochemical networks, implements a similar approach. In *sbevolve*, kinetic parameters are mutated by adjusting their value. In addition, interactions could be added or removed from the network, and new nodes could be added to the network, by a mutation. To confirm that the mutations were not systematically biased towards increasing network complexity, or had a systematic effect on increasing or decreasing kinetic parameters, *sbevolve* was run with a flat fitness function (see section A.3.2). There was no overall bias in the effect of mutations on the network or on the magnitude of kinetic parameters in the
2.2. METHODS

absence of differential fitness.

\textit{sb evolve} was written in C++ to take advantage of that language’s object-oriented
capabilities. Genetic algorithms, mutational operators, objective functions, network
storage, and kinetics equations were implemented as modules, allowing for potential
reuse of \textit{sb evolve} for further research. \textit{sb evolve} is capable of importing and exporting
the networks in the Systems Biology Markup Language (SBML) (Bornstein et al.,
2008). Exporting SBML is useful for analyzing the networks, since it allows them to
be loaded into other systems biology tools such as COPASI (Hoops et al., 2006). A
handful of networks were exported from SBML to COPASI to confirm that the kinetics
of the biochemical networks were correctly implemented by \textit{sb evolve}.

\textit{sb evolve} can use a number of different genetic algorithms, including truncation selec-
tion, tournament selection and multi-objective genetic algorithms (Fonseca and Flem-
ing, 1993; Murata and Ishibuchi, 1995). However, tournament selection with elitism
was used for this particular computational experiment (see section A.2.1). Elitism was
found to be important for good performance because of the computational expense of
simulating oscillations of biochemical networks. To ensure that elitism did not bias
the results, the analysis was performed on the fittest individual when it reached 95%
fitness. The method used to measure fitness is described next.

2.2.3 The fitness function for evolving oscillators

Since the oscillators are evolved using a genetic algorithm, it is necessary to define a
fitness function. An obvious objective is to take the concentration of a protein in the
network and compare it to a pre-determined oscillatory pattern. In other words, we take
the mean squared difference between the actual and desired concentration at various
time points. Such an approach was taken by Paladugu et al. (2006), with some success.
However this will not always evolve an oscillator. This is because there is a strong
local minima in the fitness landscape in which the genetic algorithm gets stuck – that is to have a constant concentration through the mean value of the oscillator (see figure 2.2.1. Paladugu et al. (2006) also tried using a non-linear dynamics technique to detect the oscillator (numerically estimating the eigenvalues to detect a Hopf bifurcation). However, this method did not give them better results.

Figure 2.2.1: Steady state solution instead of oscillator: rather than evolving an oscillator that matches the desired solution (blue line), fitness based on least squares fit may get stuck in the steady state solution through the middle of the oscillator (red line).

Another approach is to use a Fourier transform. However, this would only be easy to use in the case of a sinusoidal oscillator. Any other oscillator would have a complex Fourier spectrum. Essentially, we wish to assess an oscillator’s period and amplitude without being concerned with its shape. Period and amplitude can be calculated directly from the time series. However, a naive approach to assessing period and amplitude also sometimes fails to produce an oscillator. The method which was found to be most successful is as follows. Amplitude and period are considered separately.

Amplitude is the difference between the peak and trough concentration. However, an oscillator with a peak concentration of 2 (units are arbitrary) and a trough concentration of 1 has the same amplitude as an oscillator with a peak of 1,000,002 and a trough of 1,000,001. The difference is the same, yet clearly the second oscillator does
2.2. METHODS

not have the same dynamic range. Hypothetically, any biological system which must respond to the signal of the second oscillator would need to be very, very sensitive. The oscillator would also be subject to more noise. The problem is that this inferior type of oscillator is what is typically evolved in a genetic algorithm to maximize the amplitude.

Instead, we should perhaps try to maximize the ratio between the peak and trough, instead of the amplitude. That way, a peak of 2 and a trough of 1 would be equivalent to a peak of 2,000,000 and a trough of 1,000,000 – the ratio is the same in each case. This is reasonable, but there is a new, practical problem. The same ratio can also be achieved by having a peak of \(2 \times 10^{-15}\) and \(1 \times 10^{-15}\). This is reaching the limit of the computer’s precision, and so any oscillator at such low values are really just numerical errors. Again, attempts to use the peak/trough ratio in the objective function usually led to a poor solution such as this.

Finally, a new way of assessing the amplitude was developed, which was found to be very successful, if less intuitive. The fitness of the amplitude of an oscillator was given by the following formula:

\[
\frac{\text{peak} - \text{trough}}{1 + \text{peak} + \text{trough}}
\]

This is maximal when the peak of the oscillator is high and the trough is close to zero.

The next problem is assessing period (or frequency). The most accurate way to determine period from a time series is to take the same approach a physicist would to measuring the period of a pendulum: start a timer, and count a fixed number of oscillations (say 10), then stop the timer. The period is the total time divided by the number of oscillations counted.
However, this was found to be unsuitable because the genetic algorithm was unable to pass through intermediate solutions where there were just two or three oscillations which died out. Clearly such an intermediate solution is not desired in itself, but the genetic algorithm is likely to fail if this intermediate solution does not score better than no oscillations at all. This can be understood from considering the fitness landscape. The genetic algorithm works by exploring the fitness landscape, gradually working uphill to a maximal solution. If the fitness landscape is flat except very close to the solution, then it may never chance upon that solution.

Instead of fixing the number of oscillations and measuring the time taken to oscillate that number of times, our approach was to fix the time and count the number of oscillations. The accuracy was limited by the number of oscillations counted, but for a computer it is very easy to count a large number of oscillations. If the desired period is 1 second, then the time series should be taken for 100 seconds. This allows good solutions to oscillate 100 times. The accuracy is the reciprocal of that number, i.e. 1%.

Finally, the scores for amplitude and period must be combined. A standard genetic algorithm requires a single fitness value for each organism. This is done by rewarding fitness for the amplitude of every oscillation, up to the number of oscillations expected. Further oscillations instead penalise fitness. This is much more effective than simply adding or multiplying the fitness of the amplitude and period together. The pseudo code for the successful objective function is shown in algorithm 1. Note that the fitness is assessed for every protein in the network, and the best returned.

### 2.2.4 The fitness function for robustness

To assess an oscillator’s robustness to protein concentration, algorithm 2 on page 63 was applied to the fitness function of the oscillator. This finds the protein in the network which oscillates the best at an initial concentration of 1.0, and then finds the fitness for
Algorithm 1 Objective function to evolve oscillators with desired period and maximal amplitude

FUNCTION OscillatorObjective(network, P = proteins in network)
INTEGRATE timeseries for 1 second
FOR EACH p in P:
    delta[p] = concentration[p] - initial[p]
    last[p] = concentration[p]
    peak[p] = last[p]
    trough[p] = last[p]
NEXT p
FOR t = 1 to 1000:
    INTEGRATE timeseries for 1 second
    delta'[p] = concentration[p] - last[p]
    FOR EACH p in P:
        IF delta[p] > 0 and delta'[p] < 0
            peak[p] = concentration
            amp = (peak[p] - trough[p]) / (1 + peak[p] + trough[p])
            IF oscillations[p] < 100
                score[p] = score[p] + amp
            ELSE
                score[p] = score[p] - amp
            END
            oscillations[p] = oscillations[p] + 1
        ELSE IF delta[p] < 0 and delta'[p] > 0
            trough[p] = concentration
            amp = (peak[p] - trough[p]) / (1 + peak[p] + trough[p])
            IF oscillations[p] < 100
                score[p] = score[p] + amp
            ELSE
                score[p] = score[p] - amp
            END
            oscillations[p] = oscillations[p] + 1
        END
    END
NEXT p
NEXT t
RETURN MAX(score)
the same protein at 5-fold lower and 5-fold greater initial concentrations. The overall fitness is the sum of the fitness at the three different initial concentrations. It is worth observing that this is a very different form of robustness to that discussed in appendix A, in which robustness refers to the effect of mutations, rather than the effect of noise in proteins on the phenotype. Since fitness is dependent upon robustness of protein concentration in the present case, robustness is not measured using fitness, since this could lead to a certain amount of circularity. Rather, robustness is measured directly as the amount of variation in frequency or amplitude with protein concentration.

Note that changing the initial concentration for all proteins has the same effect as changing the total concentration of phosphorylated and unphosphorylated pairs, since the concentration of each pair is always conserved. The maximum amplitude of the oscillator is the total concentration of one pair of proteins.

2.2.5 The pruning algorithm

Populations of oscillatory networks were successfully evolved with algorithm 1. However, no penalty was imposed upon the fitness measure for network or genome size. There were two reasons for not imposing such a penalty. Firstly, we did not wish to introduce a possible bias in the results towards topologies associated with small networks. Secondly, attempts to introduce a size penalty were found to either have no effect or to dominate the fitness function, and result in tiny networks which did not oscillate. The reason for this failure to balance a fitness reward (for effective oscillations) and a fitness penalty (for network size) is well known in genetic algorithms, and is the reason for the more complex multiple objective genetic algorithms or MOGA (Murata and Ishibuchi, 1995). However, MOGAs do not generate networks that uniquely balance the rewards and penalties, but rather generate a range of solutions from maximal reward through to minimal penalty, which adds an additional dimension to the subsequent analysis.
Algorithm 2 Objective function to evolve oscillators robust to correlated changes in total concentration

FUNCTION RobustObjective(network)
# set initial concentrations to 1.0:
FOR EACH protein IN network
    initial[protein] = 1.0
NEXT p
# find the protein which has the best fitness:
fitness1, bestprotein = OscillatorObjective(network)
# set initial concentrations to 0.2:
FOR EACH protein IN network
    initial[protein] = 0.2
NEXT protein
# get the fitness for the best protein
fitness2 = OscillatorObjective(network, bestprotein)
# set initial concentrations to 5.0:
FOR EACH protein IN network
    initial[protein] = 5.0
NEXT protein
# get the fitness for the best protein
fitness3 = OscillatorObjective(network, bestprotein)
RETURN fitness1 + fitness2 + fitness3
They are also computationally expensive, and the standard GA already required several months of computational time on a large cluster of computers. Therefore, to reduce the analytical and computational complexity of the study, we opted not to penalise for network size.

Since there is no selection pressure for the networks to be small, they usually evolve to be much larger than necessary. The evolved networks frequently have redundant or non-functional proteins and interactions. This makes it much more difficult to understand how the oscillator functions, or to compare oscillators because they may function similarly but be surrounded by different non-functional interactions. This problem was recognised by Paladugu et al. Paladugu et al. (2006) as well as François & Hakim François and Hakim (2004), but they chose to solve the problem by manually removing interactions until no more could be removed without loss of function.

More efficient and reproducible results could be achieved by automating the process on the computer. It would also be very arduous to manually ‘prune’ thousands of networks, as would be necessary for our study. Instead, we used algorithm 3 on the next page. Because some interactions fine-tune the kinetics, the algorithm removes interactions one at a time to see which reduces the fitness the least. This process is repeated until removing any more interactions would take the fitness more than 5 percent below the original fitness. By always trying to remove the interaction which contributes to fitness the least, the maximal number of interactions can be removed. Many interactions have no effect on fitness, and can be removed in any order. All such trivial interactions are removed. In the pruning of non-trivial interactions, the order of removal can be very important. This could be due to the non-linear effects of interactions on the oscillator’s kinetics, and is suggestive of epistasis. Studying the epistasis in further depth could be interesting, although it is complicated by the way in
2.2. METHODS

which network connections are removed, whether by pruning or mutation. If a protein-protein cycle is removed from the system, then all interactions with those proteins are also removed so that the differential equations are well formed. This means that in general one individual interaction is not removed at a time.

Algorithm 3 Pruning a network to find the functional core

\[
\text{threshold} = \text{objective\_function} (\text{network}) \times 0.95 \\
\text{DO} \\
\text{marked} = \text{none} \\
\text{fitness} = -\infty \\
\text{FOR EACH INTERACTION i} \\
\quad \text{network'} = \text{network} \\
\quad \text{REMOVE i FROM network'} \\
\quad \text{fitness'} = \text{objective\_function} (\text{network'}) \\
\quad \text{IF fitness'} > \text{fitness AND fitness'} > \text{threshold} \\
\quad \quad \text{marked} = i \\
\quad \quad \text{fitness} = \text{fitness'} \\
\text{END} \\
\text{NEXT i} \\
\text{IF marked != none} \\
\quad \text{REMOVE marked FROM network} \\
\text{END} \\
\text{WHILE marked != none}
\]

2.2.6 Population size and mutational parameters

There were 1000 replicate simulations, with a population size of 100 individuals. Each member of the initial population was created individually by adding a random number of connections. The random number of connections was drawn from a geometric distribution using a probability of 0.1. All possible types of new connection had an equal probability of being created (protein-protein cycle, inhibitor or activator) as long as it was possible to create an interaction of the given type.

As mentioned in section 2.2.2, tournament selection with elitism was used. The
tourney size was 2, and top 10 'elite' were preserved to the next generation without
mutation. Mutations were equally likely to change a kinetic parameter, change the way
two proteins were connection, remove a connection, or add a new connection. The net
chance of a mutation was 10% per protein-protein cycle. The population size is small
and mutation rate is high, which is less than ideal. However, attempts to run the
simulation for a population size of 1000 and mutation rate of 1% led to prohibitively
long simulation times. As alluded to previously, the simulations took several months
to complete on a network of >50 modern computers. Reducing the mutation rate
without increasing the population size decreased the performance of the GA itself.
Nevertheless, it is probable that the high mutation rate is partly responsible for the
redundant connections in the networks (see section A.3.4 and Appendix A in general
on the subject of code bloat).

The initial value of kinetic parameters were assigned by drawing a random real
number \( u \) over a uniform range such that taking the kinetic parameter to be \( e^u \) it can
be any positive value between 0.001 and 1000, with an even distribution for numbers
over several orders of magnitude (hence the use of the exponent of \( u \) rather than
just drawing a random number over 0.001 and 1000 as a uniform interval). Kinetic
parameters were mutated by drawing a random number from a normal distribution
with a variance of 0.01. The parameter was then multiplied by this value. This gives a
large probability of a small change and a small probability of a large change, and was
found to be the most effective was of mutating parameters. It is also unbiased between
increasing and decreasing the kinetic parameter.
2.3. Results

2.3.1 Robustness as a Selected Trait

Fitness of the oscillators was based upon high amplitude and having the correct period. This means that an oscillator might have a robust period but its amplitude may not be as robust. We therefore separately compared the responses of amplitude and period to changes in initial concentrations of the proteins. Figure 2.3.1 shows the amplitude and period for oscillators evolved with and without the selection pressure to be robust, at the lower and higher initial concentrations used to evolve robustness. Kolmogorov-Smirnov tests were used throughout, since some of the data are very clearly skewed/non-normal.

For the lower initial concentration of 0.1, the non-robust oscillators have near-zero amplitudes. Also the period is broadly distributed around the ideal periodicity, with a coefficient of variation of 1.1. Also, 32% of the non-robust networks did not oscillate at this lower amplitude. By contrast, the robust oscillators have close to the maximum possible amplitude, and their period is narrowly distributed around the target periodicity, with a coefficient of variation of 0.08. All of the robust oscillators oscillated at this initial concentration. These results demonstrate that robustness is a trait which is selected for in the protein-protein signaling networks.

At the higher initial concentration of 2.5, the differences between the robust and non-robust oscillators are still present but not as strong. 24% of the non-robust oscillators had an amplitude within 1% of the maximum possible amplitude. Only 13% of the non-robust networks did not oscillate at the higher initial concentration. The periodicity had a coefficient of variation of 0.81 for the non-robust networks and 0.11 for the robust networks. This suggests an asymmetry in the robustness at low and high concentrations.
Figure 2.3.1: Robustness of both amplitude and period of the evolved oscillators. These histograms compare oscillators evolved without selection pressure for robustness (blue) and those with selection pressure for robustness (red). Maximum amplitude in each case is twice initial concentration, while the period should always be 1. Units of concentration and time are arbitrary.
2.3. RESULTS

(a) mean fragile 4.66, mean robust 6.21, Wilcoxon test significant (W=272547, p<0.0001)

(b) mean fragile 8.75, mean robust 12.53, Wilcoxon test significant (W=169414, p<0.0001)

(c) mean fragile 2.29, mean robust 3.58, Wilcoxon test significant (W=304466, p<0.0001)

(d) mean fragile 0.66, mean robust 1.51, Wilcoxon test significant (W=306274, p<0.0001)

Figure 2.3.2: Network properties, compared between non-robust (blue) and robust (red) oscillators.
2.3.2 Network Complexity and Robustness

To assess whether robust networks had different topologies to non-robust networks, we examined the number of modifiers (allosteric interactions between proteins), as well as the number of positive and negative feedback loops in the network. A negative feedback loop is essential for a biochemical oscillator, and positive feedback loops have been known to confer robustness to amplitude and tunable period (Tsai et al., 2008). Feedback loops were identified from the signs of the Jacobian matrix over a time course of oscillations. Specifically, the signs of the Jacobian were recorded for the same 100 time points used to calculate fitness. The signs of each element of the Jacobian were then confirmed to be the same over the 100 time points in every case, to ensure that positive and negative feedback loops were well-defined. Loops in the Jacobian matrix were then found using the method of Tiernan (1970). Interactions which form loops are a negative feedback loop if multiplying the signs of the corresponding Jacobian elements together gives a negative sign, else it is a positive feedback loop. As expected, all 2000 non-robust and robust oscillators had at least one negative feedback loop. The results of the comparisons are shown in figure 2.3.2.

From these comparisons, we can see that robustness (for period, amplitude, or both) can be achieved in a substantial number of cases with the minimum number of feedback loops (just one negative feedback loop, and no positive feedback loops). However, there is a statistically significant shift in each of these (Wilcoxon test p<0.0001) with a difference in means of 2.2 species, 3.4 modifiers, 1.2 negative feedback loops and 0.6 positive feedback loops. This suggests that robust oscillators are more complex on average.

It took on average 180 generations to evolve a non-robust oscillator with 95% maximal fitness, whereas robust oscillators took an average of 1700 generations. Section
2.3. RESULTS

A.3.4 shows that genome size can increase with generation time. In this chapter, genome size is identical to the total number of protein-protein pairs as well as modifiers in the network. To see if the greater number of generations required to evolve robust oscillators underlies their greater size and number of interactions in the network, we therefore checked for a correlation between the number of generations to reach 95% fitness and genome size, both before and after pruning to obtain the core network. The correlations are shown in figure 2.3.3. In the case of the non-robust oscillators, generation time and genome size had an R-squared value of 0.20 before pruning and 0.15 after pruning. For robust oscillators, generation time and genome size had an R-squared value of 0.11 before pruning and 0.08 after pruning. In each case, the p-value of the linear model was statistically significant (p<0.0001). To see if the generation time explains the variance better than robustness, we performed an ANOVA.
to examine the comparative amount of variance explained by generation time together with the period and amplitude at the lower and higher initial concentrations. We used the partial eta-squared for this. The results are shown in table 2.1. It is clear that this suggests that the difference in genome size – and therefore network size and number of modifiers – between robust and non-robust oscillators may be due to the increased time required to evolve the networks.

<table>
<thead>
<tr>
<th>Network size (before pruning)</th>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>η²</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>no. of generations</td>
<td>1</td>
<td>284567</td>
<td>284567</td>
<td>841</td>
<td>0.30</td>
<td>&lt;3E-16</td>
</tr>
<tr>
<td>period at lower conc.</td>
<td>1</td>
<td>21312</td>
<td>21312</td>
<td>63</td>
<td>0.03</td>
<td>4E-15</td>
</tr>
<tr>
<td>amplitude at lower conc.</td>
<td>1</td>
<td>35</td>
<td>35</td>
<td>0.1</td>
<td>5.2E-05</td>
<td>0.75</td>
</tr>
<tr>
<td>period at higher conc.</td>
<td>1</td>
<td>1965</td>
<td>1965</td>
<td>5.8</td>
<td>2.9E-03</td>
<td>0.02</td>
</tr>
<tr>
<td>amplitude at higher conc.</td>
<td>1</td>
<td>189</td>
<td>189</td>
<td>0.6</td>
<td>2.8E-04</td>
<td>0.46</td>
</tr>
<tr>
<td>Residuals</td>
<td>1992</td>
<td>674053</td>
<td>338</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network size (after pruning)</th>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>η²</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>no. of generations</td>
<td>1</td>
<td>20738</td>
<td>20738</td>
<td>554</td>
<td>0.22</td>
<td>&lt;3E-16</td>
</tr>
<tr>
<td>period at lower conc.</td>
<td>1</td>
<td>476</td>
<td>476</td>
<td>13</td>
<td>0.01</td>
<td>4E-04</td>
</tr>
<tr>
<td>amplitude at lower conc.</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>0.1</td>
<td>4.0E-05</td>
<td>0.77</td>
</tr>
<tr>
<td>period at higher conc.</td>
<td>1</td>
<td>109</td>
<td>109</td>
<td>2.9</td>
<td>1.5E-03</td>
<td>0.09</td>
</tr>
<tr>
<td>amplitude at higher conc.</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>0.1</td>
<td>4.0E-05</td>
<td>0.77</td>
</tr>
<tr>
<td>Residuals</td>
<td>1986</td>
<td>74315</td>
<td>37</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1: Determination of amount of variance in network size explained by number of generations compared to period and amplitude at concentrations non-robust oscillators were not exposed to. The anova model was: size ~ (no. of generations + period at lower conc + amplitude lower conc + amplitude at higher conc + period at higher conc + amplitude at higher conc). Partial eta squared was used; its calculation requires the sum of squares hence its inclusion in the anova tables.

2.3.3 Feedback loop topology was highly varied

To examine whether feedback loops were associated specifically with robustness, we performed a pair-wise comparison of networks to find equivalent topologies. Two topologies can be considered equivalent if the only differences between the networks are the kinetics parameters and the ordering of feedback loops and proteins in the network.
2.3. RESULTS

Self-interactions are ignored. Topologies were then compared for equivalence (more technically, isomorphism) using the software vlib 2.0 (Cordella et al., 2004).

The topologies of the networks were highly diverse, with at most 39 of the 2000 non-robust and robust networks sharing any one topology. The three most frequent topologies were:

1. The 3-node network with 1 negative feedback loop. 32 non-robust but only 7 robust oscillators had this topology. This is significantly more present amongst non-robust oscillators than robust oscillators (Fisher’s exact test, \( p<0.001 \)).

2. A further 19 robust but only 6 non-robust oscillators were as above but had a positive feedback loop. This is significantly more present amongst robust oscillators (Fisher’s exact test, \( p<0.02 \)).

3. The next most common oscillator was the 2-node network with 1 negative feedback loop. This was not significantly more important in either the robust or non-robust set, unsurprising as there were only 6 such oscillators.

The remaining topologies had too few instances to allow a statistical comparison. Therefore, no single topology of feedback loops was exclusively robust. Given that robust oscillators exist with only a single negative feedback loop, we can conclude that a specific pattern of feedback loops are neither necessary or sufficient for robustness.

2.3.4 What makes an oscillator robust?

To further check whether greater complexity in robust networks was essential for robustness, we compared the simplest robust and non-robust networks. The 1000 non-robust oscillators included networks with only two protein-protein cycles and three modifiers.
However, the simplest networks from the robust set had either two cycles and six modifiers, or three cycles and five modifiers. Therefore, the simplest networks from the robust set are more complex than the simplest networks from the non-robust set.

These simplest networks are shown in 2.3.4. Full details of the kinetics are also provided for the two robust oscillators in appendix B. To see whether these additional modifiers conferred robustness, we examined all networks which had the same number of protein-protein cycles and modifiers. There were six networks with two cycles and six modifiers, of which two were not robust. There were twenty-five networks which had three cycles and five modifiers, of which twenty-two were not robust. This means that the additional modifiers are not sufficient for robustness.

This suggests that the kinetic parameters must also be important for robustness. To explore this, we need to consider the Michaelis-Menten kinetics of networks. Michaelis-Menten kinetics in its basic form is:

\[ v = \frac{k_{\text{cat}} \cdot [S]}{K_m + [S]} \]

Where \( K_m \) is the Michaelis constant, \([S]\) is the concentration of the substrate and \( k_{\text{cat}} \) is the catalytic constant. In both networks, the Michaelis constant \( K_m \) is two orders of magnitude smaller than \( k_{\text{cat}} \) in the forward Michaelis Menten reaction of the protein-protein cycle which produces the main oscillatory output. As long as the Michaelis constant is small compared to the substrate, it can be neglected, so that:

\[ v_{\text{forward}}^A = \frac{k_{\text{cat}}^A \cdot [B] \cdot [A_p]}{K_m^A + [A_p]} \]

Becomes:

\[ v_{\text{forward}}^A = \frac{k_{\text{cat}}^A \cdot [B] \cdot [A_p]}{[A_p]} = k_{\text{cat}}^A \cdot [B] \]
Figure 2.3.4: Examples of some of the simplest oscillators evolved. (a) Simplest non-robust oscillator. (b) and (c) simplest robust oscillators, where (b) has the fewest protein cycles, and (c) has the fewest modifiers. Boxes represent proteins, those inset with a circle are phosphorylated form; solid arrows are reactions, dashed lines ending with circle are activators while those ending with flat line are inhibitors.
CHAPTER 2. THE EVOLUTION OF ROBUSTNESS TO NOISE IN PROTEIN-PROTEIN OSCILLATORS

Figure 2.3.5: Relationship between initial concentration, amplitude and period. There is a greater tendency for oscillators to be robust going to higher concentrations than lower concentrations (blue bars represent non-robust oscillators, red bars represent robust oscillators). Concentration and time have arbitrary units.

(a) median non-robust 0.00094, median robust 0.0017, KS test significant (D = 0.14, p-value < 0.0001)
(b) median non-robust 9, median robust 50, KS test significant (D = 0.41, p-value < 0.0001)
(c) median non-robust 0.030, median robust 0.043, KS test significant (D = 0.25, p-value < 0.0001)
(d) median non-robust 0.1, median robust 0.6, KS test significant (D = 0.37, p-value < 0.0001)
This then scales linearly with total protein concentration, as should the amplitude of the oscillations. However, this is only true if $[A_p]$ is large compared to $K_m$. The asymmetry of the robustness between lower and higher initial concentrations in figure 2.3.1 supports this. To examine this further, we evaluated the periodicity and amplitude of the oscillators at novel initial concentrations, an order of magnitude higher and lower than the initial concentrations that either the robust or non-robust oscillators were exposed to during evolution. The results are shown in figure 2.3.5. This figure shows that the tendency is for robustness to extend readily to higher concentrations but not so readily to lower concentrations, as would be expected if the concentrations are no longer large compared to $K_m$.

If it is true that linear scaling with protein concentration is important, then the derivative of the rate equation (the Jacobian) should be constant, at least for the protein which determines the ‘output’ of the oscillator, since this is the protein which determines the fitness of the oscillator. To test this, we solved the Jacobian matrix for two initial concentrations used and compared them in each case. We did this both at low concentrations and at high concentrations, compared to the starting concentrations for non-robust oscillators. The results are shown in figure 2.3.6. In these results, a value of zero would indicate a perfectly constant Jacobian, and therefore true linear rate equations. First, even at the low concentrations, both robust and non-robust oscillators scale well, with mean changes to the Jacobian of of 0.48% and 0.84% respectively for the low concentrations. Yet the robust oscillators scale more effectively than the non-robust. This was confirmed with a Wilcoxon test ($W = 365,000$ with $p$-value < 0.001). This suggests the robust oscillators are closer to linear than the non-robust oscillators. At the higher concentrations, the constancy of the Jacobian is even closer to zero. However, the robust oscillators are still more closely linear than the non-robust oscillators ($W=540,602$, $p$-value = 0.002). This is what we would expect if the $k_{cat}$ and
Figure 2.3.6: Scaling of the Jacobian. The Jacobian should be constant if the rate equations are approximately linear with correlated changes in the initial concentrations better at higher concentrations. Both non-robust and robust oscillators are approximately linear, but the robust oscillators are more strongly so. Blue points are for non-robust oscillators and red points are for robust oscillators. (a) Initial Concentrations 0.1 and 0.2, (b) initial Concentrations 10 and 20 (arbitrary units)
2.3. RESULTS

$K_m$ kinetic parameters have been tuned in the manner described above, where $k_{cat}$ is larger than $K_m$. However, the strong linearity of even the non-robust oscillators suggests that both robust and non-robust oscillators will have larger $k_{cat}$ values in proportion to $K_m$ values, if less so.

To test whether the $k_{cat}$ values are in indeed larger than $K_m$ values across oscillators, we extracted the parameters and compared them. The result is shown in figure 2.3.7. Here, we divide $K_m$ by $k_{cat}$ and take the log, to show the order of magnitude difference (e.g. -1 would indicate that $k_{cat}$ was 10 times larger than $K_m$). For both robust and non-robust oscillators, $k_{cat}$ values are typically larger than $K_m$ values, with the mean values on this log scale of -1.68 for robust oscillators and -1.28 for non-robust oscillators, a difference in means of 0.41. The difference in $k_{cat}$ and $K_m$ values is therefore somewhat stronger on average in the robust oscillators, confirmed to be significant (see t-test in figure 2.3.7a). Note that this takes into consideration all the protein-protein cycles, showing the mean log ratio of $K_m/k_{cat}$ in each oscillator. We also found the smallest and largest $K_m/k_{cat}$ ratios for each oscillator. While t-tests were significant in both cases, as shown with figures in figure 2.3.7, the effect was large between the robust and non-robust oscillators for the lower $K_m/k_{cat}$ ratios, with a difference in means of 0.77, nearly a whole order of magnitude difference. In comparison, the means of the largest $K_m/k_{cat}$ ratio in the robust and non-robust oscillators differed by only 0.12. This suggests that lower $K_m$ values compared to $k_{cat}$ values are indeed a part of robustness, although it is not necessary for all of the protein-protein cycles in the oscillator to have small $K_m$ values to be robust. The asymmetry between high and low ratios of these constants confirms that this result is not just an artifact of the robust oscillators having more protein-protein cycles on average (which allows for a greater range of kinetic parameters by chance).
Figure 2.3.7: The order of magnitude difference between kinetic parameters $K_m$ and $k_{cat}$ for robust (red) and non-robust (blue) evolved oscillators, shown (a) The average for all protein-protein cycles in the oscillator, (b) The smallest $K_m$ compared to $k_{cat}$ in the oscillator, and (c) The largest $K_m$ compared to $k_{cat}$ in the oscillator. Note that Welch’s t-test is used so as to relax the assumption of equal variances: the degrees of freedom are therefore approximations (under student t-test, degrees of freedom would be 1999 in each case).
2.4 Discussion

Different types of biochemical network vary in the extent to which they can readily support robustness to perturbations, such as correlated noise in gene expression. To explore this in the context of signaling pathways, we created a systems biology-focused genetic algorithm software platform called \textit{sbevolve}. This software platform is designed as a library and is readily extensible for further simulations of the evolution of biochemical networks.

The topologies of protein-protein cycles and feedback loops were highly varied amongst both the non-robust and robust evolved oscillators. For oscillators based upon signaling networks, we found that those which were evolved to be robust to correlated noise tended to be more complex than those which evolved without the robustness constraint (figure 2.3.2). This tendency could be explained by the larger number of generations required for robustness to evolve (figure 2.3.3), and oscillators with two protein-protein pairs and a single negative feedback loop could be robust. Additional feedback loops were neither necessary or sufficient for robustness.

While the simplest robust oscillators had more modifiers interconnecting the proteins than the simplest possible oscillator, the additional modifiers were also not sufficient for robustness. Since topology gave no clear indication of what was required for robustness, we instead tried to understand the robustness of the oscillators from their dynamics. The dynamics of a network requires both the structure and also the kinetic parameters of the rate equations. We found that both robust and non-robust oscillators had rate equations which were close to linear, but that the robust oscillators had more closely linear rate equations. Linearity in the differential equations means that all the terms in the differential equations scale equally with protein concentration. The consequence that changes in protein concentrations due to phosphorylation and
CHAPTER 2. THE EVOLUTION OF ROBUSTNESS TO NOISE IN PROTEIN-PROTEIN OSCILLATORS

dephosphorylation cancel each other out. This therefore confers robustness in the case of correlated changes in protein concentration. As described in the introduction, correlated changes in protein expression is equivalent to the noise in proteins expressed on the same operon, and has been observed in living systems: for example the KaiABC circadian system, where protein expression is correlated due to KaiA promoting KaiB and KaiC on the same operon (Clodong et al., 2007). Linearity in Michaelis-Menten kinetics (figure 2.3.6) can be approximately achieved by having small Michaelis constants ($K_m$) compared to catalytic constants ($k_{cat}$). We found that not all of the protein-protein cycles had small Michaelis constants in robust oscillators, but robust oscillators typically contained more small Michaelis constants than non-robust oscillators (figure 2.3.7). This indicates that a key contribution to robustness comes from the kinetic properties of the oscillators.

Modeling protein-protein networks with michaelis-Menten kinetics had both advantages and disadvantages. From a purely practical point of view, it a very useful that phosphorylation-dephosphorylation cycles of protein pairs are conserved, because this greatly improves the numerical stability of the simulation compared to kinetics with unconserved variables. It is also a simple and a classical model of enzyme kinetics. However, other kinetic forms may be more appropriate due to the comparable concentrations of effectors and substrates (both are proteins, in contrast to enzymatic systems in which the substrate is a metabolite) Chen et al. (2010). Preliminary exploration of Hill kinetics, which are also used to model signal transduction (Qu and Vondriska, 2009), has also been undertaken. While this work is in its very early stages it appears that oscillators form in far fewer generations using Hill equations – perhaps due to the bistability introduced by the sigmoidal form of the kinetics. This preliminary work with Hill equations is suggestive of Goldbeter and Koshland (1981), who demonstrated that a given sensitivity (activity change in response to variation of a signal) can be
2.4. DISCUSSION

achieved by different means in Michaelis-Menten and in Hill kinetics. They found that
the same sensitivity can be achieved by a chain of multiple Michaelis-Menten protein
cycles, or a single protein cycle with Hill kinetics. The Hill kinetics need to have a
sufficiently high Hill coefficient to achieve the same effect – though very high Hill co-
efficients are unrealistic. Given that Michael-Menten kinetics are comparable to Hill
kinetics with a Hill coefficient of 1, it would be interesting to vary the constraints on
the Hill coefficient and compare the topological properties of the resulting oscillators.
These are interesting ideas to pursue, as they could provide further evidence of the
critical interrelatedness of kinetics and topology in achieving robustness in biochemical
networks.

A limitation of our study is the simplifying assumption that proteins always in-
teracted on the same timescale, and that there were no therefore no time delays in
the protein interactions. However, circadian clocks such as those found in Neurospora
crassa also involve transcriptional regulation, which is slower than protein modification
and therefore introduces time delays. Computational methods of solving differential
equations with time delays tend to be less stable and/or require more computing time.
However, the simplification of not including time delays limits the extent to which
we can compare our results to full eukaryotic circadian systems. Evolving circadian
systems which also have the ability to reset the oscillations as the day/night cycle
seasonally changes would be a challenging but natural next step in the open-ended
evolution of dynamic biological networks. Understanding the full range of potential
circadian networks would help to address the critical question of how complex these
networks need to be, in terms of interlocking feedback loops, a problem which has
previously been addressed with hand-crafted models rather than open-ended evolution
(Dunlap, 2006).

The open-ended evolutionary approach has a lot to offer such questions; feedback
loops might not actually be necessary to achieve the circadian clock. To see this, consider that we pruned each oscillator to remove as many connections as was possible before reducing the fitness below 95% of its starting value, so that removing more interactions would have been deleterious in each case. Some of the oscillators contained positive feedback loops, and taking one of these oscillators and removing a positive feedback loop would have made the oscillator less robust or even ‘break’ the oscillator. While it would be correct to conclude that the positive feedback loop was important in that particular network, it would be wrong to conclude that positive feedbacks had an essential role in protein-protein oscillators generally; we also found simpler networks in our open-ended approach which are robust and do not have more than one negative feedback loop. By going beyond hand-designed models and searching for other ways to potentially fulfill the role of a biochemical network we can make stronger conclusions about the constraints that nature has faced in the evolution of gene and protein networks, and why these networks are as complex as they are in nature.

To summaries our results in terms of their general application to the modeling of biochemical systems, we have found that robustness in dynamical systems such as oscillators requires not just network topology and complexity but also the tuning of kinetic parameters. While the simplest robust oscillators tend to be more complex than the simplest non-robust oscillators, the networks we evolved did not employ any specific ‘design’ or motif such as an additional feedback loop. Of course, topology can still play an important role. Kinetic parameters are meaningless without the network structure and reaction stoichiometries to which they relate. In conclusion, the best approach is to consider both potential network structure and kinetics when attempting to understand the properties of a dynamic biochemical network.
Chapter 3

Generalism and Genetic Architecture in Varying and Constant environments

Abstract

Simulations of evolution from the field of systems biology have shown that varying environments can lead to modular metabolic and regulatory networks, increasing evolvability. Meanwhile, evolutionary and ecological theory has long associated greater environmental variability with generalism. However, the systems and evolutionary approaches have not been brought together into a coherent theory. By modeling natural selection under varying environments using the Avida platform, we examined the relationship between generalism, environmental variability and modularity. We found that varying the environment on short timescales resulted in generalism and genomes characterized by less epistasis but higher modularity and evolvability, while slowly varying environments led to specialists with a less modular genetic architecture. Populations dominated by modular
genotypes were shown to be more evolvable. However, we also find that modularity and evolvability can arise with generalism in constant environments with diverse resources. In summary, our work shows that differences in modularity and evolvability can in many cases be explained by niche restriction and organism lifestyle, rather than being specific to temporally fluctuating environments.

3.1 Introduction

An emerging theme in systems biology is the interaction between the properties of biochemical networks and the evolutionary biology of the organism, with a particular emphasis on evolvability. Evolvability can be defined as the ability of an organism to generate novel, heritable phenotypic variation (Kirschner and Gerhart, 1998), although there are a number of other related meanings in the literature. It is also used in the context of heritable variation at the level of the population, such as standing genetic variation, or the rate at which a single trait adapts to a novel environment. Since all of these contribute to how effective organisms are at performing in diverse environments, Pigliucci (2008) recommends acknowledging that evolvability comprises a broad spectrum of effects. While it would be impractical for in vivo studies to measure all forms of evolvability, it is possible to assess different aspects of evolvability in computational models of evolution. However, network approaches to evolvability, such as Kashtan et al. (2007) and Crombach and Hogeweg (2008), have focused on how the properties of a biochemical network may affect the adaptive rate of traits only of the fittest individual. There is therefore scope for a study which integrates the network approach with population biology.

Previous research suggests that evolvability is itself heritable, and is selected for over
3.1. INTRODUCTION

time in populations experiencing environmental fluctuations (Earl and Deem, 2004). Environmental fluctuations have also been the focal point of evolutionary studies in the systems biology community, particularly because both environmental uncertainty and evolvability have been linked to the emergence and role of modularity in simulations of evolving networks (Kashtan et al., 2007; Parter et al., 2008). Modularity is a measure of the structure of a network, where the members of a module are more densely connected to each other than to the members of other modules. Modules often correspond to a particular function. For example, the genes which express the flagella in bacteria interact more with each other than with functionally unrelated systems (Wagner et al., 2007). Intuitively, such modularity improves evolvability because traits can more readily vary independently of each other. However, there remains considerable debate about the generality of this positive association, as well as its underlying causes. The potential link between modularity and evolvability has been discussed for over fifty years in evolutionary biology (Rolian and Willmore, 2009), but the extensive information on biochemical networks being generated in the post-genomic era – such as the metabolic network maps in the KEGG database (Ogata et al., 1999) – has enabled a systems approach to investigating the evolution of evolvability in variable environments (Parter et al., 2007).

While systems biology has the tools to analyse network properties, and to simulate evolution with computational models, understanding the network’s role in population biology requires a comprehensive analysis of both the genotype and phenotype of individuals in a population over evolutionary time. Efficiently recording and analysing a complete ‘fossil record’ of digital organisms is by no means a trivial software engineering effort. The artificial life and evolutionary biology communities responded by creating **Avida**, a complete software environment for studying the evolution of artificial organisms (Ofria and Wilke, 2004). It is a mature application which has been actively
CHAPTER 3. GENERALISM AND GENETIC ARCHITECTURE IN VARYING AND CONSTANT ENVIRONMENTS

maintained and developed since 1993. Because of its long history, it has been widely used to investigate the genetics of evolution (Adami, 2006). This gives it the advantage of being a existing system, rather than a bespoke program written with a specific hypothesis in mind. Avida has a number of features which makes it useful for studying both genetics and networks. It can store extensive details of the population at regular intervals, allowing for analysis at both the population and individual level. It also has a built-in mechanism for performing gene knockouts, allowing the network structure of each artificial organism to be determined (Gerlee et al., 2009).

Using gene knockouts to determine the gene interaction network in Avida allows us to directly explore the genetic architecture of the digital organisms, i.e. the interactions between genotype and phenotype. Considering the properties of the genotype-phenotype map is familiar territory for evolutionary biologists and researchers who use Avida - e.g. Misevic et al. (2006) - but is less common in systems biology studies of the evolution of modularity. In systems biology approaches to the evolution of traits, the phenotype directly represents a network of interacting components, and the genome codes for the network. The genotype is directly translated into its network phenotype, e.g. networks of logic functions in Kashtan et al. (2007). Avida is a more conservative system to study network effects, since we cannot phrase the fitness criteria in terms of a desired phenotypic network. As in nature, networks emerge from genotype-phenotype interactions in Avida organisms rather than as a blueprint in their genome.

There is nevertheless good reason to believe that Avida organisms could support modular networks. Kashtan et al. (2007) suggest that modularity in a network arises most definitely when environments have what they describe as ‘modularly varying goals’. This is a varying environment where the environments share functional requirements, and these functions need to be used in different ways. They give the example of chemotaxis, which could be useful in an environment where the nutrient sources
3.1. INTRODUCTION

change, but where the environments share the need for motility and sensing to reach those nutrients.

This has a counterpart in Avida, where resources in the environment are accessed by successfully completing logical computations which can share calculation steps (Adami, 2006). For example, the function “NOR” represents the logical condition that neither of two statements can be true (see table 3.1). As one might expect, Avida organisms can build this function by applying the logical operation NOT to the logical operation OR to give NOT OR. These functions are in turn built out of simpler instructions. Rewards for resources where the computational results vary but are built up from more basic logical operations allows for the reuse of these sub-components, and therefore the potential for modularity or to facilitate rapid adaptation to a change in environment. On the other hand, sharing sub components could lead to more epistatic interactions between traits, with implications for the ability of those traits to vary in response to a changing environment.

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$A \text{ NOR } B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

Table 3.1: The NOR Boolean logic function

Avida therefore provides an ideal system in which to study the emergence and consequences of modular networks, as well as the mechanism to integrate this systems approach with an in-depth study of population biology. It also raises its own unique challenges. Where chapters A and 2 used genetic algorithms, which have discrete generations, Avida does not. As in natural populations, Avida’s digital organisms (Avidians) do not all have the same lifespan or birth rate. Generation times are therefore not
fixed. This is critical because we are interested in environmental fluctuations on different timescales. The continuous nature of generation time means that the relationship between timescale of environmental change and generation time is not fixed. Why not 'simplify' the study by using genetic algorithms? The key reason is that timescales less than a single generation time cannot be clearly compared to those spanning multiple generations with genetic algorithms. Specifically, genetic algorithms do not in themselves have a measure of time less than a generation, since the generation is the unit of discrete time. In studying the robustness of oscillators in chapter 2, the oscillations over time were continuous but this had no relationship to generation time. Modeling seasonal resource availability and utilisation with differential equations introduces difficult questions as to how to maintain continuity from one generation to another. What happens to the concentrations of network elements when they are added or removed in offspring? The numerical solvers of ordinary differential equations would need to be reset on each generation due to network changes, a questionable approach which would likely introduce some numerical instability. Along with some of the practical problems with genetic algorithms faced in chapter 2, where small network sizes and high mutation rates were needed to reduce computational cost, there is a strong case for studying emerging networks in seasonal environments with the existing Avida system, rather than directly modelling networks and seasonal fitness functions so as to fit into a genetic algorithm.
3.2. Methods

3.2.1 Avida Software

Avida 2.9.0 (Jedi Master) built for 32-bit linux was used for all simulations in this chapter. For the purposes of this experiment, mutation rates and other fine tuned parameters remained at their defaults in Avida.cfg. While examining the effect of mutation rate on the results would be interesting, it is beyond the scope of this particular study and, with the treatments given above, the experiment already explored the evolution of nearly 30 million Avida organisms. Mutations included both substitutions with a probability of 0.0075 and indels with a probability of 0.05 for both insertion and deletion. Full details of the avida configuration are provided in appendix C.

3.2.2 Treatments

We constructed two Avida environments, labeled environment 1 and environment 2. Each environment contained four resources, accessed by performing a given logical computation, with a reward value as given in table 3.2. The value of the resources increases with the complexity of the genotype which is required to express a given trait. The more complex trait resources require more instructions in the genome, and so the reward for performing them has to increase, since a genome which is larger is more expensive to copy in terms of resources. Because of this trade-off between genome size and replication rate traits that are unrewarded in a particular environment carry a direct cost and are therefore likely to be selected against and rapidly decay.

A number of treatments were formulated with these two environments. In two control treatments, environments 1 and environments 2 were held constant, and the organisms in those respective environments never experienced the other environment.
CHAPTER 3. GENERALISM AND GENETIC ARCHITECTURE IN VARYING AND CONSTANT ENVIRONMENTS

<table>
<thead>
<tr>
<th>Environment 1</th>
<th>Value</th>
<th>Environment 2</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOT</td>
<td>1</td>
<td>NAND</td>
<td>1</td>
</tr>
<tr>
<td>AND</td>
<td>2</td>
<td>ORN</td>
<td>2</td>
</tr>
<tr>
<td>OR</td>
<td>3</td>
<td>ANDN</td>
<td>3</td>
</tr>
<tr>
<td>NOR</td>
<td>4</td>
<td>XOR</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 3.2: Avida resources and their value for the two environments

There was one treatment which was also held constant but had a superset of all eight potential resources available. There were then five treatments which alternated between the two environments in a predictable manner, with timescales over five orders of magnitude: the environment alternated either every 1, 10, 100, 1000 or 10,000 updates. Updates are an arbitrary unit of time specific to Avida, in which approximately 30 instructions can be processed. Individual organisms can potentially replicate every 10 updates, although this varies depending upon their fitness. Therefore, the period of fluctuation of each treatment covered a wide range of timescales from less than one generation to many generations. Updates are used as the unit of time, rather than generations, because of the variability in each individual’s reproductive rate.

3.2.3 Populations

There were 100 replicate populations for each treatment, and each population was exposed to its treatment for 100,000 updates. Each population began with the same hand-crafted Avida ancestor which is capable of replication but can perform no calculations and therefore cannot express any of the traits required to thrive in either of the two environments. This handcrafted ancestor is one which is included with the version of Avida named “default classic”. There were 3600 individuals in each population, held constant by replacing one random individual each time another individual divides. All resources are technically infinite, but an organism can gain the reward for expressing a
3.2. METHODS

trait only once. In this way, we emulated the mass action, well-distributed properties of a chemostat.

Following the initial period of 10,000 updates, each population was evolved for a further 1000 updates in one of the evolutionary environments to determine the role of our treatments on subsequent evolvability and diversity. This will be explained further in section 3.2.7.

3.2.4 Population Statistics

Given the complete record of the population history of each replicate and treatment, numerous potential analyses were open to us. *Avida* has a built-in analysis mode which provides several useful population statistics. These include genetic diversity, birth rate, and the traits which each individual in the population expresses. The latter statistic was used to calculate both phenotypic diversity and a measure of specialism.

Phenotypic diversity was calculated using the Shannon-Weaver diversity index: 

\[ H' = - \sum_i S_i (p_i \ln p_i) \]

where \( S \) is the number of distinct phenotypes and \( p_i \) is the relative abundance for each phenotype. The phenotype in our experiment is defined by the distinct set of resources which an individual can use (e.g. two individuals which can utilize NAND and XOR, but cannot utilize any other resources, would have the same phenotype).

Specialism was calculated by arithmetically comparing the number of resources which an individual could utilize from one environment to how many it could utilize in the other. If all its traits were specific to one environment, then it would have a specialism score of 1, whereas if it had an equal number of traits from each environment, it would have a specialism score of 0. Specialism was calculated for each individual, and then averaged over the whole population, to determine the degree to which each population was specialized to each environment.
CHAPTER 3. GENERALISM AND GENETIC ARCHITECTURE IN VARYING AND CONSTANT ENVIRONMENTS

3.2.5 Genotype-Phenotype Mapping

*Avida* can generate a mapping of genotype (instructions coding an organism’s sequence) to phenotype (traits expressed), allowing the genetic architecture of each digital organism to be examined. The genotype-phenotype map is generated by deleting each instruction individually, and determining which traits are still expressed by the organism. Epistasis was then calculated as the number of traits each instruction is required for, divided by the number of instructions and the number of traits the wild type can express. This gave a number between zero and one. For example, if every instruction was essential for every expressed trait, the epistasis score would be one. Likewise, if it were possible for none of the instructions to code for any of the expressed traits, the epistasis score would be zero.

This approach has the limitation that the form of epistasis cannot be determined, i.e. whether cumulative mutations are more or less deleterious with each mutation, as is done in *Avida* studies such as Misevic et al. (2006). To determine directional epistasis, combinations of multiple mutations must be sampled. Also, epistasis is then defined in terms of detrimental changes to fitness, whereas we use epistasis in the context of genes interacting in the genotype-phenotype map. Therefore, epistasis as defined in our study cannot be directly compared to epistasis in mutational studies of asexually reproducing populations such as Elena and Lenski (1997).

3.2.6 Determination of Modularity

Two modularity scores have previously been published for *Avida* genotype-phenotype maps, Functional Modularity and Physical Modularity, and are presented in (Misevic et al., 2006). These measure the overlap between instructions that express the same traits. However, it was found using simulated genomes that Functional Modularity has
3.2. METHODS

an exactly parabolic relationship with epistasis (see figure 3.2.1). We determined this using the following procedure. 100 genotype-phenotype maps were generated randomly generated for every possible combination of genome length from 20 to 100 instructions and number of traits in the phenotype from 3 to 7 traits. This ensured coverage of the range of genotype-phenotype map sizes observed in our simulations. We also ensured coverage of the range of possible densities of the genotype-phenotype map. We did this by choosing a random probability \( p \) for each network to be randomly generated, and using this as the probability with which to decide whether each possible interaction in the genotype-phenotype map was present. \( p \) close to 1 would give a high number of interactions on average and a dense genotype-phenotype map, for example. therefore limiting its use as a means to examine these two genome properties independently.

Functional modularity is maximal at an epistasis score of 0.5 and minimal at either extreme of epistasis (0 or 1). Epistasis contains more information because we can predict functional modularity from epistasis, while the reverse is not true. Therefore we decided not to use functional modularity. The other modularity score in (Misevic et al., 2006) is Physical Modularity, which measures something akin to genetic linkage. This is not relevant to us since recombination/crossover of genetic material is not available as a means of variation in our simulations.

A further key disadvantage of both scores is that they are not based upon the standard method of determining modularity in the systems biology literature (Newman and Girvan (2004)), as used by Kashtan and Alon (2005) as well as Parter et al. (2007) to assess the modularity of biochemical networks. We therefore adopted the Newman and Girvan algorithm for our study. This approach determines the structure of a community or network, dividing it up such that the members of each subdivision interact with each other more strongly than with the members of any other subdivisions. In the context of biological networks, each subdivision is a module. This method fits
the natural definition of modularity as given in the introduction.

The number of potential subdivisions of a network is too large to try every possible combination, so the Newman and Girvan algorithm takes a heuristic approach. Very briefly, it finds the modules by removing the least central connections, i.e. the connection involved in the fewest shortest paths between any two network members, one at a time. After each removal, we calculate the modularity $Q$ based upon how well the network interactions are subdivided, compared to the expected (i.e. average) modularity of all possible networks with the same number of interactions. This is effectively like calculating the modularity before and after disrupting the network structure by rearranging the interactions randomly (however, it is not actually necessary to randomise the network because the expected modularity score can be calculated algebraically). The network with the best modularity score $Q_m$ is chosen.

We generate a suitable network from the genotype-phenotype map by treating each trait as a member of the network. Two traits are connected if an instruction is essential for both traits. The more traits the two instructions share, the stronger that connection is. This is similar to the concept of modular pleiotropy (Wagner et al., 2007), and is a measure of modularity with respect to the ability of the genotype-phenotype map to vary. We find the modules for the dominant genotype of the population, calculating the modularity score $Q_m$ as well as recording how the traits are grouped into modules. Due to the finite number of members in the network, the modularity score $Q_m$ takes discrete values, and can be exactly zero if there is no modularity in the traits. We can therefore assign genotypes as being either modular ($Q_m > 0$) or non-modular ($Q_m = 0$). Note, however, that $Q_m$ is not affected by the trivial case that a trait is entirely separate from all other traits i.e. has no instructions in common with it whatsoever. Since the full separation of traits can be seen as a form of modularity, genotypes can also be compared based upon how many modules (including trivial subdivision) the Newman
3.2. METHODS

and Girvan algorithm detects. Figure 3.2.2 on page 101 shows the relationship between epistasis and modularity in randomized networks, which were generated in the same manner as those for 3.2.1 with the exception that only 1 network was generated for each combination of size of genotype-phenotype map due to the long calculation time required to calculate modularity for genotype-phenotype maps with a high number of connections. A low to intermediate level of epistasis appears to be a necessary – if not sufficient – condition for modularity. While modularity measured in this way is negatively related to epistasis, it does not have the strict relationship with epistasis that functional modularity has.

3.2.7 Evolvability

3.2.7.1 Measuring Evolvability

To assess evolvability, we adapted each evolved population for 1000 updates to the alternative environment from the last one it experienced and then recorded the change in fitness in the population over this interval. For example, if the last environment experienced was environment 1 (ie. the environment experienced at the 10,000 update benchmark), the population would be transferred to environment 2 for 1000 updates. In control runs, the control populations above were were either transferred to the alternative environment or to the same environment that it had experienced throughout the first 10,000 updates.

The entire, often highly diverse, population was transferred in this way to its new environment. We carried out 10 replicates for each population, and averaged the resulting fitness change for each population. By transferring the entire population, rather than only the dominant genotype, both standing genetic variation in the population and evolvability (intrinsic to the evolved genomes within the broader population) could
potentially contribute to the response to environmental change. To discriminate between these ecological and evolutionary responses, the populations were transferred into alternative environments either with or without the inclusion of new mutations. Turning mutation off is a standard option within Avida. With mutation active, genetic changes can contribute to population evolvability, whereas this is not possible with mutation inactivated.

3.2.7.2 Testing the Link between Modularity and Evolvability

Modular and non-modular genotypes evolved in all treatments, which allows us to address the question of whether evolvability is causally linked to modularity by direct comparison. Since we wish to examine evolvability at the population level, we would ideally like to calculate an overall modularity score for all genotypes in each population. However, due to the computation cost of calculating modularity, we instead make the comparison based upon whether the dominant genotype in each population is modular.

3.3 Results

3.3.1 Treatments can be partitioned by Specialism/Generalism

We first tested to see which treatments resulted in the evolution of either specialists or generalists. First, we examined the number of different resources which the dominant genotype of each population could use. This is shown in figure 3.3.1a. Dominant organisms from slowly varying environments, as well the controls which only experience one environment, perform fewer tasks overall. Post-hoc testing of an ANOVA (tasks performed ~ treatment, F=281, df=7, df.D=803, p<0.0001) confirms this, as shown in table 3.3.1b. There are many other pairwise significant results, indicating a high
3.3. RESULTS

degree of parallelism in the number of resources used across replicate populations in a given treatment. Note that we found that many populations in environment 2 did not evolve to utilize the XOR task, the most complicated task in our experiment. This explains why the environment 1 leads to a higher average number of tasks.

The number of sites coding for tasks might be expected to vary with the number of functions which the organism can perform; performing more tasks is likely to require more instructions. We therefore determined the number of sites in the genome which coded one or more tasks (figure 3.3.2a on page 103). As expected there was a strong positive correlation between number of coding sites with the number of tasks performed \( (R^2 = 0.62, p<0.01) \) as shown in figure 3.3.2b. Because of this variation in the number of coding sites, other measures of genetic architecture such as epistasis were normalized so as to be relative to the total number of coding sites. However, a higher proportion tasks and of sites coding for tasks did not translate into a longer genome overall (figure 3.3.3). Consistent with this, we found that fixing the genome length at 100 instructions and repeating the simulations and analyses in this chapter had no effect on the significance or direction of the results.

We also expect that the organisms which perform fewer traits are specialized to their particular environment. We determined the bias of each individual to one environment or the other, and averaged this for the population. In the case of the slowly varying environments (those varying every 1000 and 10000 updates) and the separate environment controls, the populations were, typically, highly specialised to one of the environments (specialism ~ treatment, \( F=572, df=7, df.D=803, p<0.0001 \), figure 3.3.4 on page 105).

In summary, we found that the rapidly varying environments (1, 10 and 100 update timescales) and combined control led to generalists with which could perform many traits from both environments, while the slowly varying environments and separate
Figure 3.2.1: The parabolic relationship found between our measure of epistasis and functional modularity
Figure 3.2.2: Relationship between modularity and epistasis. Data points are shown for random genotype-phenotype networks.
Figure 3.3.1: Number of resources utilized (tasks which can be performed by organism) by treatment, with post-hoc test results. Error bars in figure (a) are 95% confidence intervals. The ANOVA model is: tasks performed ~ treatment. The post-hoc table shows effect sizes in units of number of resources utilized, with significance in parentheses (** p < 0.01, * p < 0.05). Labels 1, 10, 100, 1000 and 10000 represent the alternative environments. ‘both’ is the combined control, and ‘env1’ and ‘env2’ represent the single-environment controls for environment 1 and 2. In summary, significance is largely between both/1/10/100 and 1000/10000/env1/env2.
3.3. RESULTS

Figure 3.3.2: Coding size of genotype, and correlation with tasks performed ($R^2 = 0.62, p<0.01$). Population averages are used in this figure. Error bars in figure (a) are 95% confidence intervals.
Figure 3.3.3: Overall genome size at 100,000 updates (error bars are 95% confidence intervals)
3.3. RESULTS

Figure 3.3.4: Specialism in digital organisms. This is the proportion to which each organism is biased towards tasks in one environment, averaged over the population. Error bars are 95% confidence intervals.
controls led to specialists.

3.3.2 Phenotypic Diversity

Figure 3.3.5 shows that phenotypic diversity by treatment. These results, as with generalism versus specialism, group “both”, “1”, “10” and “100” together (associated with generalism) and also group “1000”, “10,000”, “environment 1” and “environment 2” together (associated with specialism). Phenotypic diversity is higher within the generalist-associated environments. An ANOVA confirmed that the differences between treatments is significant (phenotypic diversity ~ treatment, $F=104$, df=7, df.D=803, $p<0.0001$). Since phenotypic diversity is characterised by unique signatures of phenotypic traits, this is perhaps unsurprising. There are more possible combinations of phenotypes in generalist environments than in specialist environments.

![Figure 3.3.5: Mean phenotypic diversity of evolved populations within each treatment. Error bars are 95% confidence intervals.](image-url)
3.3. RESULTS

3.3.3 Genetic Architecture: Epistasis

Epistatic interactions are a key component of genetic architecture. We therefore tested how epistasis in the dominant organisms from each population varied between treatments. The results are shown in figure 3.3.6a on the next page. These results are once again in two clear groups. Specifically, this figure shows that epistasis is higher overall in the treatments with fewer evolved functions ($F=2216$, df$=7$, df.D$=803$, $p<0.0001$). This difference in epistasis develops early during the 100,000 updates and remains stable (figure 3.3.6b on the following page). The lower number of traits expressed in the genotype-phenotype map of specialists cannot in itself explain the higher level of epistasis, nor can difference in number of coding instructions, since epistasis has been calculated per instruction and per trait. Therefore, we will examine further the properties of the populations before attempting to explain the difference in epistasis.

3.3.4 Genetic Architecture: Mutational Robustness

One aspect of genetic architecture which is implicated in evolvability is mutational robustness, which we define as the average effect of all possible mutations in all possible sites of the genome on fitness. We tested how the environment in which the digital organisms evolved affected the mutational robustness of the dominant genotype at the end of the 100,000 updates of evolution. The results are shown in figure 3.3.7. We found a significant effect of treatment on mutational robustness (mutational robustness $\sim$ treatment, $F=36.3$, df$=7$, df.D$=791$, $p<0.0001$), and post-hoc testing shows that the specialist treatments are more mutationally robust than the generalist treatments (see table 3.3). This may seem surprising at first because Wagner (2005, 2008) suggests that robustness is a contributing component of evolvability; for example, neutral mutations allow sequence diversity to accumulate, and further, that mutational robustness may be
Figure 3.3.6: Density of epistatic interactions in dominant organisms. (a) The distribution of epistasis at the end time points for each treatment. (b) The trajectory of epistasis over the 100,000 generations. Error bars are 95% confidence intervals.
3.3. RESULTS

Figure 3.3.7: Average neutrality of mutations for instructions coding for traits. This is measured across both environments.
a consequence of robustness to environments. Therefore, we might have expected higher mutational robustness in the rapidly fluctuating environments. However, our results have associated mutational robustness not with constant environments per se (the combined generalist environment also has low mutational robustness) but rather with environments which give rise to specialists. The apparent contradictions disappear if we consider specialism and generalism to be the key factors, rather than environmental variability itself.

<table>
<thead>
<tr>
<th></th>
<th>both</th>
<th>1</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>10000</th>
<th>env1</th>
</tr>
</thead>
<tbody>
<tr>
<td>both</td>
<td>0.012</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.018</td>
<td>0.03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.0057</td>
<td>0.018</td>
<td>0.012</td>
<td>0.012</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>0.12 (***)</td>
<td>0.13 (***)</td>
<td>0.1 (***)</td>
<td>0.12 (***)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10000</td>
<td>0.17 (***)</td>
<td>0.18 (***)</td>
<td>0.15 (***)</td>
<td>0.16 (***)</td>
<td>0.046</td>
<td></td>
<td></td>
</tr>
<tr>
<td>env1</td>
<td>0.11 (***)</td>
<td>0.12 (***)</td>
<td>0.094 (***)</td>
<td>0.11 (***)</td>
<td>0.011</td>
<td>0.056 (*)</td>
<td></td>
</tr>
<tr>
<td>env2</td>
<td>0.13 (***)</td>
<td>0.14 (***)</td>
<td>0.11 (***)</td>
<td>0.12 (***)</td>
<td>0.0075</td>
<td>0.038</td>
<td>0.018</td>
</tr>
</tbody>
</table>

Table 3.3: Post-hoc Tukey test of mutational robustness of fitness, after ANOVA (df=7, df.D=791, F=36.3, p < 0.0001). Numbers in table show pairwise differences in the means of mutational robustness, i.e. effect sizes. P values are shown in parentheses; * p < 0.05; ** p < 0.01; *** p < 0.001. Labels 1, 10, 100, 1000 and 10000 represent the alternative environments. ‘both’ is the combined control, and ‘env1’ and ‘env2’ represent the single-environment controls for environment 1 and 2. In summary, significance is largely between both/1/10/100 and 1000/10000/env1/env2.

3.3.5 Modular Pleiotropy

Modular pleiotropy has previously been proposed as a source of evolvability (Hansen, 2003). We determined the modularity of pleiotropy in the genetic architecture of evolved Avida creatures. Noticing that many digital organisms did not evolve modular pleiotropy, we first tested for significant differences between whether the dominant phenotype was modular (Q_m > 0) or non-modular (Q_m = 0) between treatments. The results of this categorization are shown in figure 3.3.8. Pleiotropic modularity was high
Figure 3.3.8: Modularity of traits in dominant genotypes. (a) Proportion of modular and non-modular dominant genotypes. Error bars are 95% confidence intervals calculated using the Pearson-Klopper exact method for binomial proportions. (b) Proportion of dominant genotypes by the amount of subdivisions in the network. This does not coincide exactly with modularity, because trivial subdivisions of the network (for example, a single trait which is unconnected to any other) do not contribute to the modularity score.
CHAPTER 3. GENERALISM AND GENETIC ARCHITECTURE IN VARYING AND CONSTANT ENVIRONMENTS

in the rapidly varying environments and in the combined control environment, but low in the slowly varying treatments and separate environment controls ($\chi^2=181.4$, df = 7, p-value < 0.0001). Excluding non-modular genotypes, no difference in the magnitude of $Q_m$ was found between treatments (Kruskal-Wallis $\chi^2=2.25$, df = 7, p=0.94). Therefore, genetic architectures will be partitioned into modular ($Q_m > 0$) and non-modular ($Q_m = 0$) in further analyses. Part (b) of figure3.3.8 partitions dominant phenotypes not by modularity but by how many subdivisions of the genotype-phenotype map were detected by the Newman and Girvan algorithm, which is perhaps more intuitive than the modularity score itself. Note that there is not an exact correspondence between modularity score being greater than zero and the number of subdivisions being greater than 1; this is because entirely disconnected parts of the genotype-phenotype map do not increase the modularity score.

3.3.6 Evolvability with Changing Environment

Evolvability was measured as a direct effect on fitness after the environment was changed for 1000 updates. Populations whose average fitness had increased more after the perturbation would be considered more evolvable than those where the fitness changed only slightly. Fitness as calculated in Avida is an absolute, dimensionless measure of reproductive efficiency and is comparable across the two environments, because they have the same potential rewards for resource usage. Fitness at the population level could change through selection of fitter individuals as a result of standing genetic variation and through the production of novel genotypes following mutation. To disentangle these possibilities, we perturbed the environment with the mutation rate set to zero as well as with mutations permitted. Any fitness change in the case where mutation was disallowed could only arise due to existing genetic variation in the population.
Figure 3.3.9: Fitness before and after the environment was perturbed by switching populations to a different environment for 1000 updates, with (a) the mutation rate as per the original 100,000 updates, and (b) no mutations (fitness can change only due to standing variation in genotypes/phenotypes). Errors bars give 95% confidence.
CHAPTER 3. GENERALISM AND GENETIC ARCHITECTURE IN VARYING AND CONSTANT ENVIRONMENTS

Figure 3.3.10: Changes in fitness after 1000 updates in the perturbed environment, (a) with and (b) without further mutation being allowed, for each treatment. The perturbed environment is the alternate environment from the one in which the 100,000 updates ended. Fitness is the *Avida* measure of fitness which, when compared, gives the relative reproductive rates of the *Avidians*. Error bars show 95% confidence intervals.
3.3. RESULTS

Figure 3.3.9 shows the fitness of the population in its perturbed environment before the perturbation and after 1000 updates, figure 3.3.10 shows the differences overall. The results are shown both with and without mutation rate set to zero, and from these figures it appears, unsurprisingly, that fitness increases more overall with mutations allowed. A one-tailed paired t-test confirmed this \((t=11.2, \text{df}=810, p<0.0001)\). On average, the fitness increase after 1000 updates was 58% lower with standing variation only. Interestingly there are large differences between the starting fitness and end fitness in the two fixed environments of four resources, but only when mutation was allowed (figure 3.3.9). This suggests that the very low fitness of these populations in the alternate environment, which contains resources that they have not experienced in their evolutionary history, allows for a quick gain in fitness. Though the resulting fitness is not comparable to the generalists which have been adapting to rapid changes in the environment for 100,000 updates, they have had a similar gain in fitness. In contrast, the 1000 and 10,000 populations have barely changed in the 1000 updates in the alternate environment. The change in fitness is lower than that of the fixed environments (figure 3.3.10). However, it is worth highlighting that their fitness starts higher than that of the two fixed environments with four resources in the alternate environment. It appears that, while the 1000 and 10,000 populations are not particularly evolvable, they are robust to the environmental change that they have experienced over the 100,000 updates and have adapted to both environments.

Using Analysis of variance, we found that the magnitude of fitness increases varied with treatment, both in the case of mutation rate remaining unchanged (fitness change ∼ treatment, \(F=16.1, \text{df}=7, \text{df.D}=803, p<0.0001\)) and from standing genetic variation only (fitness change ∼ treatment, \(F=26.2, \text{df}=7, \text{df.D}=803, p<0.0001\)). To explore how the fitness changes varied with treatment, post-hoc pairwise testing by the Tukey method was performed. The pairwise test results are shown in table 3.4. The fitness
changes due to standing variation was greater in populations exposed to more rapidly varying environments. The combined controls also responded with a greater change in fitness, compared to all other treatments except the two most rapidly fluctuating environments. With mutations included as part of the response of the populations, there are several striking results. The populations from the most rapidly fluctuating treatment – those where the resources changed every update – had a greater fitness increase than any others. Also, the slowly varying environments actually respond less effectively than the separate-environment controls. The reason for this is unclear, and is something we would like to explore in future work.

<table>
<thead>
<tr>
<th></th>
<th>both</th>
<th>1</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>10000</th>
<th>env1</th>
</tr>
</thead>
<tbody>
<tr>
<td>both</td>
<td>63</td>
<td>63</td>
<td>63</td>
<td>63</td>
<td>63</td>
<td>63</td>
<td>63</td>
</tr>
<tr>
<td>10</td>
<td>19</td>
<td>43</td>
<td>43</td>
<td>43</td>
<td>43</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>100</td>
<td>12</td>
<td>74</td>
<td>74</td>
<td>74</td>
<td>74</td>
<td>74</td>
<td>74</td>
</tr>
<tr>
<td>1000</td>
<td>37</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>10000</td>
<td>64</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
</tr>
<tr>
<td>env1</td>
<td>0.84</td>
<td>62</td>
<td>62</td>
<td>62</td>
<td>62</td>
<td>62</td>
<td>62</td>
</tr>
<tr>
<td>env2</td>
<td>2.1</td>
<td>66</td>
<td>66</td>
<td>66</td>
<td>66</td>
<td>66</td>
<td>66</td>
</tr>
</tbody>
</table>

(a) fitness change ~ treatment, F=16.1, df=7, df.D=803, p<0.0001

<table>
<thead>
<tr>
<th></th>
<th>both</th>
<th>1</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>10000</th>
<th>env1</th>
</tr>
</thead>
<tbody>
<tr>
<td>both</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
<td>1.7</td>
</tr>
<tr>
<td>100</td>
<td>11</td>
<td>23</td>
<td>23</td>
<td>23</td>
<td>23</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>1000</td>
<td>31</td>
<td>43</td>
<td>43</td>
<td>43</td>
<td>43</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>10000</td>
<td>35</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>env1</td>
<td>43</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>env2</td>
<td>45</td>
<td>57</td>
<td>57</td>
<td>57</td>
<td>57</td>
<td>57</td>
<td>57</td>
</tr>
</tbody>
</table>

(b) fitness change ~ treatment, F=26.2, df=7, df.D=801, p<0.0001

Table 3.4: Post-hoc pairwise testing of changes in fitness after 1000 updates in the perturbed environment, (a) with and (b) without mutation. Numbers show effect size of fitness change (dimensionless unit) and significance levels are shown in parentheses; * p<0.05, ** p<0.01, *** p<0.001
3.3. RESULTS

3.3.7 Putting it All Together: Genetic Architecture and Evolvability

We have presented a wide range of measures of the interactions in the genetic architectures of digital organisms. To bring these together, we performed an analysis of covariance on evolvability to determine which aspects of genetic architecture had contributed to the ability of populations to respond to environmental change. The results of the ANCOVA are shown in table 3.5. To understand the sizes of the results, the partial eta-squared values were calculated for each factor and interaction (the higher the eta-squared value the more variance is explained by the given factor).

As expected, the magnitude of evolvability varies significantly with treatment. This also has the highest partial eta-squared (0.14 with mutations). Number of tasks performed (i.e. generalist or specialist) was the only other also explain significant amounts of the variation both with and without mutation, although the size of the effect is small (partial eta squared was 0.01 with mutations). Modularity, however, did not explain differences in evolvability between treatments, and this was true both with and without mutation. We also tried replacing modularity with the number of subdivisions of the network as a factor in the model, but this did not have any notable effect.

Interestingly, epistasis is only significant without mutation, i.e. standing genetic variation. This may be because epistasis is a compromise in that it can both contribute to evolvability by allowing co-adaptation of complexes of traits – such as the evolutionary switches in Crombach and Hogeweg (2008) – and hinder it by preventing traits from adapting independently. Therefore, under mutation, the effect of epistasis on evolvability may not be very clear.

There were no significant interactions terms for evolvability in the case of standing genetic variation. With mutation allowed, there was a significant interaction term
CHAPTER 3. GENERALISM AND GENETIC ARCHITECTURE IN VARYING AND CONSTANT ENVIRONMENTS

<table>
<thead>
<tr>
<th>Term</th>
<th>df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F</th>
<th>η² Sq</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>treatment</td>
<td>7</td>
<td>993304</td>
<td>141901</td>
<td>16.7</td>
<td>0.14</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>epistasis</td>
<td>1</td>
<td>7607</td>
<td>7607</td>
<td>0.9</td>
<td>0.0012</td>
<td>0.34</td>
</tr>
<tr>
<td>modular</td>
<td>1</td>
<td>4280</td>
<td>4280</td>
<td>0.5</td>
<td>0.00063</td>
<td>0.48</td>
</tr>
<tr>
<td>number of tasks</td>
<td>1</td>
<td>67350</td>
<td>67350</td>
<td>8</td>
<td>0.01</td>
<td>0.005**</td>
</tr>
<tr>
<td>robustness</td>
<td>1</td>
<td>9329</td>
<td>9329</td>
<td>1.1</td>
<td>0.0015</td>
<td>0.29</td>
</tr>
<tr>
<td>phenotypic diversity</td>
<td>1</td>
<td>3757</td>
<td>3757</td>
<td>0.44</td>
<td>0.0006</td>
<td>0.51</td>
</tr>
<tr>
<td>treatment:epistasis</td>
<td>7</td>
<td>79613</td>
<td>11373</td>
<td>1.34</td>
<td>0.012</td>
<td>0.23</td>
</tr>
<tr>
<td>treatment:modular</td>
<td>7</td>
<td>59229</td>
<td>8461</td>
<td>1</td>
<td>0.01</td>
<td>0.43</td>
</tr>
<tr>
<td>treatment:number of tasks</td>
<td>7</td>
<td>189416</td>
<td>27059</td>
<td>3.2</td>
<td>0.03</td>
<td>0.0024**</td>
</tr>
<tr>
<td>treatment:robustness</td>
<td>7</td>
<td>34271</td>
<td>4896</td>
<td>0.58</td>
<td>0.005</td>
<td>0.77</td>
</tr>
<tr>
<td>treatment:phenotypic diversity</td>
<td>7</td>
<td>35972</td>
<td>5139</td>
<td>0.6</td>
<td>0.005</td>
<td>0.75</td>
</tr>
<tr>
<td>Residuals</td>
<td>750</td>
<td>6358277</td>
<td>8478</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) With mutation

<table>
<thead>
<tr>
<th>Term</th>
<th>df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F</th>
<th>η² Sq</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>treatment</td>
<td>7</td>
<td>333286</td>
<td>47612</td>
<td>26</td>
<td>0.2</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>epistasis</td>
<td>1</td>
<td>12264</td>
<td>12264</td>
<td>6.8</td>
<td>0.009</td>
<td>0.0095**</td>
</tr>
<tr>
<td>modular</td>
<td>1</td>
<td>4104</td>
<td>4104</td>
<td>2.3</td>
<td>0.003</td>
<td>0.13</td>
</tr>
<tr>
<td>number of tasks</td>
<td>1</td>
<td>39916</td>
<td>39916</td>
<td>2.2</td>
<td>0.028</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>robustness</td>
<td>1</td>
<td>801</td>
<td>801</td>
<td>0.44</td>
<td>0.0006</td>
<td>0.51</td>
</tr>
<tr>
<td>phenotypic diversity</td>
<td>1</td>
<td>162</td>
<td>162</td>
<td>0.09</td>
<td>0.00011</td>
<td>0.76</td>
</tr>
<tr>
<td>treatment:epistasis</td>
<td>7</td>
<td>9487</td>
<td>1355</td>
<td>0.75</td>
<td>0.007</td>
<td>0.63</td>
</tr>
<tr>
<td>treatment:modular</td>
<td>7</td>
<td>22341</td>
<td>3192</td>
<td>1.76</td>
<td>0.016</td>
<td>0.092</td>
</tr>
<tr>
<td>treatment:number of tasks</td>
<td>7</td>
<td>9168</td>
<td>1310</td>
<td>0.7</td>
<td>0.007</td>
<td>0.65</td>
</tr>
<tr>
<td>treatment:robustness</td>
<td>7</td>
<td>6893</td>
<td>985</td>
<td>0.54</td>
<td>0.005</td>
<td>0.80</td>
</tr>
<tr>
<td>treatment:phenotypic diversity</td>
<td>7</td>
<td>4211</td>
<td>602</td>
<td>0.33</td>
<td>0.003</td>
<td>0.94</td>
</tr>
<tr>
<td>Residuals</td>
<td>750</td>
<td>1360699</td>
<td>1814</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Standing genetic variation only

Table 3.5: Analysis of Covariance for evolvability, including the main properties discussed in this study and their paired interactions (fitness change ~ treatment + epistasis + number of tasks + modular + robustness + diversity + treatment*epistasis + modular*treatment + treatment*number of tasks + robustness*treatment + diversity*treatment). Epistasis, robustness and phenotypic diversity are continuous covariables. Modularity is a factor (Q>0 modular or Q=0 non-modular) (a) model when mutation remains at the same rate for period of perturbation and (b) standing genetic variation effects only.
3.3. RESULTS

between treatment and the number of tasks performed. To explore the relationship between generalism, experimental treatment and evolvability further we plotted evolvability against number of tasks for two groups: the combined controls and rapidly changing environments as one group, the slowly changing environments as the other group in figure 3.3.11. Under mutation, populations with intermediate levels of generalism (numbers of tasks performed) gain fitness more than either specialists or the more complete generalists. This may be because they have a greater scope for fitness increase, since novel mutations could introduce the traits missing for resources in the changed environment. Complete generalists can already perform the tasks for these resources, and so can only gain fitness by becoming more efficient (e.g. deleting sites associated with unrewarded traits).

With only standing genetic variation, the result is slightly different. Here, the trend is for the generalists with the most tasks performed to gain fitness more effectively. Surprisingly, phenotypic diversity – which is calculated by Avida using a Shannon index and was included in this ANCOVA – does not explain any difference in the evolvability, while generalism does. This is in spite of phenotypic diversity being higher in the combined environmental control and rapidly varying environments (referring back to figure 3.3.5). The simplest explanation of the differences in evolvability from standing variation is that populations of generalists already contain phenotypes which are capable of utilizing the resources in the changed environment, allowing them to rapidly take over the population. While this would implicate phenotypic diversity, which is indeed higher amongst the generalists, it could be achieved even with a relatively low diversity, since there are only two environments in our study.
3.4 Discussion

This study has attempted to integrate a network theoretic approach with population biology to understand how fluctuating environments may lead to evolvability, and to understand the relationship between evolvability and modularity, by treating the genotype-phenotype mapping or genetic architecture as a network. We found that digital organisms exposed to rapidly varying environments, as well as the control which combined resources from all environments, evolved to become ecological generalists. The genetic architecture in these digital organisms typically had less densely interacting genotype-phenotype maps (less epistasis, see figure 3.3.6a), were more likely to be modular (figure 3.3.8), and were less mutationally robust (figure 3.3.7). Such properties have previously been associated with evolvability (Kashtan et al., 2007; Crombach and Hogeweg, 2008; Pigliucci, 2008).

It is an important observation that the same properties of modularity and epistasis were seen not just in the rapidly varying environments but also in the control in which the two environments were combined to contain all eight resources. The combined environment has a less restricted niche, compared to the other controls which have only half of the various kinds of resources. The greater diversity of resources led to generalism in much the same way as those populations exposed to rapidly varying environments. Since the commonality between the combined control and the rapidly varying environments is the openness of the niche, modularity and evolvability can in this case be identified as properties of generalists, rather than as a property which arises due to environmental fluctuations over time.

The importance of generalism is confirmed in our study when we look at the relationship between genetic architecture and evolvability. We found that fitness could increase more in generalists (those dominant organisms which performed more tasks).
3.4. DISCUSSION

Relationship between fitness change and tasks performed
mutation rate unchanged
number of tasks performed
fitness change after 1000 updates
−50 0 50 100 150 200 250
both,1,10,100
1000,10000,env 1,env 2

(a)

Interaction of treatment/generalism on evolvability
mutation rate unchanged
treatment
fitness change
0 10 20 30 40 50 60
both,1,10,100
1000,10000,env 1,env 2

(b)

Relationship between fitness change and tasks performed
standing genetic variation only
number of tasks performed
fitness change after 1000 updates
0 10 20 30 40 50 60
both,1,10,100
1000,10000,env 1,env 2

(c)

Interaction of treatment/generalism on evolvability
standing genetic variation only
treatment
fitness change
0 10 20 30 40 50 60
both,1,10,100
1000,10000,env 1,env 2

(d)

Figure 3.3.11: Evolvability, tasks performed, and interaction with treatment
CHAPTER 3. GENERALISM AND GENETIC ARCHITECTURE IN VARYING AND CONSTANT ENVIRONMENTS

when the environment was changed (figure 3.3.11). This was strictly the case for evolvability from standing variation in the population. With mutation, intermediate levels of generalism had the greatest evolvability, perhaps because of the potential increase in fitness by gaining mutations resulting in traits for resources in the changed environment. Critically, greater evolvability was also not specific to the changing environments; indeed, the slowly varying environments had the lowest capacity to increase fitness after environmental change, in the presence of mutations (figure 3.3.10) – although it had adapted to both environments. Moreover, only generalism unambiguously explained the differences in fitness gains between treatments, whether considering only standing genetic variation or also mutation. Modularity, while associated with generalism, did not explain the variation in evolvability significantly.

This has important consequences for research which has previously suggested a strong link between environmental variability, modularity and evolvability (cite). While environmental variability can lead to an increase in modularity, this is not necessarily because modularity is selected for so as to be able to adapt to the environmental change. Nor indeed does modularity necessarily increase evolvability in itself. Where modularity, environmental variability and evolvability have been observed together, it may be that the modularity is simply a byproduct of generalism, and that generalism or the phenotypic diversity associated with it is the reason for greater evolvability. This allows an alternative interpretation of the results of Parter et al. (2007); they used lifestyle as a measure of environmental variability in prokaryotes, and found that species living in more variable environments had more modular metabolic networks. However it may be the extent to which the niche is restricted in the lifestyle, rather than an assumed temporal variability in the environment, which is responsible for the differing levels of modularity they observe. Network size, which indicates generalism, was strongly correlated with environmental variability. While the effect of network size
3.4. DISCUSSION

On modularity was controlled for in their work, this essentially relied upon contracting each network to the same size while maintaining the overall topology of the larger network; the difference in topology could be a result of the network spanning a wider range of metabolic pathways.

This is a key point, because an organism’s metabolic network is a subset of all possible metabolic pathways, the topology of which is constrained by chemistry. Not all metabolic reactions that could exist in a theoretical network are chemically feasible (no enzyme could perform a reaction that does not conserve atomic numbers or charge for example). The chemistry therefore constrains the network such that reactions associated with the metabolism of a certain compound will be more well connected than with unrelated compounds. This is why it is possible for the KEGG pathways to be superimposed upon a reference pathway, and to be curated into distinct modules. In the extreme that only one pathway existed in an organism (an idealised specialist), the metabolic network would have low modularity, while a generalist would be more modular due to incorporating multiple pathways. The ‘metabolism’ of resources simulated by Avida digital organisms is less constrained than in vivo metabolism, and yet even here generalism and network size are key factors in evolvability and modularity.

We have highlighted an important property of generalism in the context of networks properties, as well as providing a caution of assuming that environmental variability is itself responsible for network properties. These insights required both network and ecological properties of the system to be taken into account. This highlights the contributions that a multidisciplinary approach can make to the shared topic of evolvability.
Chapter 4

Contingency Loci, Housekeeping

Genes and the *E. coli* Core Genome

Abstract

Contingency loci, which are simple sequence repeats (SSRs) of nucleotides which confer localized hyper-mutation, have largely been associated with antigen variability in pathogens in the past. However, a study by Guo and Mrázek (2008) found long SSRs in genome regions associated with housekeeping functions, such as amino acid synthesis and DNA repair. Hypermutable sequences would not be expected to be associated with these functions. But genes associated with these functions are not necessarily essential in every environment. This could be an important distinction between housekeeping and essentiality in opportunistic pathogens which experience unpredictable environments, such as *E. coli*. We examine SSRs in 28 sequenced strains of *E. coli* to determine if they are associated with housekeeping genes. In addition to defining housekeeping by function, we also examine SSRs in the context of pathogenicity and essentiality, as well as in terms of the ‘core genome’ – those genes which are found in all
of the strains. We find that SSRs are more common in the pathogenic ecotypes, in non-essential genes and in the non-core genome, but that they are not consistently linked to specific ‘housekeeping’ functions. This confirms the expectation that SSRs are associated with peripheral genes, and highlights a critical distinction between housekeeping function and essentiality.

4.1 Introduction

Contingency loci are sites of localized hyper-mutability in bacterial genomes, commonly associated with rapidly fluctuating responses in pathogens to novel and unpredictable environments (Bayliss et al., 2001). The hyper-mutability associated with contingency loci arises due to the slippage of the DNA replication machinery at simple sequence repeats (Mrázek et al., 2008). Simple sequence repeats (SSR) are tandem sequences of repeating nucleotides in the genome sequence, whether mononucleotide (e.g. AAAAA) or oligonucleotide (e.g. AGAGAG). Contingency loci can be located in reading frames or be associated with promoters. When located in a gene’s coding region the mutations associated with slippage of the polymerase can lead to frame shifts, deactivating or reactivating the gene. A frame shift does not occur with expansion and contraction of triplet repeats, or other multiples of codon size. Tandem repeats of trinucleotides are nevertheless potent sources of hyper-mutability and functional change (Harvey, 1997). Indeed, triplet repeats within the coding regions of genes have been widely studied in human pathology, where their capacity to expand uncontrollably with each replication cycle can lead to hereditary neurological diseases in which greater severity and earlier onset is observed with each successive generation (Wells, 1996). While we will focus on bacteria in this chapter, such diseases provide striking examples of the far-reaching importance of simple repeats in DNA sequences. They also highlight that fact that
the genomic instability caused by sequence repeats can be highly deleterious, and we would not \textit{a priori} expect hyper-mutability to be a general feature of an organism’s genome.

Nevertheless, localized hyper-mutability mediated by tandem repeats is common, and these sites can act to increase polymorphism in the population, potentially generating diversity as a strategy to cope with environmental uncertainty and change. Hypermutability could be beneficial if it is localized to genes where polymorphism is likely to be beneficial rather than those sites where mutations cause loss of fundamental viability. This is why previous studies have implicated contingency loci in surviving unpredictable environments, e.g. evading rapidly changing host immune responses in pathogens as well as during opportunistic infections by commensal bacteria (Bayliss et al., 2001; Moxon et al., 2006). Contingency loci have wider roles in responses to stress which go beyond pathogenicity. For example, in \textit{E. coli}, the trinucleotide TCT appears five times in tandem in the \textit{ahpC} gene, allowing it to readily and reversibly mutate between two different enzymatic activities; the difference of just a single amino acid provides either protein recycling during anaerobic growth or peroxide reduction under oxidative stress (Ritz et al., 2001). The two functions reflect the host-associated facultative lifestyle of \textit{E. coli}, rather than tasks associated with pathogenicity. It is also a key example of a trinucleotide contingency locus with adaptive potential in bacteria, and there are a number of other examples of such loci (Metzgar et al., 2001).

In spite of evidence such as this that contingency loci can operate within the open reading frame, previous studies have largely focused on contingency loci which cause frame shifts, such as tetranucleotide repeats (Bayliss et al., 2005). Mrázek (2006) go as far as distinguishing contingency loci from other SSRs on the basis of whether they cause frame shifts. They also separate out repeats by number of nucleotides in the subunit and by length, and counted the number of repeats which are found compared
to those that might be found by chance in randomized genomes. They further subdivide repeats by length, focusing upon the properties of those that are either short or long. Long SSRs are those where, for a given subunit length (e.g. mononucleotide), the number of repeats detected in the actual genome departs from the expected number of repeats in randomized genomes. For example, in *E. coli* K12, they find that the number of mononucleotide repeats depart from expectations from randomized models at a length of more than 8 base pairs, and are underrepresented. This approach allows them to identify genes with significant repeats within a specific bacterial genome, and look for recurring themes in the functionality of the genes associated with SSR in that genome.

Surprisingly, in a cross-species survey of SSRs, they found that long SSRs defined in this way were significantly associated with both housekeeping genes and genomic regions associated with the antigenic capabilities of host-adapted pathogens (Guo and Mrázek, 2008). Specifically, they find long SSRs associated with replication, recombination and repair, with amino acid transport and metabolism, and with rRNA. This finding appears to contradict the prevailing theory that the role of SSRs is to induce hyper-mutation in genes associated with pathogenicity and novel environmental change, and are negatively selected in genes essential to viability. In addition, the specific functional categories identified as being significantly associated with SSRs are different for each bacterial species examined, making it difficult to generalize their role.

To address these problems, we depart from the species-level identification of contingency loci used in previous studies, and focus instead on a broad examination of SSR within a single species. This offers two substantive advantages over cross-species analyses. First, this allows direct assessment of whether SSR identified in genes of individual strains are shared across the species and are hypervariable, as predicted for contingency loci. Second, this approach allows the distribution of SSR and putative contingency loci to be partitioned between portions of the genome that are core to
4.1. INTRODUCTION

the species, and are present in all strains; those regions which are not found across all strains are likely to encode functions peripheral to the species or provide accessory functions, and might naively be predicted to be enriched for contingency loci. Similarly, this approach enables the assessment of the degree to which SSR and putative contingency loci are found, genome wide, in genes that are essential to organismal performance as opposed to those functions which are dispensable. We will therefore go beyond descriptive functional categorization of genes and examine SSRs in quantifiable terms of gene essentiality, gene conservation and genetic variability.

Contingency loci produce hypermutability, and result in rapid genomic changes. The resulting genetic variability can be on a timescale of generations, and thus repeat-rich genes should be readily comparable between strains. Indeed, mononucleotide repeat loci have been successfully used to discriminate strain phylogeny within *Escherichia coli* (Diamant et al., 2004). Using a within species comparison, we predict that genes containing SSR acting as true contingency loci will be more genetically variable than repeat-poor genes or genes with SSR that are not contingency loci, . We further predict that if SSR containing genes are not essential to all strains they will be less well conserved and thus enriched for SSR or contingency loci mediating peripheral functions.. These predictions have not been explicitly tested in previous bioinformatic studies of contingency loci which have interpreted SSRs from individual genomes as contingency loci, without examination of within species genetic variation, or else have sought contingency loci across species – or even genera (Mrázek et al., 2007).

We tested these hypotheses and explored simple tandem repeats in 28 sequenced strains of *Escherichia coli*. This model organism is ideal, not just because of the many sequenced strains, but because it includes both pathogenic and non-pathogenic strains, which gives our results greater generality by considering contingency loci as a general mechanism to respond to environmental variability and not just the variability
CHAPTER 4. CONTINGENCY LOCI, HOUSEKEEPING GENES AND THE E. COLI CORE GENOME

induced by the immune response. In addition to gene essentiality, we will look at the extent to which SSRs affect those genes which are found in all 28 strains, i.e. the *Escherichia coli* core genome, compared to the non-core genome (i.e. the rest of the pan-genome) as defined by Touchon (2009). The core genome has the advantage of being objective, well-defined and easily determined for sequenced strains of a single species. We have predicted that, in line with expectations posed in the literature, SSR will be found largely in non-core, non-essential, variable genes. This chapter will test these predictions.

4.2 Methods

4.2.1 *E. coli* Sequences

The 28 *E. coli* sequenced strains used in this analysis are shown in table 4.1. The sequences of each strain were obtained from NCBI GenBank (Benson et al., 2010). Only chromosomal sequences were used; plasmids were excluded from the dataset. Not all *E. coli* strains are associated with sequenced plasmids, but all have a single bacterial chromosome. The chromosomal data allows for a direct comparison between all strains.

Ecotype and ecological information on the *Escherichia coli* strains were obtained from the EBI Integr8 database (Kersey et al., 2005). The PEC (Profiling of *E. coli* Chromosome version 4) database (Hashimoto et al., 2005; Ichik Kato and Hashimoto, 2007) was used to assess gene essentiality. All data from NCBI and EBI databases used in this chapter were last retrieved on 11th September 2010.
4.2. METHODS

4.2.2 Detection of Simple Sequence Repeats

While many programs have been written to detect repeats in sequences in various ways – Treangen et al. (2009) lists 35 – the vast majority attempt to detect individual repeats by length and subunit. However, we wish to examine broad patterns of repeat richness across related genomes, rather than in single genome. We would expect SSRs to operate as contingency loci both as frame shifting and with 3-, 6- and 9- nucleotide repeats (Metzgar et al., 2001), and so consider all subunits and repeat lengths together to give a comprehensive measure of repeat richness. To suit this approach, SSRs were identified using a purpose-written C++ application called Santayana. As input, Santayana took the FASTA nucleotide sequence format of an E. coli strain, as well as the gene annotations in its associated GFF annotation file. It identified the number of base nucleotides which were in repeats for each gene and intergenic region. A repeat was defined as any repeating unit length 1-11, repeated at least twice. The minimum allowed repeat length was four nucleotides, thereby excluding codons such as GGG, and also two-nucleotide repeats within a codon, e.g. GGC. Any ambiguous nucleotides were excluded from repeats, i.e. repeats could only contain A, C, G or T. Repeats could overlap, but base pairs in overlapping repeats were not counted twice. The total number of nucleotides which were in repeats in the region, i.e. the repeat coverage, was used as the measure of repeats in a given gene or intergenic region, integrating the lengths of the repeats into the data.

To assess the significance of repeat coverage compared to what may be expected by chance given the sequence composition of each gene, the repeat identification process was bootstrapped. This was done by randomizing the genome 1000 times and each time searching again for repeats. Genes were randomized by shuffling the codons in the open reading frame of each gene, and individual nucleotides in intergenic regions,
similar to the method employed by Karlin and Ghandour (1985). Shuffling preserves the GC content of each region of the genome, as well as any codon bias in the ORF. Therefore, comparing actual repeat coverage to the shuffled repeat coverage should remove these as confounding factors in repeat coverage. This is essential because a high GC content increases the likelihood of detecting repeats containing G and C nucleotides, and likewise for low GC content and repeats containing A and T nucleotides.

A z-score for the repeat coverage of an inter- or intra-genic region could then be calculated. The z-score is the difference between actual and expected number of base pairs in repeats, divided by the standard deviation in the expected number of base pairs in repeats. Unlike repeat coverage, the z-score is dimensionless and is scaled for region size. A score greater than 1.96 indicates over-representation of repeat coverage with 5% significance (i.e. SSR whose presence is unexpected given the sequence of the gene), and a z-score less than -1.96 indicates under-representation of repeat coverage, again with 5% significance (i.e. less SSR coverage than would occur randomly). Note that this method does not attempt to identify the statistical significance of individual repeats, only whether the region of the genome contains significant differences in repeat coverage from what would be expected by chance.

4.2.3 Orthology of Genes and Functional Annotation

Orthologous genes were identified from the NCBI Protein Clusters Database (Klimke et al., 2009), in which sequence similarity between each pair of proteins is determined by BLAST, and then proteins are clustered so that members of a cluster score more highly against each other than against proteins in other clusters. This clustering has the advantage of not requiring either an arbitrary cutoff or phylogenetic analysis. Of the 2634 clusters the NCBI Protein Clusters Database present in the 28 strains of
4.2. METHODS

*E. coli* 683 have either been reviewed and/or curated. This is clearly a more extensive approach than could be taken by performing our own BLAST identification of orthologous genes. The NCBI Protein Clusters Database also has the advantage that functional annotation is available, including COG functional categorization from the NCBI Clusters of Orthologous Groups (Tatusov et al., 2000).

Z-scores of repeat coverage for each clustered gene were obtained by combining the z-scores of the coding regions from each strain in a cluster to determine if that gene had over- or under-represented repeat coverage. To do this, the differences between actual and expected repeat coverage as well as the variances in the expected repeat coverage were added together for each gene. This is roughly equivalent to joining the sequences of the genes together and finding the repeats for the combined sequence. What these procedures result in is a collection of genes that are, across the 28 genomes, over-represented for repeats, under-represented for repeats, and those that contain about as much coverage of repeats as you would expect given their sequence composition. These categories can then be partitioned into various categories of interest.

Core genes were identified as those clusters which were present in all 28 sequenced *E. coli* strains. All genes which were missing in at least one strain were considered non-core, as defined for *E. coli* by Touchon (2009).

4.2.4 Variability of Genes

Genetic variability of *E. coli* genes was determined by counting nucleotide differences in aligned gene sequences and then normalizing by the number of base pair, giving a percent variability. Genes in each cluster were aligned using T-Coffee 8.14 using the default settings (Notredame et al., 2000). This approach to assessing variability does not take into account recombination, where multiple nucleotide changes can result from a single recombination event. Therefore, genes where recombination of fragments
could be detected were eliminated from the analysis of variability. Such gene fragments were detected by applying a gene conversion statistical test, GeneConv (Sawyer, 1989), to the aligned \textit{E. coli} gene clusters. This excluded 4.3\% of \textit{E. coli} genes from the analysis of variability. In total, 2523 \textit{E. coli} gene clusters could be analyzed for genetic variability.

4.3 Results

4.3.1 GC Content

We first checked that the over- or under-representation of repeats was not correlated with GC content, a bias that is known due to more repeats being likely if the GC content is different from 50\% – either high or low (Mrázek 2006). This bias is not of interest, since it can be explained by the greater likelihood of finding a repeat by chance when there are a greater proportion of the same base pairs in the nucleotide sequence (Treangen et al., 2009). However, it was a good way of testing the effectiveness of the procedure I developed for bootstrapping repeat detection. Using a linear regression model it was found that deviation in GC content explains only 0.2\% of variation in the repeat coverage when compared to the bootstrap model, confirming that the bootstrapping had removed GC content as a confounding factor in the analysis. This is in comparison to Mrázek et al. (2007), where GC content was not excluded from many of the repeat types that they attempted to detect (such as long repeats). This demonstrates that our inclusive approach of combining repeats to give an overall measure of repeat coverage is effective in removing artifacts such as GC content which have complicated previous studies.
### 4.3. RESULTS

<table>
<thead>
<tr>
<th>NCBI TaxID</th>
<th>Strain</th>
<th>Ecotype</th>
<th>Environment</th>
<th>Humans Pathogen?</th>
</tr>
</thead>
<tbody>
<tr>
<td>155864</td>
<td>EDL933</td>
<td>EHEC</td>
<td>Entero</td>
<td>yes</td>
</tr>
<tr>
<td>199310</td>
<td>CFT073</td>
<td>UPEC</td>
<td>Urino</td>
<td>yes</td>
</tr>
<tr>
<td>316385</td>
<td>DH10B</td>
<td>Commensal</td>
<td>Research</td>
<td>no</td>
</tr>
<tr>
<td>331111</td>
<td>E24377A</td>
<td>ETEC</td>
<td>Entero</td>
<td>yes</td>
</tr>
<tr>
<td>331112</td>
<td>HS</td>
<td>Commensal</td>
<td>Entero</td>
<td>no</td>
</tr>
<tr>
<td>362663</td>
<td>536</td>
<td>UPEC</td>
<td>Urino</td>
<td>yes</td>
</tr>
<tr>
<td>364106</td>
<td>UTI89</td>
<td>UPEC</td>
<td>Urino</td>
<td>yes</td>
</tr>
<tr>
<td>386585</td>
<td>Sakai</td>
<td>EHEC</td>
<td>Entero</td>
<td>yes</td>
</tr>
<tr>
<td>405955</td>
<td>01K1</td>
<td>APEC</td>
<td>Lung</td>
<td>not known</td>
</tr>
<tr>
<td>409438</td>
<td>SE11</td>
<td>Commensal</td>
<td>Entero</td>
<td>no</td>
</tr>
<tr>
<td>413997</td>
<td>REL606</td>
<td>Commensal</td>
<td>Research</td>
<td>no</td>
</tr>
<tr>
<td>439855</td>
<td>SMS-3-5</td>
<td>SECEC</td>
<td>Metal-toxic coastal</td>
<td>not known</td>
</tr>
<tr>
<td>444450</td>
<td>EC4115</td>
<td>EHEC</td>
<td>Entero</td>
<td>yes</td>
</tr>
<tr>
<td>481805</td>
<td>ATCC 8739</td>
<td>not known</td>
<td>Entero</td>
<td>not known</td>
</tr>
<tr>
<td>511145</td>
<td>K12 MG1655</td>
<td>Commensal</td>
<td>Research</td>
<td>no</td>
</tr>
<tr>
<td>544404</td>
<td>TW14359</td>
<td>EHEC</td>
<td>Entero</td>
<td>yes</td>
</tr>
<tr>
<td>573235</td>
<td>11368</td>
<td>EHEC</td>
<td>Entero</td>
<td>yes</td>
</tr>
<tr>
<td>574521</td>
<td>E2348/69</td>
<td>EPEC</td>
<td>Entero</td>
<td>yes</td>
</tr>
<tr>
<td>585034</td>
<td>IAI1</td>
<td>Commensal</td>
<td>Entero</td>
<td>no</td>
</tr>
<tr>
<td>585035</td>
<td>S88</td>
<td>ExPEC</td>
<td>Cerebrospinal</td>
<td>yes</td>
</tr>
<tr>
<td>585055</td>
<td>55989</td>
<td>EAEC</td>
<td>Entero</td>
<td>yes</td>
</tr>
<tr>
<td>585056</td>
<td>UMN026</td>
<td>ExPEC</td>
<td>Urino</td>
<td>yes</td>
</tr>
<tr>
<td>585057</td>
<td>IAI39</td>
<td>ExPEC</td>
<td>Urino</td>
<td>yes</td>
</tr>
<tr>
<td>585395</td>
<td>12009</td>
<td>EHEC</td>
<td>Entero</td>
<td>yes</td>
</tr>
<tr>
<td>585396</td>
<td>11128</td>
<td>EHEC</td>
<td>Entero</td>
<td>yes</td>
</tr>
<tr>
<td>585397</td>
<td>ED1a</td>
<td>Commensal</td>
<td>Entero</td>
<td>no</td>
</tr>
<tr>
<td>595496</td>
<td>BW2952</td>
<td>Commensal</td>
<td>Research</td>
<td>no</td>
</tr>
<tr>
<td>866768</td>
<td>BL21 DE3</td>
<td>Commensal</td>
<td>Research</td>
<td>no</td>
</tr>
</tbody>
</table>

Table 4.1: Sequenced *E. coli* strains and their ecotypes
4.3.2 Repeats across the Genome and Ecotypes

*E. coli* is an opportunistic pathogen with both pathogenic and non-pathogenic sequenced isolates. SSRs have previously been associated with pathogenicity and so it is useful to examine repeat coverage by ecotype. We sought to generalize SSR and to assess differences as a function of strain history and lifestyle. To do this, we performed a generalised linear model on the repeat coverage for each ecotype (repeat coverage $\sim$ ecotype, Null Deviance:=457.2 and df=28; Residual Deviance 277.7 and df=25). Repeat over- and under-representation for each ecotype was characterized using a z-score calculated from all genes and their intragenic regions combined. These z-scores are summarized in figure 4.3.1.

![](image.png)

Figure 4.3.1: Over-representation of repeats by ecotype. Pathogenic *E. coli* strains have higher over-representation of repeat sequences compared to commensal strains. Error bars show 95% confidence intervals.

The generalized linear model showed that intestinal pathogenic strains – such as
4.3. RESULTS

enterohemorrhagic (EHEC) and enteropathogenic (EPEC) ecotypes—possess genomes with more significantly over-represented repeats than commensal strains ($t=3.9$, $p << 0.001$), as do extra-intestinal pathogens ($t=2.1$, $p=0.044$). Extra-intestinal pathogenic bacteria include those isolated from the lungs or urinary tract (UPEC). Overall, we can conclude that pathogens have a higher repeat coverage than non-pathogens. This is consistent with predictions.

It is important to note that there is no significant difference in the proportion of the genome which is non-core in commensal, EHEC and ExPEC ecotypes (figure 4.3.2). This suggests that the greater presence of repeats in the pathogenic ecotypes is not in itself explained by having a greater number of non-core genes which themselves contain more repeats. This is surprising in light of the fact that many of the virulence factors for pathogenic E. coli are presumed to reside upon mobile virulence cassettes (Hacker et al., 2003), which would necessarily be accessory genes by our criterion.

4.3.3 Repeats and the Genetic Variability of Genes

If the SSRs detected are largely contingency loci (sites of localized hyper-mutability), then one would predict that the SSRs and the genomic regions containing those SSRs would be more genetically variable than non-SSR regions. We tested this by comparing genetic variability of those genes with significant over-representation of repeats (a z-score greater than 1.96), non-significant repeat coverage, and significant under-representation of repeats (a z-score less than -1.96). The results are shown in figure 4.3.3, with an ANOVA in table 4.2. This shows that the scaled genetic variability of genes with SSRs differs between core and non-core genes, as well as by whether repeats are over or under represented. Specifically, the core genes are more genetically conserved than accessory genes, and there is higher genetic variability amongst genes over-represented for repeats than under-represented for them. Consistent with this,
Figure 4.3.2: Ecotype and proportion of genome which is non-core. Error bars show 95% confidence intervals.
Figure 4.3.3: Variability of core and non-core genes, broken down into genes with and without significant fractions of repeats. Key: ns = repeats not significant in genes, over = over-representation of repeats in genes, under = under-representation of repeats in the genes. Error bars are 95% confidence intervals.
in pair-wise posthoc tests, we found that the significant differences in variability were between genes with over- and under-represented repeat coverage ($p < 0.0004$), as well as between genes with under- and non-significant repeat coverage ($p < 0.0004$).

<table>
<thead>
<tr>
<th></th>
<th>df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>P value</th>
</tr>
</thead>
<tbody>
<tr>
<td>core vs non-core</td>
<td>1</td>
<td>3147</td>
<td>3147</td>
<td>102</td>
<td>$&lt; 0.001$ ***</td>
</tr>
<tr>
<td>over/ns/under</td>
<td>2</td>
<td>414</td>
<td>207</td>
<td>6.7</td>
<td>0.0012 **</td>
</tr>
<tr>
<td>core vs non-core * over/ns/under</td>
<td>2</td>
<td>68</td>
<td>68</td>
<td>1.1</td>
<td>0.33</td>
</tr>
<tr>
<td>residuals</td>
<td>2412</td>
<td>74375</td>
<td>31</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: ANOVA of genetic variability (variability $\sim$ repeat coverage significance * core or non-core); core vs non-core and repeat coverage significance (over-represented, not significant, under-represented).

One issue with these comparisons is that the data is not independent, but related by phylogeny at the level of *E. coli* strains. However, the data being examined is the sequence repeats in the genome sequences themselves – which due to their hyper-variability can be used to construct the phylogeny at the strain level (Diamant et al., 2004). Various attempts to remove the phylogenetic relatedness in the data of SSRs (e.g. using independent contrasts) were therefore found to entirely remove the signal.

### 4.3.4 Repeats and Core Genes

The ANOVA in 4.2 shows that core and non-core genes have differing levels of repeats. To explore this further, we broke genes down into core and non-core and into whether they were significantly over- or under-represented with repeats. The results are shown in contingency table 4.3. A Pearson’s $\chi^2$ is in agreement with the result from the ANOVA there is a significant difference between core and non-core genes with respect to repeats. We can see in the table that the difference arises from a large number of core genes with an under-representation of repeats. Therefore, genes conserved within all *E. coli* strains have less repeat coverage than genes specific to one or more strains.
Table 4.3: Cross tabulation of core/non-core genes and repeat significance. Pearson’s $\chi^2 = 27.9$, df=1, p-value<0.0001
4.3.5 Repeats and Functional Categorization

To partition SSR by functional categories of genes, we examine the specific refined COG categories in terms of both repeat coverage and the content of core/non-core genes.

We first test how functional categories of genes vary as a function of whether or not they are core/non-core, excluding consideration of SSRs, as shown in 4.3.4. Functional categories are ranked in descending order of the proportion of core genes within that category. Fisher’s exact tests were used to compare the results to the overall distribution of core and non-core genes in the *E. coli* pangenome. Functions with significantly more core genes than average were cell cycle control, the transport and metabolism of coenzymes, amino acids and nucleotides, the post-translational modification and chaperoning of proteins, and cell wall/membrane synthesis. Many of these can readily be described as housekeeping functions. Conversely, the functional categories with more non-core genes notably includes motility, which varies greatly in *E. coli* depending upon both strain and its requirements in a given environment. Carbohydrate metabolism/transport is also included and could be argued to have important differences between ecotypes (Touchon, 2009). There are several substrates whose utilization is exclusive to pathogenic strains, such as N-acetylgalactosamine and deoxyribose. Metabolism is also implicated in pH homeostasis, which requires differential regulation in uropathogens due to the pH of urine (Brzuszkiewicz et al., 2006).

This analysis suggests that lifestyle and ecotype are important factors determining which functional categories contain core genes or a fraction of genes that are non-core. We next asked how SSR are partitioned across functional categories in core and non-core genes. These results, along with details of the tests, are shown in figure 4.3.5.
**Figure 4.3.4:** Functional categories organized by descending proportion of genes that are core. *n* values show number of genes in the category. Fisher’s exact tests were used to compare each category to the *E. coli* pangenome (ALL CATEGORIES). *p*<0.05; **p**<0.01; ***p***<0.001.

Functional categories are ranked by the proportion of genes with over-represented repeats. Fisher’s exact tests were used to compare the results to the overall distribution of core and non-core genes in the functionally annotated *E. coli* pangenome, and significance is marked on the right of each category label. We found that over-represented or under-represented SSR regions are not as strongly associated with functional category as they are with the core/non-core division overall. For coding regions, the only significant categories with a greater proportion of repeats than the pan-genome average (all categories in the figure) were the Unspecified and Function unknown, suggesting that
there are a number of genes that contain contingency loci but these are either variable in function or poorly understood/poorly conserved. Carbohydrate metabolism is the sole category with significantly fewer genes with SSRs; this category was found to contain largely non-core genes.

Figure 4.3.5: Functional categories organized by descending proportion of genes that have significantly over-represented repeat coverage, subdivided by core/non-core and significance of repeats. \( n \) values show number of genes in category. Fisher’s exact tests were used to compare the proportion of over- and under-represented repeats in each category to the E. coli pangenome (ALL CATEGORIES); significance levels for these tests are shown on the right of the category labels. Significance levels on the left correspond with the tests in table 4.3.7. * \( p<0.05 \); ** \( p<0.01 \); *** \( p<0.001 \).

Upstream non-coding regions were examined in the same way (figure 4.3.6). The only specific significant category in upstream regions with a greater over-representation
4.3. RESULTS

of repeats than average was Nucleotide Transport and Metabolism, with 47% of genes being core and having upstream regions with over-represented SSR. It is surprising to find a category which is both over-represented for repeats and had significantly more core genes than the E. coli pan-genome overall.

Figure 4.3.6: Functional categories organized by descending proportion of upstream regions for genes in that category that have significantly over-represented repeat coverage, subdivided by core/non-core and significance of repeats. n values show number of genes in category. Fisher’s exact tests were used to compare the proportion of over- and under-represented repeats in each category to the E. coli pan-genome (ALL CATEGORIES); significance levels for these tests are shown on the right of the category labels. Significance levels on the left correspond with the tests in table 4.3.7. * p<0.05; ** p<0.01; *** p<0.001.

Therefore, to examine whether these SSRs are associated with core genes in this category, we examined the relationship between core/non-core genes and over/under-
### CHAPTER 4. CONTINGENCY LOCI, HOUSEKEEPING GENES AND THE *E. COLI* CORE GENOME

**Figure 4.3.7:** Fisher exact tests of effects between core and non-core and over- and under-represented repeats for functional categories in coding regions (a) and upstream regions (b). Significant results are highlighted in bold. Significance levels are also marked on the left of the category labels in figure 4.3.5 and 4.3.6 for ease of cross-referencing.

#### (a)

<table>
<thead>
<tr>
<th>Category</th>
<th>Core over</th>
<th>Core under</th>
<th>Non-core over</th>
<th>Non-core under</th>
<th>Odds ratio</th>
<th>p value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amino acid transport and metabolism</td>
<td>28</td>
<td>15</td>
<td>77</td>
<td>25</td>
<td>0.61</td>
<td>0.23</td>
</tr>
<tr>
<td>Carbohydrate transport and metabolism</td>
<td>16</td>
<td>22</td>
<td>55</td>
<td>38</td>
<td>0.51</td>
<td>0.09</td>
</tr>
<tr>
<td>Cell cycle control, mitosis and meiosis</td>
<td>4</td>
<td>1</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0.28</td>
</tr>
<tr>
<td>Cell motility</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>4.07</td>
<td>0.18</td>
</tr>
<tr>
<td>Cell wall/membrane biogenesis</td>
<td>22</td>
<td>10</td>
<td>54</td>
<td>14</td>
<td>0.57</td>
<td>0.32</td>
</tr>
<tr>
<td>Coenzyme transport and metabolism</td>
<td>18</td>
<td>7</td>
<td>36</td>
<td>6</td>
<td>0.43</td>
<td>0.21</td>
</tr>
<tr>
<td>Defense mechanisms</td>
<td>5</td>
<td>2</td>
<td>9</td>
<td>3</td>
<td>0.84</td>
<td>1</td>
</tr>
<tr>
<td>Energy production and conversion</td>
<td>33</td>
<td>17</td>
<td>54</td>
<td>21</td>
<td>0.76</td>
<td>0.55</td>
</tr>
<tr>
<td>Function unknown</td>
<td>39</td>
<td>20</td>
<td>34</td>
<td>23</td>
<td>1.32</td>
<td>0.56</td>
</tr>
<tr>
<td>General function prediction only</td>
<td>30</td>
<td>21</td>
<td>53</td>
<td>15</td>
<td>0.41</td>
<td>0.03</td>
</tr>
<tr>
<td>Inorganic ion transport and metabolism</td>
<td>17</td>
<td>19</td>
<td>38</td>
<td>16</td>
<td>0.38</td>
<td>0.05</td>
</tr>
<tr>
<td>Intracellular trafficking, secretion, and vesicular transport</td>
<td>6</td>
<td>3</td>
<td>10</td>
<td>3</td>
<td>0.61</td>
<td>0.66</td>
</tr>
<tr>
<td>Lipid transport and metabolism</td>
<td>8</td>
<td>6</td>
<td>17</td>
<td>7</td>
<td>0.58</td>
<td>0.49</td>
</tr>
<tr>
<td>Multifunctional</td>
<td>59</td>
<td>50</td>
<td>119</td>
<td>54</td>
<td>0.54</td>
<td>0.02</td>
</tr>
<tr>
<td>Nucleotide transport and metabolism</td>
<td>10</td>
<td>3</td>
<td>27</td>
<td>7</td>
<td>0.87</td>
<td>1</td>
</tr>
<tr>
<td>Protein modification/turndover/chaperones</td>
<td>19</td>
<td>2</td>
<td>26</td>
<td>8</td>
<td>2.87</td>
<td>0.29</td>
</tr>
<tr>
<td>Replication, recombination and repair</td>
<td>22</td>
<td>11</td>
<td>32</td>
<td>14</td>
<td>0.88</td>
<td>0.81</td>
</tr>
<tr>
<td>Secondary metabolism/transport</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>1.62</td>
<td>1</td>
</tr>
<tr>
<td>Signal transduction mechanisms</td>
<td>9</td>
<td>9</td>
<td>20</td>
<td>8</td>
<td>0.41</td>
<td>0.21</td>
</tr>
<tr>
<td>Transcription</td>
<td>26</td>
<td>22</td>
<td>30</td>
<td>14</td>
<td>0.56</td>
<td>0.02</td>
</tr>
<tr>
<td>Translation, ribosomal structure and biogenesis</td>
<td>17</td>
<td>14</td>
<td>69</td>
<td>16</td>
<td>0.29</td>
<td>0.01</td>
</tr>
<tr>
<td>Unspecified</td>
<td>15</td>
<td>33</td>
<td>8</td>
<td>23</td>
<td>1.3</td>
<td>0.8</td>
</tr>
</tbody>
</table>

#### (b)

<table>
<thead>
<tr>
<th>Category</th>
<th>Core over</th>
<th>Core under</th>
<th>Non-core over</th>
<th>Non-core under</th>
<th>Odds ratio</th>
<th>p value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amino acid transport and metabolism</td>
<td>64</td>
<td>24</td>
<td>29</td>
<td>9</td>
<td>0.83</td>
<td>0.83</td>
</tr>
<tr>
<td>Carbohydrate transport and metabolism</td>
<td>53</td>
<td>27</td>
<td>19</td>
<td>12</td>
<td>1.24</td>
<td>0.66</td>
</tr>
<tr>
<td>Cell cycle control, mitosis and meiosis</td>
<td>11</td>
<td>2</td>
<td>8</td>
<td>1</td>
<td>0.7</td>
<td>1</td>
</tr>
<tr>
<td>Cell motility</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>0.41</td>
<td>1</td>
</tr>
<tr>
<td>Cell wall/membrane biogenesis</td>
<td>32</td>
<td>9</td>
<td>32</td>
<td>7</td>
<td>0.78</td>
<td>0.78</td>
</tr>
<tr>
<td>Coenzyme transport and metabolism</td>
<td>29</td>
<td>4</td>
<td>14</td>
<td>3</td>
<td>1.54</td>
<td>0.68</td>
</tr>
<tr>
<td>Defense mechanisms</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>1.19</td>
<td>1</td>
</tr>
<tr>
<td>Energy production and conversion</td>
<td>36</td>
<td>24</td>
<td>24</td>
<td>7</td>
<td>0.44</td>
<td>0.11</td>
</tr>
<tr>
<td>Function unknown</td>
<td>42</td>
<td>28</td>
<td>26</td>
<td>12</td>
<td>0.69</td>
<td>0.41</td>
</tr>
<tr>
<td>General function prediction only</td>
<td>49</td>
<td>36</td>
<td>29</td>
<td>8</td>
<td>0.38</td>
<td>0.04</td>
</tr>
<tr>
<td>Inorganic ion transport and metabolism</td>
<td>40</td>
<td>21</td>
<td>11</td>
<td>4</td>
<td>0.7</td>
<td>0.76</td>
</tr>
<tr>
<td>Intracellular trafficking, secretion, and vesicular transport</td>
<td>12</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>0.81</td>
<td>1</td>
</tr>
<tr>
<td>Lipid transport and metabolism</td>
<td>18</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>1.77</td>
<td>0.65</td>
</tr>
<tr>
<td>Multifunctional</td>
<td>77</td>
<td>59</td>
<td>56</td>
<td>33</td>
<td>0.77</td>
<td>0.41</td>
</tr>
<tr>
<td>Nucleotide transport and metabolism</td>
<td>31</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0.32</td>
</tr>
<tr>
<td>Protein modification/turndover/chaperones</td>
<td>28</td>
<td>7</td>
<td>12</td>
<td>5</td>
<td>1.65</td>
<td>0.5</td>
</tr>
<tr>
<td>Replication, recombination and repair</td>
<td>27</td>
<td>17</td>
<td>21</td>
<td>6</td>
<td>0.46</td>
<td>0.2</td>
</tr>
<tr>
<td>Secondary metabolism/transport</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>Signal transduction mechanisms</td>
<td>9</td>
<td>7</td>
<td>12</td>
<td>5</td>
<td>0.55</td>
<td>0.48</td>
</tr>
<tr>
<td>Transcription</td>
<td>40</td>
<td>23</td>
<td>14</td>
<td>12</td>
<td>1.48</td>
<td>0.48</td>
</tr>
<tr>
<td>Translation, ribosomal structure and biogenesis</td>
<td>48</td>
<td>23</td>
<td>23</td>
<td>7</td>
<td>0.64</td>
<td>0.48</td>
</tr>
<tr>
<td>Unspecified</td>
<td>19</td>
<td>34</td>
<td>4</td>
<td>16</td>
<td>2.21</td>
<td>0.26</td>
</tr>
</tbody>
</table>
4.3. RESULTS

represented SSRs for each functional category. To do this, we tested whether the proportion of genes with an over- and under- representation of SSRs differs between core and non-core genes across categories. Fisher’s exact tests were performed on each category. The results are shown in figure 4.3.7. The odds ratios are less than 0.6 in each significant result, indicating that in each case the ratio of SSRs are associated more with non-core regions than core regions. A quick scan of odds ratios indicates that this trend is born out across all categories, with a preponderance of overrepresented repeats among non-core genes, consistent with results shown above on genetic variation.

It is clearly very difficult to interpret and draw consistent conclusions from specific functional categories with respect to SSRs. This may in part be a power problem; some categories have few genes in them. Nevertheless, there may be an overall relationship between functional category, gene conservation and repeats. To test for such patterns, we next plotted the distribution of over- or under- representation of repeats with respect to core and non-core genes for each functional category. The results are shown in figure 4.3.8a. Since the number of genes in each category varies, we have plotted the proportion of genes in functional categories which are core against the proportion of genes in these functional categories that are over- or under- represented with repeats. The proportion of core genes in a functional category and the proportion of over-represented repeats in those core genes are not correlated \( R^2 \approx 0.02, p=0.05 \). However, the proportion of core genes which are under-represented for repeats, or do not contain a significant coverage of repeats, does scale with the proportion of the category which is core \( R^2 \approx 0.72 \) and \( \approx 0.78 \) both with \( p<0.01 \). Likewise, we plotted the proportion of genes in functional categories which are non-core and the proportion of genes in these functional categories are over/under represented. As would be expected, the proportion of non-core genes with over-represented repeats increased with the proportion of non-core genes \( R^2=0.61, p < 0.05 \), as did genes with under-represented repeats \( R^2=0.61 \).
CHAPTER 4. CONTINGENCY LOCI, HOUSEKEEPING GENES AND THE E. COLI CORE GENOME

(a) The proportion of core genes with over-represented repeats does not vary with functional category.

(b) The proportion of non-core genes with over-represented repeats varies with functional category, and is negatively correlated to the proportion of the category that is core.

Figure 4.3.8: The relationship between core/non-core and repeats across functional categories
4.3. RESULTS

p<0.05) and genes with no significant over- or under-representation of repeats, if to a lesser extent (R²=0.2, p<0.05). The meaning of these results is that the core genes have a broadly similar fraction of repeat-rich genes across all categories whereas accessory (non-core) genes have more repeats overall in less conserved functions.

4.3.6 Repeats and Essentiality for Cell Growth

We identified gene clusters with gene essentiality data from the PEC database. PEC contains a curated list of genes which are essential, non-essential, or have unknown essentiality for cell growth of *E. coli* MG1655 K12, based on published studies of gene deletions. When cross-referenced to the gene clusters in our study, we found that 267 gene clusters were marked as essential and 2213 as non-essential. 156 gene clusters had unknown essentiality for growth.

To test whether SSRs are less prevalent in essential genes, we compared SSRs according to gene essentiality for both coding and upstream regions. These results are summarized in table 4.4. A Fisher's exact test on over-representation of repeats in reading frames and gene essentiality is significant (odds ratio 1.76, p-value < 0.0005), and the direction of this is that essential genes have less over-representation of SSRs. However, even the non-essential genes are under-represented overall, indicating that repeats are not found generally throughout non-essential genes. In summary, contingency loci may be selected against in the reading frames of genes essential for growth, but non-essential genes are on average low in SSRs compared to chance.

As the right-hand table shows, we find no difference between over/under-representation of repeats in the upstream stream regions, when comparing essential and non-essential genes. Note that the total number of upstream regions included is lower than the number of genes, since not all genes have an upstream region.
### Table 4.4: Contingency table of effects between repeat coverage and gene essentiality.

<table>
<thead>
<tr>
<th></th>
<th>coding</th>
<th></th>
<th></th>
<th>upstream</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>over</td>
<td>ns</td>
<td>under</td>
<td>over</td>
<td>ns</td>
<td>under</td>
</tr>
<tr>
<td>essential</td>
<td>48</td>
<td>70</td>
<td>148</td>
<td>essential</td>
<td>86</td>
<td>93</td>
</tr>
<tr>
<td>non-essential</td>
<td>619</td>
<td>669</td>
<td>924</td>
<td>576</td>
<td>887</td>
<td>434</td>
</tr>
<tr>
<td>unknown</td>
<td>38</td>
<td>73</td>
<td>45</td>
<td>unknown</td>
<td>16</td>
<td>89</td>
</tr>
</tbody>
</table>

Fisher’s exact test for over/under versus essential/non-essential (bold face categories only) are, for coding regions: odds ratio 1.76, p-value < 0.0005.
For upstream regions: odds ratio 0.97, p=0.86.
4.4 Discussion

Previous studies of contingency loci and SSRs have in general associated these hypermutable sites with antigenic variability in pathogens Moxon et al. (2006). However, Mrázek et al. (2008), found long SSRs in the housekeeping genes of some pathogens, suggesting perhaps that SSR were widespread features of genomes rather than loci designed to deal with the uncertainty within a host or other unpredictable environments. Because Mrázek’s study, and indeed, most bioinformatic analyses of SSR were designed to examine SSR across species, it is impossible to directly address the validity of this conclusion. Contingency loci are considered to be sites of local hypermutability that are thought to confer a very rapid response to populations of microbes persisting in uncertain habitats, on a timescale much less than that of species formation. We therefore believe the appropriate first level of analysis is to look at SSR within species, since strains of conspecifics are more likely than congeners or even more distantly related species to share essential aspects of organismal ecology.

In our study, we have examined all SSRs (whether short or long) in the context of 28 pathogenic and non-pathogenic strains of *E. coli*. We included the classification of gene functions in our analysis, but also went beyond this and examined SSRs between genes which were conserved across the species or accessory (non-core), essential and non-essential, as well as in terms of genetic variability between strains.

We found that, in line with the wider expectations of the literature, SSRs in *E. coli* are significantly more prevalent in pathogenic strains than in non-pathogenic strains (figure 4.3.1). The greater prevalence of SSRs in pathogenic ecotypes could not be explained by a correlation with the proportion of the genome which was accessory rather than species-wide, suggesting that contingency loci in pathogenic strains are not restricted to genes in mobile, unstable pathogenicity islands specific to entero- or
Genes rich in SSRs are more variable on average (figure 4.3.3). SSRs are also more common in the genes which are not conserved throughout the species (figure 4.3), and are more likely to be associated with non-essential rather than essential genes (figure 4.4). These results satisfy our predictions that genes rich in repeats should be less conserved and peripheral to the organism’s basic ability to persist.

However, these clear patterns are not realized when we look more closely at SSRs partitioned by functional category. Most repeat-rich categories were non-specific, general prediction or unknown in function; in other words the ‘miscellaneous’ categories of genes with poorly understood or variable function (figure 4.3.5). There was one specific exception, *nucleotide transport and metabolism*, a category where many core genes had an over-representation of repeats in their upstream regions. However, the test for whether the conserved (core) genes were more repeat rich than the non-core genes was not significant (figure 4.3.7). Also, genes essential for cell growth of the gene downstream of non-coding regions did not significantly vary with over- or under-representation of repeats, as it did for the reading frame itself. Therefore, the over-representation of SSRs in upstream regions does not appear to impact upon essential genes and have been under negative selection. This could indicate a limitation of how SSRs have been analyzed in upstream regions in our work. We took the full region upstream of the gene, but it may be better to take a smaller fraction of the upstream region directly associated with the promoter, since not all of the upstream region may be associated with gene regulation.

While SSRs were not associated with specific functional categories, we found that there were broadly the same proportion of core genes with over-represented repeats across all functional categories, while the proportion of over-represented repeats was
4.4. DISCUSSION

proportional to the fraction of non-core genes (figure 4.3.8a). Given this broad background of core genes associated with repeats, we should expect to detect some statistically significant SSRs in some number genes regardless of function, and these may be false positives. This shows the dangers of interpreting specific cases of detected SSRs as contingency loci without evidence that they really operate as contingency loci. This is why we focus here on elucidating patterns of effects rather than extracting specific cases of SSR-rich genes in our study. The pattern we found was that SSR-associated genes are typically non-essential, genetically variable and less conserved within species.

It may be that SSRs do exactly what previous studies have suggested, providing hypervariability in accessory genes associated with unpredictable environments or host responses – but that functional categories assigned to genes are ambiguous and context-dependent. The lack of certainty surrounding the functional annotation of proteins has been highlighted before (Shrager, 2003; Furnham et al., 2009). And even though E. coli is a model organism, we know too little about what each gene does in its many habitats. E. coli is associated with a number of hosts and is found in environmental and clinical settings, even extreme environments: E. coli SMS-3-5, included in our study, was isolated from an industrial metal-toxic coastline (Fricke et al., 2008). It is worth noting that in spite of being an environmental isolate, SMS-3-5 has record levels of resistance to multiple antibiotics, the origin of which is not entirely clear. This shows how difficult it is to separate gene function from environmental context and evolutionary history.

In summary, we found that SSRs are associated with more genetically variable, less conserved genes. Genes with contingency loci should be analyzed not just in terms of functional annotation but also whether those genes are conserved, essential for cell growth, and genetically variable. In addition, studies of simple sequence repeats should take into account effects across strains and isolates within a species before looking at
higher level organization, due to their hyper-mutability. This is becoming increasingly feasible, as sequence databases continue to gain depth in terms of strains as well as breadth across species and genera. This allows the computational equivalent of the approach of experimentalists such as (Metzgar et al., 2001), where contingency loci are compared in strain catalogues, but on a comprehensive scale unimaginable in the laboratory. Genetic variability can be measured experimentally with clonal populations of *E. coli* from strain catalogues as done by Metzgar *et al.*, or it can be obtained directly by comparing aligned sequenced strains, as in our study. Whether studying contingency loci experimentally or using bioinformatics techniques, we can conclude that SSRs should be understood with a comparative approach across strains, and that considerable caution is needed when interpreting the functional categories of genes associated with repeats.
Chapter 5

Experimental evolution of *E. coli* in a varying environment

Abstract

Theoretical work has concluded that changing environments lead to modularity in biochemical and regulatory networks (Kashtan and Alon, 2005; Parter et al., 2007), and that this can increase evolvability (Kashtan et al., 2007; Parter et al., 2008). Computer simulations predict that the modularity is rapidly lost if the environment becomes fixed. In this chapter, we aim to put the theory to the test in the laboratory. We evolved a strain of *E. coli* B, as well as a derived line which has adapted to unchanged conditions for 20,000 generations, in environments with seasonal nutrient sources. We found that there were differences in fitness effects when comparing the ancestral and 20,000 generation treatments, but not when comparing between treatments which originated from the same strain. Evolutionary history may therefore be more important than seasonality in determining
short-term evolvability.

5.1 Introduction

The evolution of evolvability has been a central topic of evolutionary biology for many years. Evolvability has several definitions but here we focus on just one of these, the rate of change in fitness or another phenotypic character in response to selection, following Griswold (2006). The origin of mechanisms underlying evolvability, particularly as a selectable trait by itself rather than a correlated by-product of evolution on other traits, is an unresolved issue (Pigliucci, 2008). Several computational studies have attempted to address this question by comparing models or simulations of evolution in unchanging environments to evolution in fluctuating and unpredictable environments. Earl and Deem (2004) found that, in a molecular model of protein evolution, evolvability was selected for under rapid or extreme environmental changes. Crombach and Hogeweg (2008), in a systems biology study of evolving gene networks, studied the in silico evolution of simple logical networks in environments with seasonally alternating conditions. They found that a “genetic switch” evolved in the varying environments, where the traits changing between the varying conditions were under the control of a few genes. These genes were sensitive to mutations, allowing rapid generation of the phenotypes associated with the different environmental conditions, while other genes were mutationally robust. They conclude that evolvability, in the form of mutational sensitivity and genetic switches, has evolved in the alternating environment. Other in silico simulations of biological evolution have also suggested that varying environments can “speed up” evolution by increasing modularity (Kashtan et al., 2007). Varying evolutionary “goals” – the term used in the paper to describe the fittest possible
5.1. INTRODUCTION

phenotype – increase the rate of convergence towards an optimal solution of a mathematically stated problem, but only when the varying goals are made up of shared sub-goals. The sub-goals correspond to modules in the network itself. Since there are fewer interactions between the modules, they can adapt independently as the environment changes (Kashtan and Alon, 2005). They later showed that the modular networks were more evolvable, and that the modularity rapidly decayed over time Parter et al. (2008). Overall, these computational studies have suggested that variable environments increase modularity, and that modular networks are more evolvable.

Studies of organisms with well-characterized metabolic networks lend credence to these theories. It has been suggested that a hub connecting modules in the *Escherichia coli* heat shock response system allows such evolvability (Kurata et al., 2006). In a broad study, Parter et al. (2007) obtained the metabolic pathways of over a hundred bacteria from the KEGG database (Ogata et al., 1999) and calculate their modularity. They also scored the bacteria based upon the variability of their lifestyle (for example, they considered facultative bacteria to have more variable lifestyles than obligates). A positive correlation between variability of habitat and modularity was found; they claim that this provides evidence that environmental variation can lead to modularity. Whether this modularity confers greater evolvability for these organisms is unknown.

How important is this type of theoretically predicted response in real organisms? The difficulty in testing this it that all real organisms can sense their environment, and model bacterial species such as *E. coli* already have evolved substantial regulatory mechanisms to cope with the diverse environments in which they are found. However, this can be turned to an advantage with the right experimental design. In one notable example, Lenski and colleagues allowed replicate populations of *E. coli* to evolve in a well mixed single-resource environment for thousands of generations. Comparing the ancestral strain from this experiment to two strains that had been evolving for 20,000
generations in glucose-limited flasks, Cooper et al. (2008) found that the network of the crp regulon had greatly increased in overall size, which had the effect of increasing the number of epistatic interactions within this network. Given that crp already has a substantial impact on the connectedness of the E. coli regulatory network (Cooper et al., 2006), this result suggested that the replicate populations in this experiment evolved to have less modular regulatory networks. This may be similar to the decay of modularity observed by Parter et al. (2008) after a modular in silico organism switched to a constant environment. If this is the case, and following from the hypotheses outlined above, then we would predict that the ancestral strain could be more modular and thus more evolvable than the 20,000 generation derived line.

However, testing this theory directly is rather difficult. Consider an experiment comparing the ancestral strain's and 20,000 generation strain's response to a period of evolution in a novel environment, in terms of fitness increase or change in growth rate. To control such an experiment, we would want to introduce only one change to the environment for the 20,000 generation strain, for example a change in carbon source. But then the 20,000 generation strain has already adapted to the other aspects of the environment such as temperature and growth cycle, while the ancestral strain has not. Therefore, differences in response between the ancestral and 20,000 generation strains would be expected even without making reference to fixed and varying environments.

Therefore, we wish to compare not just how the ancestor and 20,000 generation strain respond to a novel environment, but also how they differ in their response to seasonal and 'fixed' environments. If the ancestral strain is indeed already modular, then it may adapt equally well to both varying environments and constant environments. Meanwhile, if it is true that varying environments 'speed up' the evolution of organisms by conferring modularity on them, then populations with prior evolution in a fixed environment might then adapt more rapidly to novel seasonal environments.
5.1. INTRODUCTION

compared to the same population in novel constant environments. This would require the 20,000 generation strain to gain modularity quickly enough to affect its evolvability. Note that we are saying that the 20,000 generation strain could adapt rapidly on introduction to a seasonal environment, not that it would adapt more rapidly than a population which is already modular. To put the hypothesis a different way, we are predicting that populations which have adapted to a fixed environment are more sensitive to being introduced to novel seasonal environments than those already adapted to seasonal environments.

However, there are alternative predictions. There may be pleiotropic interactions between the alternating resource conditions of the varying environment, and these may differ between the ancestral and 20,000 generation strains. If there is antagonistic pleiotropy then the populations, regardless of evolutionary history, in alternating environments may evolve a trade-off, with antagonistic indirect effects measured for controls grown in constant environments. If there is synergistic pleiotropy then the direct and indirect responses of those controls would be in the same direction, and it is then not clear that the alternating environments would be different from the controls.

An experiment was performed to investigate these predictions. Ancestral *E. coli*, as well as *E. coli* which had previously evolved in a glucose-limited environment for 20,000 generations, were evolved for 500 generations in both fixed and periodically alternating carbon sources. The evolved lines were then competed against their respective ancestors and their growth characteristics measured in each novel carbon source.
CHAPTER 5. EXPERIMENTAL EVOLUTION OF E. COLI IN A VARYING ENVIRONMENT

5.2 Materials and Methods

5.2.1 Strains

We used Escherichia coli strain B REL 606. This is the sequenced ancestral strain used by Richard E. Lenski in his group’s long-term evolution experiment (Barrick et al., 2009) as well as many other laboratory evolution experiments which are directly related to that project (e.g. Ostrowski et al. (2005)). We also used strain REL 607, a spontaneous Ara+ mutant of REL 606 capable of utilizing arabinose sugar. Ara+ and Ara- are phenotypic markers. When grown on agar plates supplemented with tetrazoleum dye and arabinose (TA agar), colonies of each strain can be distinguished due to differential reduction or oxidation of the indicator dye. Specifically, Ara- strains incapable of utilizing arabinose produce red colonies, while Ara+ strains that can use arabinose produce white or pinkish colonies.

We obtained a glycerol stock of a clonal isolate from one of the 12 Lenski populations, Ara-5, which had adapted for 20,000 generations to daily serially transferred batch culture in glucose at 37°C. While the Ara-1 population is more fully studied, and would be the ideal starting population, Ara-5 was accepted as a substitute so as to avoid any potential conflicts between future work at the experimental evolution group at Michigan State University and our own work. Ara-5 was subjected to the same treatment as Ara-1 in the original long-term evolution experiment. Like Ara-1, Ara-5 retains the ancestral mutation rate, as opposed to displaying a higher mutation rate (as found in Ara-2, Ara-4 and Ara+3) [Sniegowski, P. D., P. J. Gerrish, and R. E. Lenski. 1997. Evolution of high mutation rates in experimental populations of Escherichia coli. Nature 387:703-705]. Ara-5 did not develop polymorphism, as found in Ara-2 (Rozen and Lenski, 2000).

A spontaneous gain-of-function Ara+ mutant from the Ara-5 clone was isolated
by growing the parent strain to high density and plating on a minimal agar plate containing arabinose as a sole resource. A single Ara+ colonies was picked, and the stability and heritability of its phenotype was confirmed by re-streaking the colony on a tetrizolium and arabinose agar plate (TA plate). Neutrality of the arabinose markers for both ancestral and 20,000 generation strains were confirmed in glucose, glycerol and lactose using the fitness assay described below. The absence of polymorphism was also confirmed on these plates.

In summary, we used four parent strains in this experiment: the ancestral Ara- (REL 606), the ancestral Ara + mutant (REL 607), the 20,000-generation Ara-5 isolate, and the 20,000-generation Ara+ isolate derived from the Ara-5 clone. For brevity, these strains will hereafter be referred to as 0K-, 0K+, 20K- and 20K+ respectively.

5.2.2 Treatments

As discussed in section 5.2.4, glycerol and lactose were chosen as the two carbon sources between which the varying environments would alternate.

Kashtan et al. (2007) suggests that the effect of varying environments on modularity and rate of adaptation arises over a range of timescales, but that there is an optimal timescale. This is intermediate between the timescale on which beneficial mutations first arise and the timescale on which those mutations fix in the population. Since this timescale is difficult to know in advance, two timescales were chosen for the alternating environments. In one case, the environment changed daily. In the other case, the environment alternated every ten days. This covers an order of magnitude and corresponds to an environmental change occurring roughly every 7.6 or 76 generations. Since the experiment ran for 70 days, the switching was experienced repeatedly in each case. These two varying environments are referred to as the ‘fast’ and ‘slow’ treatments henceforth.
CHAPTER 5. EXPERIMENTAL EVOLUTION OF E. COLI IN A VARYING ENVIRONMENT

There were three controls for each strain. In one control, populations adapted only to glycerol. In another control, populations adapted only to lactose. A third control was a mixture of lactose and glycerol, equal by molecular weight.

In summary, there were five different treatments: lactose, glycerol, mixed, fast and slow. For each treatment, we included the four 0K-, 0K+, 20K- and 20K+ strains. There were four replicates in each case (given that there was found to be no bias between the Ara-/+ strains, this essentially results in eight replicate lines). In total, there were 80 lines.

5.2.3 Media and Growth Conditions

Davis minimal media (Lenski et al. 2001) was supplemented after autoclaving with 3g/L of filter-sterilized lactose or glycerol, according to the treatment. In the mixed control, there was instead 1.5g/L of each carbon source. In all treatments populations reached equivalent cell densities at stationary phase.

Populations were incubated at 37°C in 96-well plates without shaking. Each well contained 200µL of media. To ensure that there were no edge effects due to evaporation, each plate was incubated within an enclosed secondary container humidified with a dampened tissue. This approach was tested and found to be effective; no edge effects were detectable on enclosed humidified plates filled with media, while edge effects were apparent in exposed plates.

All replicate lines were transferred daily into fresh media using a pin replicator (Boekel Scientific), switching the environment as applicable. The pin replicator gave a 200-fold dilution of each well, corresponding to approximately \( \log_2(200) = 7.6 \) generations each day. In total, over 70 days of serial transfer, approximately 535 generations elapsed.
5.2. MATERIALS AND METHODS

To facilitate contamination detection, Ara+ and Ara- lines were arranged in alternate wells separated by media controls (similar to figure 5.2.1). When a media control was contaminated, and at regular intervals, all wells of the microplate were plated out on a large culture dish of tetrizolium-arabinose agar. During the 70 day course of the evolution experiment, cross-contamination was only ever detected in neighbouring media control blanks. Putatively contaminated populations were restarted from frozen freezer stocks of these populations, which were archived every two weeks.

Figure 5.2.1: Contamination checking plate used during competition assaying. Darker red colonies have the Ara- phenotypic marker (negative for growth on arabinose), pink colonies are Ara+ (positive for growth on arabinose). Ara+ and Ara- alternate to check for cross-contamination. The central two columns of the 96 well plate were media blank controls. During the 70 days of evolution, wells were in fact split between plates so that all cultures were surrounded by media blanks.
CHAPTER 5. EXPERIMENTAL EVOLUTION OF E. COLI IN A VARYING ENVIRONMENT

5.2.4 Selection of the carbon sources

If the two carbon sources share a very similar metabolism and uptake mechanism, then the selection pressure from the varying environments may be negligible. There are also some practical considerations which can rule out many carbon sources: they should not be prohibitively expensive, due to the quantity needed, and they should be convenient to store and handle. But *E. coli* has an extensive metabolism and can grow on a variety of potential carbon sources. Thankfully, as a model organism, its metabolism is also thoroughly described.

A literature review could be used to select the carbon sources *ad hoc*. But by inspecting the pathways for *E. coli* available in the KEGG database (Ogata et al., 1999; Kanehisa et al., 2006, 2010), the difference in the metabolic pathways between the various potential carbon sources can be more precisely quantified. For example, there are a small number of connecting steps between fructose, mannose and glucose (see the Fructose and Mannose Pathway in KEGG). However, the relationship between many other carbon sources is cumbersome to assess by eye, due to the sheer size of the reference metabolic pathway on KEGG. A computational approach would be much more convenient and accurate.

To develop a quantified comparison of metabolic pathways, a detailed model of *E. coli* metabolism and biomass production, known as iAF1260 (Feist et al., 2007) was used. This is a reconstruction of transport and metabolism in *E. coli* strain K-12 MG1655 – no extensive flux model exists specifically for *E. coli* B (flux refers to the steady state rate of a reaction step). In addition to the stoichiometries of the *E. coli* metabolic network, iAF1260 provides an empirically determined biomass ‘reaction’, which represents how the metabolic pathways contribute to cell growth. This can be an effective proxy for growth rate in aerated batch cultures of *E. coli* (Edwards et al., 2007).
5.2. MATERIALS AND METHODS

Note that the kinetic parameters and temporal dynamics of the reactions are not included in the reconstruction, so a model of how the metabolic network would respond to a varying environment cannot be made directly. Nevertheless, the metabolism of different carbon sources can be explored in this model, using a method known as Flux Balance Analysis.

Flux Balance Analysis (FBA) uses linear programming to find the optimal steady state growth rate (or other condition such as maximal ATP production) of the metabolic network based on the reaction stoichiometries (Edwards et al., 2002). FBA has proven effective at demonstrating the metabolic capabilities of *E. coli* (Edwards et al., 2001), and has even been used to make predictions of its optimal growth rate after adaptation to a single carbon source (Fong et al., 2003, 2005). The iAF1260 model of *E. coli* is very large, representing 1260 ORFs, and so the FBA problem must be solved on a computer using numerical methods. The linear programming (LP) problem of maximizing growth rate is as follows:

Maximize $Z = f^T v$

Subject to $Nv = 0$ and $v_{\text{min}} \leq v \leq v_{\text{max}}$.

$N$ is the stoichiometric matrix representing the *E. coli* metabolic pathways. $v$ is the solution of rates which are biologically constrained by the iAF1260 model to be between $v_{\text{min}}$ and $v_{\text{max}}$ (for example, irreversible reactions will have $v_{\text{min}} = 0$). Since we seek a steady-state solution, the product of $N$ and $v$ must be zero; at steady state, the rates of individual reaction steps can be non-zero but every reaction must be balanced stoichiometrically so that the concentrations of metabolites remain unchanged. $f$ is a vector specifying the reactions to be maximized – i.e. those reactions which contribute to biomass. The product of $f$ with $v$ will give the rate of change of biomass, $Z$, and this must be as high as possible to be optimal.

While the optimal growth rate $Z$ has one unique solution under limited resources,
the flux pattern $v$ which gives the optimal growth does not. Different flux patterns through the metabolic network may give rise to the same maximal growth rate. Therefore, the flux patterns found by the solver for each carbon source cannot be compared directly. However, a new linear programming problem can be phrased: find the minimum amount of change in the flux necessary to give the optimal growth rate of each carbon source. This does have a unique solution.

To do this, first find the maximum growth rates for two carbon sources, $Z_1$ and $Z_2$. Then define a vector $v$, decomposable into four sub-vectors: the reaction rates in carbon sources 1 and 2 ($v_1$ and $v_2$) and the positive and negative changes in reaction rates between them, $v^+$ and $v^-:

$$v = \begin{pmatrix} v_1 \\ v_2 \\ v^+ \\ v^- \end{pmatrix}$$

The linear programming problem is to minimize:

$$Z = v^+ + v^- = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}^T v$$

Where the lower and upper bounds on $v$ are:
5.2. MATERIALS AND METHODS

\[
\begin{pmatrix}
v_{1}^{min} \\
v_{2}^{min} \\
0 \\
0
\end{pmatrix}
\leq
\begin{pmatrix}
v_1 \\
v_2 \\
v^+ \\
v^-
\end{pmatrix}
\leq
\begin{pmatrix}
v_{1}^{max} \\
v_{2}^{max} \\
\infty \\
\infty
\end{pmatrix}
\]

The LP problem is subject to the new condition:

\[
\begin{pmatrix}
N & 0 & 0 & 0 \\
f & 0 & 0 & 0 \\
0 & N & 0 & 0 \\
0 & f & 0 & 0 \\
I & -I & -I & I
\end{pmatrix}
\begin{pmatrix}
v
\end{pmatrix}
=
\begin{pmatrix}
0 \\
Z_1 \\
0 \\
Z_2 \\
0
\end{pmatrix}
\]

Where \(N\) is the stoichiometric matrix and \(f\) is the biomass function previously used to find the optimal growth rates \(Z_1\) and \(Z_2\). \(I\) is the identity matrix. The inclusion of these rows in the condition matrix is to ensure that the solution satisfies the maximum growth rates and stoichiometries of the metabolic pathways. The last row is the new condition. Expanding this row gives:

\[v_1 - v_2 - v^+ + v^- = 0\]

Or, rearranging:

\[v_1 - v_2 = v^+ - v^-\]

Now, given that \(v^+\) and \(v^-\) are non-negative, their sum is the absolute difference in the rates under carbon sources 1 and 2:
\[ Z = v^+ + v^- = |v_1 - v_2| \]

Which is minimized by the linear solver.

A C++ program, \textit{Unique}, was written to solve this problem. \textit{Unique} reads an SBML-formatted file containing the iAF1260 reconstruction using libsbml (Bornstein et al., 2008). It also reads a file listing the carbon sources of interest. \textit{Unique} then generates the stoichiometric matrix \( N \) from the reactions in the SBML file and systematically finds the minimum difference in flux for every pair of carbon sources, using the LP solver from the GNU Linear Programming Kit (GNU, 2008). The primal simplex method found the solution efficiently.

While the solution itself is not necessarily unique, redundant reactions and futile cycles of flux will have been eliminated. Reactions with a flux in one carbon source, but with no flux in the other, are likely to be essential to the catabolism of that particular carbon source, or represent as short a pathway as possible to biomass, since including more reaction steps than necessary would increase the minimum absolute difference in flux. \textit{Unique} therefore prints out the reactions which are found only in one of the two carbon sources in each pair. The number of such reactions is itself a good measure of how different the metabolic pathways are.

<table>
<thead>
<tr>
<th></th>
<th>Fructose</th>
<th>Glucose</th>
<th>Glycerol</th>
<th>Lactose</th>
<th>Maltose</th>
<th>Melibiose</th>
<th>Trehalose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fructose</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Glucose</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Glycerol</td>
<td>312</td>
<td>313</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lactose</td>
<td>458</td>
<td>434</td>
<td>659</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maltose</td>
<td>445</td>
<td>437</td>
<td>646</td>
<td>109</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Melibiose</td>
<td>458</td>
<td>450</td>
<td>659</td>
<td>64</td>
<td>109</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trehalose</td>
<td>421</td>
<td>397</td>
<td>623</td>
<td>115</td>
<td>88</td>
<td>117</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.2.2: The minimum absolute differences in flux between pairs of candidate carbon sources, totaled over all reaction steps in the iAF1260 metabolic construction. Fluxes are measured in mmol/gDW.
5.2. MATERIALS AND METHODS

<table>
<thead>
<tr>
<th></th>
<th>Fructose</th>
<th>Glucose</th>
<th>Glycerol</th>
<th>Lactose</th>
<th>Maltose</th>
<th>Melibiose</th>
<th>Trehalose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fructose</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Glucose</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Glycerol</td>
<td>17</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lactose</td>
<td>39</td>
<td>36</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maltose</td>
<td>38</td>
<td>37</td>
<td>39</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Melibiose</td>
<td>36</td>
<td>35</td>
<td>37</td>
<td>11</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trehalose</td>
<td>34</td>
<td>31</td>
<td>35</td>
<td>15</td>
<td>16</td>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.2.3: The minimum difference in number of used reactions between pairs of candidate carbon sources
Figure 5.2.4: A hierarchical clustering of the shortlisted carbon sources by (a) minimum flux difference and (b) minimum number of reactions changed.
5.2. MATERIALS AND METHODS

Several carbon sources were shortlisted, on the basis of cost-effectiveness and the existence of literature relating to that carbon source in *E. coli*. The shortlisted carbon sources were glycerol, fructose, lactose, trehalose, melibiose and maltose. These carbon sources, along with glucose as a comparison, were analyzed with the procedure outlined above. The pair-wise minimum total difference of flux is shown in figure 5.2.2, which can be used to directly cluster the carbon sources according to similar minimal metabolism (5.2.4a). Likewise, figures 5.2.3 and 5.2.4b compare how many reaction steps differ between carbon sources.

The results are consistent with the extensive literature on metabolic pathways. For example, fructose and glucose are both PTS (phosphotransferase system) sugars, and should be expected to cluster together by inspection of the KEGG pathways, as suggested above. It would have been much more difficult to predict that lactose and glycerol have the greatest difference in both number of reaction steps and overall flux.

Once identified by Unique, the reactions for the chosen carbon sources were inspected. Many of the relevant reactions were actually present in both carbon sources, for example glycerol kinase, presumably as a means to reduce the flux difference as far as possible. However, glycerol alone required the glycerol uptake facilitator protein, *glpF*, and lactose alone required β-galactosidase to split the lactose disaccharide into two galactose molecules, as per normal lactose catabolism. The two carbon sources share the glycolysis pathway from Glycerate-3-Phosphate en route to biomass.

Metabolic flux difference is unlikely to give a complete measure of how much selection pressure there is in switching between two carbon sources. While lactose and glycerol have the greatest required difference in flux, they are both transported through the outer membrane by *OmpF* in *E.coli* B (Travisano and Lenski, 1996). They are transported across the inner membrane by different mechanisms (the lactose carrier *lacY* and glycerol facilitator protein *glpF*), but the regulatory organization of *E.coli*
B may require a more substantial change in regulation change between PTS and non-PTS sugars, for example glucose and lactose, because of the inhibitory effect of crp via cAMP on the lacYZA operon.

For this reason, the large differences in lactose and glycerol metabolism also do not necessarily imply that there would be diauxic growth in a mixture of lactose and glycerol. Diauxie is a specific regulatory measure in bacteria to suppress the catabolism of one sugar in the presence of another (most famously, lactose suppression in the presence of glucose). *E. coli* B may not employ a regulatory mechanism between lactose and glycerol. Growth curves taken in the mixed environment, before and after the period of laboratory evolution, have shown that there is no diauxic regulation between lactose and glycerol in *E. coli* B (data not shown).

### 5.2.5 Fitness Assays

Fitness was measured using the method of (Lenski et al., 1991) adapted to the format of 96-well plates. Briefly, frozen stocks of Ara+ and Ara- evolved lines were prepared in one 96-well plates, while their complementary Ara- and Ara+ ancestors (either 0K or 20K) were prepared in another plate. These two master plates were used to initiate preconditioning growth of competitor isolates prior to paired competition. For each replicate fitness assay, two 96-well plates containing 200µl LB were inoculated with either the evolved or ancestral isolates, and incubated overnight at 37°C. The next days 2µl from each well was transferred to 96 well plates containing the same media as was to be used in the competition (lactose or glycerol), and again grown overnight to further precondition the populations to the carbon source. The preconditioned populations were then combined into the competition environment at a 1:1 volumetric ratio and plated on TA agar, immediately after the competition plates were prepared and again after 24 hours of incubation. Fitness was calculated as the ratio of each
5.2. MATERIALS AND METHODS

strain's Malthusian parameter during one day of paired competition, following Lenski (1991). Plates were incubated in the same conditions used for the evolution experiment presented in this chapter. Five replicate fitness assays were measured in each case.

5.2.6 Determination of Growth Rate, Lag and Stationary Phase Times

Growth curves of all populations were measured using a plate reader measuring optical densities at 550nm at 5 minute intervals over 14 hours. Between measurements, the plates were shaken and incubated at 37°C. The 96-well plates and the growth media were unchanged from the environment used over the 500 generations of evolution.

The growth curves were then plotted (as log of optical density) and fitted to a smooth curve using 5th-order polynomial splines, so that the first and second derivative could be used to extract the lag, exponential and stationary phases. The exponential phase was taken as the linear region of the growth curve, which is identified as beginning at log optical densities of -7 (the lower end of the region in which the optical densities increase smoothly) and ending 10% below the maximum measured optical density of the fitted growth curve. Using a point 10% below maximum, rather than the maximum itself, was found to be effective in identifying an upper point in the linear region. Using a value nearer to the highest optical density measurement is unreliable, because it sometimes includes part of the stationary phase when optical densities peak and then decay. While the beginning and end points of this linear region chosen in this way are arbitrary, they are very effective at extracting the linear region of the growth curve. All linear models fitted to the log phase had R-squared values exceeding 0.99. However, the endpoints can only be used to identify the subset of data for the linear model of the exponential phase, rather than as measures in themselves, because they
are arbitrary and conservative. Further calculation is required to correctly extract the end of the lag phase and the beginning of the stationary phase. The end of the lag phase was identified as the time point at which the second derivative first becomes non-zero (ignoring small fluctuations within the maximum resolution of the measurements).

To find the stationary phase is slightly more complex. The trendline of the exponential phase is first plotted against the real data. Then, where the fitted line diverges from the peak optical density, the linear region has ended. These methods, while somewhat counter-intuitive, were found to be very robust. The placement of all lag, exponential and stationary phases were inspected for appropriateness. An example of a processed growth curve is shown in figure 5.2.5. Note that the interpolated polynomial splines levels off differently to the raw data in stationary phase, but this does not adversely affect the measures themselves (growth rate in exponential phase, and time points for lag phase and stationary phase). Indeed, attempting to fit the curve more closely also results in fitting noise in the data, with no improvement to the measures desired. Meanwhile, attempts to fit the data directly to a logistic equation were found to be less stable than the method described, with the growth phase sometimes misidentified.

5.3 Results

5.3.1 Fitness of Controls Relative to Ancestor

Evolved populations from all treatments were competed in both lactose and glycerol. The direct effects of adaptions in the controls (e.g. glycerol-adapted populations competed against their ancestor in glycerol) may differ from the indirect effects (e.g. glycerol-adapted populations competed against their ancestor in lactose). The direct
5.3. RESULTS

Figure 5.2.5: An example of a fitted growth curve. The circles (black) are the raw log optical density values, while the smooth curve (blue) shows the polynomial splines fitted to those measurements. The sloping line (red) is the fitted exponential growth phase; its gradient is used as a measure of growth rate. The first vertical line (purple) is the end of the lag phase, while the second vertical line (green) is the beginning of stationary phase. All data fitting was inspected to ensure that they were sensible.
and indirect effects may also be asymmetric between lactose and glycerol, as well as depend upon whether the evolved populations are derived from either the 0K or 20K strain. Direct and indirect effects of adaptation to different fixed environments were analyzed using analysis of variance (table 5.1).

<table>
<thead>
<tr>
<th>Term</th>
<th>df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F</th>
<th>P Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ancestor</td>
<td>1</td>
<td>0.2</td>
<td>0.2</td>
<td>15.12</td>
<td>&lt;0.001***</td>
</tr>
<tr>
<td>Treatment</td>
<td>1</td>
<td>0.03</td>
<td>0.03</td>
<td>1.91</td>
<td>0.17</td>
</tr>
<tr>
<td>Competition</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.13</td>
<td>0.718</td>
</tr>
<tr>
<td>Ancestor:Treatment</td>
<td>1</td>
<td>0.67</td>
<td>0.67</td>
<td>50.37</td>
<td>&lt;0.0001***</td>
</tr>
<tr>
<td>Ancestor:Competition</td>
<td>1</td>
<td>0.13</td>
<td>0.13</td>
<td>9.51</td>
<td>&lt;0.003**</td>
</tr>
<tr>
<td>Treatment:Competition</td>
<td>1</td>
<td>0.02</td>
<td>0.02</td>
<td>1.46</td>
<td>0.23</td>
</tr>
</tbody>
</table>

Table 5.1: ANOVA of relative fitness in control populations (relative fitness ~ ancestor * treatment * competition environment). Ancestor represents whether the control was derived from the 0K or 20K strains, Treatment is the environment in which they evolved for ~ 500 generations in our project (lactose, glycerol, mixed, fast, slow), and Competition is the environment in which they were competed against their ancestor (lactose or glycerol). The number of residual degrees of freedom was 230. Since treatment is the term of interest, effect sizes and Tukey post-hoc testing for the interaction between treatment and competition environment is examined in table 5.2.

From this analysis, we found significant effects of the evolutionary history of the ancestor (i.e. whether the populations were derived from the 0K or 20K strains), with an overall effect size of 0.04. This is most apparent in figure 5.3.1. There is also an interaction between the control treatment and competition environment in the fitness assay. Post-hoc pairwise tests show that lactose-evolved populations have a higher relative fitness than glycerol-evolved populations when competed in lactose (see table 5.2). The converse, that glycerol-evolved populations are fitter in glycerol is non-significant.

Indeed, we found that the glycerol-adapted populations did not significantly increase in fitness relative to the ancestor in the 20K-derived populations (one-tailed
Table 5.2: Post-hoc analysis of interaction between treatment and competition environment (i.e. Treatment:Competition term of model in 5.1). Values show effect size between rows and columns; significant effects are starred. Lactose-evolved populations have a higher relative fitness than glycerol-evolved populations (p=0.035 *) when competed on lactose.

<table>
<thead>
<tr>
<th>Competition</th>
<th>Glycerol</th>
<th>Lactose</th>
<th>Glycerol</th>
<th>Lactose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glycerol</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Glycerol</td>
<td>-0.04</td>
<td>-0.04</td>
<td>0.02</td>
<td></td>
</tr>
<tr>
<td>Lactose</td>
<td>0</td>
<td>0.06 *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lactose</td>
<td></td>
<td></td>
<td>0.06</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Post-hoc analysis of interaction between treatment and competition environment (i.e. Treatment:Competition term of model in 5.1). Values show effect size between rows and columns; significant effects are starred. Lactose-evolved populations have a higher relative fitness than glycerol-evolved populations (p=0.035 *) when competed on lactose.
Figure 5.3.1: Responses to lactose and glycerol compared for controls. G=Glycerol-evolved populations, L=Lactose-evolved populations, points with black error bars have 0K ancestor and points with red error bars have 20K ancestor. Error bars represent standard error.
5.3. RESULTS

More surprisingly, the indirect response of glycerol-adapted 20K treatments was significantly higher than the direct response ($t=-3.5$, $df=48.2$, $p=0.0004$). Note that for the glycerol-adapted 20K treatments, the direct response is fitness increase on glycerol, and indirect response is fitness increase on lactose. Both the non-significance of the 20K strains competed on glycerol, and the significance of the 20K strains competed on lactose, can be seen in figure 5.3.1, where the 20K strains for both lactose and glycerol treatments are close to the line indicating no fitness increase on glycerol, but away from the line indicating no fitness increase in lactose. This interesting result was explored further using growth curve data, and is discussed below.

5.3.2 Relative Fitness in All Treatments

The effect of varying and mixed environments were examined next. Given the large amount of variation explained by the evolutionary history of the ancestral strain, 0K and 20K derived populations were separated to more clearly show their different responses. ANOVAs, including all five treatments, are given in table 5.3. No treatments had a higher or lower relative fitness overall in both lactose and glycerol. We predicted that there might be an overall difference with treatment if populations exposed to varying environments either rapidly adapted to both lactose and glycerol or evolved a trade-off between the two carbon sources. We find no evidence to support that prediction.

There were significant interactions between competition environment and treatment in the 0K ancestors. Post-hoc pairwise testing is shown in table 5.4. Almost all significant pairwise interactions are cases where fitness relative to the ancestor is greater in glycerol than in lactose, i.e. the response to glycerol is greater. The exception is lactose; this is expected because it does not have a greater indirect response in glycerol.
Table 5.3: ANOVA (relative fitness ~ competition * treatment) reveals interaction for 0K but not 20K populations. Treatment is the environment in which the populations were evolved for ~ 500 generations in our project (lactose, glycerol, mixed, fast, slow), and Competition is the environment in which they were competed against their ancestor (lactose or glycerol). In each case, the residual degrees of freedom are df.D=218.
5.3. RESULTS

Figure 5.3.2: Relative fitness of treatments in lactose and glycerol.
than in its direct environment. There are more such pairwise interactions in the mixed
treatment populations than in the varying treatment populations. However, unlike
the other treatments mixed-adapted populations are significantly less fit than lactose-
adapted populations in the lactose environment (this is the pairwise interaction in the
bottom-right quarter of the table). These results are shown graphically in figure 5.3.2
(black points represent 0K-derived populations).

<table>
<thead>
<tr>
<th>Glycerol</th>
<th>Fast</th>
<th>Glycerol</th>
<th>Lactose</th>
<th>Mixed</th>
<th>Slow</th>
<th>Fast</th>
<th>Glycerol</th>
<th>Lactose</th>
<th>Mixed</th>
<th>Slow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glycerol</td>
<td>0.1</td>
<td>0.06</td>
<td>0.04</td>
<td>0.07</td>
<td>0.12</td>
<td>0.16</td>
<td>0.09</td>
<td>0.19</td>
<td>0.16</td>
<td>0.16</td>
</tr>
<tr>
<td>Lactose</td>
<td>0.04</td>
<td>0.01</td>
<td></td>
<td></td>
<td>0.06</td>
<td>0.11</td>
<td>0.03</td>
<td>0.14</td>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>Mixed</td>
<td>0.06</td>
<td>0.02</td>
<td>0.03</td>
<td></td>
<td>0.08</td>
<td>0.13</td>
<td>0.05</td>
<td>0.16</td>
<td>0.12</td>
<td>0.12</td>
</tr>
<tr>
<td>Slow</td>
<td>0.03</td>
<td></td>
<td></td>
<td></td>
<td>0.05</td>
<td>0.1</td>
<td>0.02</td>
<td>0.13</td>
<td>0.09</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4: Post-hoc pairwise testing of interactions between environment of fitness assay and treatment. Effect sizes (differences in mean) are shown, with significant results starred (** p < 0.001, * p < 0.01, * p < 0.05). Values above the diagonal indicate that the difference is on average higher in the case given in the column labels.

Interactions were not found to be significant between competition environment (lactose or glycerol) and treatment in the 20K-derived populations. Treatment therefore seems to have had no effect at all for populations evolving from the 20K ancestor.

### 5.3.3 Analysis of Growth Curves

The non-significant direct response of the 20K-derived glycerol-adapted populations to glycerol is surprising because of the indirect fitness advantage in lactose of those populations, compared to their 20K ancestor. This suggests that there has been an evolutionary response in those populations, and yet there is no measurable advantage to those changes in the environment in which they evolved. Neutral mutations are
unlikely to fix in large batch populations without a selection pressure. This requires an explanation.

Once the growth rate data was reduced to the three measures noted earlier – lag, stationary phase and log growth – Analysis of Variance was used to examine differences in the growth characteristics of the 0K and 20K derived populations, for each measure. Test results are summarized in figure 5.3.3. The 0K derived populations had the growth characteristics we expected given the competition experiments: the growth rates are higher in all the adapted treatments than in the 0K ancestor. In keeping with this, the adapted populations entered stationary phase sooner. The only 0K-derived treatments to show a statistically significant reduction in lag phase were the lactose and mixed treatments, in comparison to the glycerol treatment.

![Figure 5.3.3: Growth in glycerol: effect on lag, exponential and stationary phase.](image)

Above the diagonal indicates that the treatment in the column header is significantly less than that indicated in the row header. Likewise, below the diagonal indicates that the treatment in the column header is significantly greater than in that indicated in the row. Stars indicate significance (**p < 0.01, *p < 0.05**).

The growth characteristics for the 20K derived populations are more revealing.
Figure 5.3.4: Growth rates of each treatment comparing lactose (x axis) and glycerol (y axis)
5.3. RESULTS

Figure 5.3.5: Length of lag phase of each treatment comparing lactose (x axis) and glycerol (y axis)
CHAPTER 5. EXPERIMENTAL EVOLUTION OF E. COLI IN A VARYING ENVIRONMENT

There is no increase in growth rate on glycerol in any treatment, which fits with non-significant change in fitness seen in the competition experiments in glycerol. However, the lag phase has been significantly reduced in all treatments, compared to the 20K ancestor, including glycerol. Since the reduction in lag phase is observed in all treatments, and not significantly more so in glycerol than in any of the other treatments, it appears to be an adaptive response to the change from glucose (or possibly another aspect of the environment not controlled for) rather than as a response to the specific carbon sources in the treatment. Furthermore, there are no other pairwise differences in growth characteristics between treatments. This fits with the observation in the previous section that there are no interactions between treatments. Figures 5.3.4 and 5.3.5 show these parallel changes in growth characteristics in graphical form.

5.4 Discussion

This experiment set out to investigate the effect of varying environments on the experimental evolution of E. coli. We evolved a strain which had been adapting to glucose in batch culture for 20,000 generations (20K), and its ancestor E. coli REL 606 (0K), to five treatments: lactose, glycerol, a mixture of lactose and glycerol, and two different timescales in which the carbon source alternated between lactose and glycerol. Each population evolved in their treatment for 500 generations.

Examining the controls, in which populations were exposed only to either lactose or glycerol, we found that they had different responses in the 0K derived populations but that they were not significantly different in the 20K populations (table 5.1 and figure 5.3.1).

Expanding the analysis to include all five treatments, we found that that none of the treatments led to a higher fitness in both lactose and glycerol. This suggests that
none of the treatments caused a rapid increase in evolvability, which was one of our predictions.

However, there were interactions between the lactose and glycerol fitness responses in the 0K-derived populations (figure 5.3.2). Those which had been exposed to both glycerol and lactose – whether in a mixture or in a varying environment – increased their fitness more on glycerol than lactose. Interestingly, this effect was clearest in the populations exposed to the mixed environment; but the populations from the mixed treatment were also on average less fit in lactose on average than the lactose-evolved populations themselves. This was not true of the other populations.

In contrast, all 20K-derived populations had a significant fitness response in lactose but not glycerol. There were no interactions between treatments and environment in the fitness response, and each treatment had a reduced lag phase while the change in growth rate was not significant. Indeed, the indirect response of the glycerol-evolved populations to lactose was significant, due to this reduced lag phase, even though the direct response to glycerol was (presumably) too small to measure significantly. This suggests that there may have been a highly parallel evolutionary response in the populations which evolved from the 20K strains, and that the pleiotropic effects of these mutations are such that lactose has a greater fitness response.

Ultimately, it appears that the 20K-derived populations have adapted to the change from the environment which they had been exposed to for 20,000 generations, rather than specifically to lactose, glycerol, or the alternation between those two environments. This may have dominated the evolutionary response of the populations. We were therefore unable to demonstrate the evolution of evolvability. This may be a limitation of the experiment; perhaps the evolutionary period needed to be longer. Also, the inability to measure fitness increase in glycerol-adapted strains, in spite of a measurable reduction in lag phase, suggests that our fitness assays lacked statistical power.
Therefore, we wish to gather more growth curve data, as well as explore alternative approaches to measuring fitness, in future work.

Nevertheless, our work did find different responses by treatment, depending upon the ancestral strain. This demonstrates a difficulty with transferring theoretical models of the evolution of evolvability to the study of modern organisms. Simulations of the evolution of evolvability often begins with randomized and unevolved digital organisms, a 'clean slate'. In contrast, experimental evolution in living systems begins with strains with their own evolutionary history. It may be that the long evolutionary past of all modern organisms carries with it substantial evolvability, and that its loss and gain cannot be easily detected. An exciting future direction could therefore be to study the evolution of evolvability by evolving RNA or proteins from de novo sequences.
General Discussion

This thesis began by introducing various aspects of genetic architecture, i.e. the mapping between genotype and phenotype. This mapping is not one-to-one: genes can affect multiple traits (pleiotropy), and multiple genes can contribute to a trait (epistasis). There are other forms of genetic architecture too, for example genes can be physically linked by being proximal on the genome. There can therefore be a complex network of interactions between genes and traits. The evolution of genetic architecture, and the effect of genetic architecture on evolution, are subtle topics which reflect the complexity of these interactions, and the debate of whether evolvable genetic architectures are selected is an old one. The emergence of systems biology has lead to an increasing number of new computational studies of the evolution of evolvability, commonly using genetic algorithms (GA) and bioinformatics to explore the evolution of evolvability in fluctuating environments.

The first main chapter of this thesis (chapter A) highlighted an insidious problem related to these studies. Genetic algorithms have become a prevalent research tool for modeling evolution, but in spite of using the language of evolutionary biology, GA were developed to solve numerical and engineering problems and not to represent natural selection. At face value, the design rationale of genetic algorithms might not appear to be an issue, since they still contain the essential elements of heritable variation and selection. Unfortunately, many widely used innovations in GA, such as the survival
of the fittest without variation (*elitism*), have no real counterpart in natural selection and are used merely for computational efficiency. We tested to see whether elitism biased the effect of two different fitness functions – GA jargon for the environmental conditions – on the evolution of genome size. We found that elitism did affect the continued evolution of genome size if the elite individuals ceased to be replaced by fitter offspring. Such a situation can readily arise in network evolution GA where fitness is absolute and can have a maximum value. This has important implications for modeling network evolution with GA because genome size is often related to network size or the number of interactions in the network. We therefore presented several solutions which allowed elitism to be used for computational efficiency without adversely affecting the results, such as measuring network properties over the whole population (the majority of which still undergoes mutation). This formed part of a methodology which we tested and included in this chapter as a resource for modelers.

This methodology was then applied in chapter 2 to examine robustness of protein networks to rapid fluctuations in concentrations, or ‘noise’. Robustness was a selected trait in networks which evolved in the presence of noise, and we examined a range of aspects of the network topology to try to understand what made those networks robust. While robust networks were more complex on average, network and genome size increased over evolutionary time just as in chapter A, possibly as a form of mutational robustness rather than as robustness to noise. Overall, we found that robustness could best be explained by the fine-tuning of biochemical kinetic parameters.

While large, complex networks were not necessary to be robust, where complex networks with additional feedback loops did evolve they could not always be ‘pruned’ to the smallest observed robust network size and remain viable. It is therefore not possible to generalize, from specific examples of dynamical systems such as the *Neurospora* circadian circuit, that network complexity or interlocking feedback loops are essential
5.4. DISCUSSION

design principles of robustness – they may just be essential in that particular network. A limitation of the work in chapter 2, however, is that it cannot be directly compared to systems as complex as circadian rhythms, because our study did not include dynamics on multiple timescales. Though computationally challenging, the next logical step is to attempt to evolve circadian rhythms in silico and compare their network topologies to those occurring across a wide range of eukaryotes.

Chapter 3 tested the effect of two alternating environments on genetic architecture and evolvability. Critically, we included controls where the resources in both of the alternating environments were present at the same time. We found modularity and evolvability in the rapidly varying environments as well as this combined control. However, modular genetic architecture did not explain the ‘evolved’ evolvability, whereas the extent to which the evolved population contained generalists did. Indeed, the evolvability of the generalists in the combined control suggests that evolvability in this case was a by-product of generalism, rather than a selected trait. It also suggests that the lifestyle of the organism (how unpredictable its environment is) may be important determinants of genetic architecture and evolvability.

While this model did not include the possibility of regulation through environmental signals, real organisms respond to such environmental changes with regulation. Environments varying on rapid regulatory timescales therefore have very different implications to environments varying on evolutionary timescales. Environmental signals which change from moment to moment during the lifetime of a single individual implicates regulation, whereas a selection pressure varying on a time scale of generations implicates evolvability. As an explanation, it is the difference between modularity being a product of regulatory architecture versus modularity being an evolutionary strategy. It is quite possible that the link between modularity and variable environments, identified in metabolic networks by Parter et al., 2007, is explained by the amount of
regulation the bacteria needs for its lifestyle. Indeed, it is arguably the simpler explanation. For example, Parter assigns facultative bacteria as having more variable environments than obligate anaerobes. But facultative bacteria such as *E. coli* can by definition *regulate* between aerobic and anaerobic conditions. Furthermore, there is well-defined modularity associated with that regulation: *E. coli* has for example the *cyo* operon for the aerobic reaction series as well as the *oxyR* and *soxRS* regulons to respond to oxidative stress when outside the gut (Bunn and Poyton, 1996; Minagawa et al., 1990). These operons and regulons, by grouping together functionally related gene products under shared promoters and transcription factors, confer modularity on the bacteria at the level of regulation.

Nevertheless, there is evidence that bacteria such as *E. coli* do respond to relatively rapid changes in the environment through localized hypermutability, through tandem sequence repeats (contingency loci). This was the subject of chapter 4, which presented a unique comparative approach to contingency loci by comparing sequenced strains *E. coli* in terms of genetic variability, gene function and conservation across the strains. We found that gene function was an ambiguous indicator of contingency loci. This makes sense, since contingency loci allow genes to respond to rapid environmental changes, potentially changing its function. A concrete example of a contingency locus which allows a direct functional change in the protein product is the *ahpC* gene of *E. coli* (Ritz et al., 2001), which we describe in the introduction to chapter 4. *E. coli* is an adaptable organism, and contingency loci can contribute to its evolvability by allowing localized hypermutability in accessory genes while maintaining lower rates in other genes; indeed, many genes have an under-representation of sequence repeats compared to chance, suggesting that repeats are selected against in general. In future work, we would like to extend the strain-level approach across multiple groups of strains in a wide range of bacterial species.
Chapter 5 attempted to put the evolvability of *E. coli* to the test in varying environments, using experimental evolution of populations derived from a strain which had been adapting to glucose in batch culture for 20,000 generations, as well as its ancestor. We ambitiously predicted that we might see an overall difference across environments for those populations which we evolved in varying environments, but no such observation was made. Also, the interactions between the various treatments were only seen in those populations derived from the ancestor, and these interactions did not clearly discriminate between varying and constant environments. Those derived from the 20,000 generation strain appeared to have a parallel response in all treatments, suggesting that the evolutionary response was to the change in environment itself, rather than the specific carbon source used in the treatment. This highlights a difficulty with comparing theoretical studies of the evolution of evolvability to real organisms; simulations of evolution usually begin with random networks, whereas living organisms have billions of years of evolutionary history. It also demonstrates the difficulty of predicting the outcome of experimental evolution, due to the complex relationship between genes, traits and evolutionary history.

In summary, evolvability and environmental variability appear to have a multifaceted relationship that reflects the complexities of the genotype-phenotype map itself. Previous studies have shown that evolutionary responses to changing environments lead to modularity and evolvability. We find that rapidly changing environments can also lead to modularity – and evolvability as a potentially separate by-product – due to generalism. Nevertheless, localized sources of hypermutability such as contingency loci can act as a rapid evolutionary response to unpredictable environments. Real biological organisms may be as evolvable as they are because they combine all of these factors, not to mention functional modularity due to regulation. Given that all these sources of evolvability are likely to be present to some extent, future work in this area should
aim to tease apart their relative contributions using appropriately designed models of evolution which fully integrate the approaches and expertise of both systems and evolutionary biology.
Appendix A

Methodology: The Application of Genetic Algorithms in Evolutionary Systems Biology

Abstract

Genetic algorithms (GA) are commonly used tools to simulate evolution, particularly to study the evolution of genetic and biochemical networks. However, most work on genetic algorithm methodologies aim to improve the efficiency and efficacy of GA for machine learning and problem solving, rather than to simulate evolution. Little work exists which places GA in a biological context. We try to fill this gap and identify some of the issues which may arise when using these problem solving tools for modeling. We focus in particular on a common technique in GA called elitism, which preserves the unaltered genome of the fittest individuals at each generation. While elitism is useful for computational efficiency – because it allows for
convergence on solutions with smaller populations and higher mutation rates – it may adversely affect the results of the simulation and has no counterpart in nature. We address this issue and review the other design problems associated with using genetic algorithms to model evolution, in the context of a computational experiment to study the evolution of large genomes. Using this system, we show that elitism adversely affects the results of evolutionary simulations if not used correctly. We also show that genomic ‘code bloat’, often observed as an unintended by-product in genetic algorithms, is a form of robustness to mutation and that penalising the fitness of individuals with large genomes is not always necessary to maintain “manageable” genome sizes. In this chapter, we test the role of different selection methods on GA outcome, and in so doing provide a number of insights into the effective use of genetic algorithms to model evolution. These insights are given in the form of a list of guidelines, both as a cautionary tale and as a resource for the development of GA in a biology research context.

A.1 Introduction

Genetic algorithms, originally developed by computer scientists and engineers based upon presumptions of the process of evolution by natural selection, have recently become an important tool for systems biologists aiming to understand the evolution of genetic and biochemical networks. Genetic algorithms (GA) work by imposing some form selection upon a population of individuals, each encoding a genome usually represented by a series of numbers or letters. This is done iteratively, and GA always have discrete generations of individuals. At each iteration of the GA simulation, the fitness
of each individual is assessed and used to determine who should replicate into the next generation, with genomes mutated at some rate, and perhaps also undergoing recombination or sexual reproduction. Therefore, genetic algorithms (GA) in their general form contain the essential aspects of evolution by natural selection: heritable variation and some form of differential reproduction/selection.

There is a considerable body of literature on GA methodologies in the discipline of computer science, the classic text being *Genetic Algorithms in Search, Optimization, and Machine Learning* (Goldberg, 1989). As the title implies, the focus on genetic algorithms is its use for solving computational problems. Most methodology papers also naturally focus upon the effectiveness of GA at solving a problem, such as Goldberg’s further work in Goldberg and Deb (1991), which compared methods of selecting individuals for the next generation with a strong focus on computational complexity. Genetic algorithms are also frequently examined for a specific domain; just as an example, consider Alander (1991) which compares GA techniques for optimising the control of robots. Some of these problem-specific studies do come from the fields of bioinformatics and systems biology, but still typically for optimising problems rather than biological modeling. Helaers and Milinkovitch (2010) uses GA for find phylogenies by maximum likelihood methods, while Rocha et al. (2008) compares evolutionary and other approaches for optimising biochemical networks. There appear to be few studies which examine the properties of GA for the domain of simulating evolution.

We seek to address this gap. This is important, because it cannot be assumed that engineering tools inspired by biology necessarily retain biological meaning. Evidence from the literature suggests that this lack of direct translation between biology and optimization is not widely recognised, and that the structure of the genetic algorithm is typically considered less important than the model solution, thus potentially limiting the degree to which results from GA can be extrapolated to real biological
APPENDIX A. METHODOLOGY: THE APPLICATION OF GENETIC ALGORITHMS IN EVOLUTIONARY SYSTEMS BIOLOGY

systems. Paladugu et al. (2006) attempted to compare how many generations it took to evolve biochemical networks based upon either Michaelis-Menten kinetics or Hill kinetics. However, the GA using these distinct kinetics were evolved using different models of selection (truncation versus tournament selection, as described in subsection A.2.1), and these may have had important confounding roles in determining the outcome of the simulations. While studies of genetic algorithms focus upon comparisons of selection methods and parameters in terms of their performance as computational tool, e.g. Goldberg and Deb (1991) as discussed above, differences in the convergence of fitness between different selection methods are well known in the computer science literature (?). The correct approach in Paladugu et al. (2006) would be to use only one model of selection, or to compare the models under both forms of selection in turn. Still other publications have not even stated the details of the genetic algorithm they use (e.g. supplementary information of Kashtan and Alon (2005)). This is an important omission and source of confusion, because it is very likely that the manner in which the GA is structured and carried out has important implications for the results and biological interpretation of these studies. We therefore feel that an examination of how GA fit into biological simulations is necessary, to prevent such methodological errors. To address this, we directly test the effect of different aspects of the genetic algorithm structure on simulating the evolution of biochemical and gene networks.

An example of how a genetic algorithm’s design might not be biologically meaningful is given by combining the genetic algorithm with a type of selection known as ‘elitism’ (prevalent in many of the papers cited in this chapter). Elitism works by preserving the fittest individuals between generations, without mutation or other genetic variation. This has the advantage of guaranteeing convergence on a solution, with a monotonically increasing fitness of the fittest individuals. However, there is no counterpart of elitism in nature. No organism can completely eliminate the possibility of mutations from
occuring in its genome each generation. When elitism is used in a GA, only the elite’s offspring experience a mutational load, not the elite themselves. This is critical because mutational load can lead to mutational robustness, which is the capacity of a genome to withstand mutations in order to retain fitness (de Visser et al., 2003). If the causes and consequences of mutational robustness are a potential property of evolutionary interest, then the inclusion of elitism in the model of selection may be inappropriate at best, and misleading at worst.

Even in studies where mutational robustness is not of primary interest, elitism may still affect the results. Individuals in the genetic algorithm can still evolve so as to be robust to mutations in cases where mutation rates are high, for example by having a large network with redundant connections. A network in this context is a system of interactions between the genes which make up the artificial genome, and the size and complexity of such networks is frequently of interest in systems biology studies of evolution. Examples can be found in Kashtan and Alon (2005); Kashtan et al. (2007); Parter et al. (2008); Crombach and Hogeweg (2008); Deckard and Sauro (2004); Paladugu et al. (2006); in these studies the fittest individual network is the main subject of analysis. As elitism retains the fittest individuals without mutational load, analysing the network of the fittest individual could be misleading when elitism is incorporated in the model of selection. For example, if a study is comparing evolution in two different environments, the elite in the population may fix earlier in one environment than in the other, and so experience different mutational loads in each case.

In order to test the impact of elitism on network evolution, we implemented several genetic algorithms to evolve networks of logic functions. We then use this platform both to test how the inclusion of elitism in the model of selection can affect genome size, as well as identify some of the other critical problems that can arise using genetic algorithms to simulate natural selection. Genome size was chosen for this study because
Appendix A. Methodology: The Application of Genetic Algorithms in Evolutionary Systems Biology

It is easily understood and it is important in the simulation of network evolution. It is important because the genome usually represents components or interactions in the network, and so a larger genome implies a larger or more highly interconnected network. It is also an appropriate case study because it allows us to explore another difficulty that systems biologists frequently encounter when using genetic algorithms to explore networks: a phenomenon sometimes described as ‘code bloat’ (François and Hakim, 2004). This is where genomes become large and contain redundant genes. The fact that ‘code bloat’ is so prevalent suggests that it may be important in the evolution of networks, possibly as a form of mutational robustness, rather than an artifact of the structure of the GA model. Modelers wishing only to use genetic algorithms to find novel network designs, but not to study evolutionary processes, may wish to counteract this, and so are referred to the review in Foster (2001).

We find that the inclusion of elitism in GA models of evolution can affect the genome or network size, independent of other aspects of the GA. We also find that the continuous increase of network size over evolutionary time, i.e. ‘code bloat’, depends upon the fitness criteria rather than being a necessary result of genetic algorithms. We also tested the effect of the selection model in the context of network evolution, and show that tournament selection results in higher diversity than truncation selection. In the process of testing for potential pitfalls in the use of genetic algorithms to model evolution, we formulate a series of guidelines which may be of interest to future modelers.
A.2. METHODS

A.2 Methods

A.2.1 Choosing the Model of Selection

Part of what makes genetic algorithms easy to use is that a ‘fitness function’ can be defined which takes the representation of one individual’s genome, translates it into a phenotype and evaluates that phenotype based upon some desired property, as explained further in A.2.4. This frees us from having to model the environment through which the individuals interact. It does not however, relieve us of the need to decide how individuals are selected based upon their relative fitnesses. We still need to choose a model of selection.

The simplest form of selection in a genetic algorithm is truncation selection, which allows the fittest individuals (say, the top 10%) to reproduce. The remaining individuals have no offspring. This approach models artificial selection where breeders deliberately select only the best individuals for the trait of interest (Muhlenbein, 1994), rather than natural selection. Nevertheless it has been used, often implicitly, to evolve models of biochemical oscillators and switches, see for example François and Hakim (2004) and Paladugu et al. (2006).

Another method in general use is tournament selection, where two or more individuals are selected at random from the population of entities and their fitness compared. Only the fittest individual in this small tournament is selected and preserved. This random pairing of individuals is repeated for as many iterations as is required to fill the next generation of individuals; in practice, this corresponds to the total size of the population. Similar to a bootstrap approach, individuals are competed and then placed back into the pool of potential competitors. For this reason, the same individuals can be picked more than once at random, and so can have more than one offspring. Diversity is maintained in this approach because unfit individuals have a non-zero,
albeit small, chance of reproducing. Tournament selection has been used in systems biology studies, e.g. Paladugu et al. (2006). This is in contrast to truncation selection, where only the fittest of the population can ever reproduce at all. If the model of selection results in a lower level of diversity, then fit individuals/solutions can take longer to evolve. This means that the genetic algorithm must be run for more generations, impacting on computational efficiency.

In either truncation or tournament selection, the fittest individuals are not guaranteed to be preserved to the next generation. The offspring of the fittest individuals could have deleterious mutations. There is the additional possibility that the fittest individuals could not have any offspring, if by chance it is never picked for a tournament (in tournament selection) or at random from the truncated population (in truncation selection). To counter this, genetic algorithms often employ an additional selection approach called **elitism**. Elitism can be used to supplement either truncation or tournament selection. Elitism is where the fittest individual (or individuals) are preserved unaltered and transferred directly into the next generation. For example, the top 1% fittest individuals of the population could immediately be replicated, without mutation, to fill 1% of the population of the next generation. Then individuals could be picked from the top 10%, including the elite, to undergo mutation and fill the remaining 99% of the next generation. This is truncation selection with elitism. Note how the elite can still have offspring which undergo mutation. This means that the elite in any one generation are preserved but can also adapt.

That the elite can still have mutated offspring also makes it difficult to assess how the fittest individuals experience mutations throughout evolution: while the elite themselves do not experience a mutational load, their offspring must undergo mutation to adapt. To test to see whether elitism can have a quantifiable impact on the evolution of networks, we compared two models, with and without elitism, in terms of the
A.2. METHODS

network size which resulted from those models. Because we find in the comparison of truncation and tournament selection that tournament selection is generally preferable to truncation selection for simulating evolution, we only compared tournament selection with and without elitism. For brevity, there is no repeat comparison of truncation selection with and without elitism. The population was fixed at 1000 individuals, with a tournament size of 2.

A.2.2 Genome Representation

Genome design can determine how that genome undergoes and experiences genetic variation (mutation and/or recombination). A common approach in systems biology GA studies is for the genome to represent simple Boolean logic functions. Usually NAND is the only function available for each ‘gene’ in the genome. Short for NOT AND, it could loosely be described as ‘A and B can’t both be true’ (see table A.1 for the truth table). No other logic functions need to be available in the genome itself, because NAND can be used to build any arbitrarily complex logical statement by connecting multiple NAND functions together. The simplest way to do this is to have every gene in the genome represent a NAND function, which takes the input from any two other genes A and B and then outputs the result of A NAND B. This is the approach taken in this study. For other examples of Boolean logic being used in genome representations, see Ofria and Wilke (2004); Kashtan et al. (2007); Crombach and Hogeweg (2008).

\[
\begin{array}{ccc}
\text{A} & \text{B} & \text{A NAND B} \\
\text{False} & \text{False} & \text{True} \\
\text{False} & \text{True} & \text{True} \\
\text{True} & \text{False} & \text{True} \\
\text{True} & \text{True} & \text{False} \\
\end{array}
\]

Table A.1: The NAND Boolean logic function
There are some subtle problems to address, even with this deceptively simple type of genome. One is how the genome can increase or decrease in size (assuming it isn’t to be fixed). If one gene in the network is deleted, what happens to the other genes which were regulated by it? Do they act as if that gene was switched off? Or are they assigned new input genes? How much can the genome size change in any one mutation, or is the genome even fixed in length?

In the model used in this paper, each gene represented a NAND logic function, taking two inputs and one output. A gene consists of three whole numbers, which give the two inputs and one output of that gene’s NAND logic function. These whole numbers refer to a ‘variable’ in the phenotypic state of the organism, which, as we are dealing with Boolean logic, could either be TRUE (‘1’) or FALSE (‘0’). Whenever the input or output of a gene changed, that whole number could change to any number from 1 to the number of genes in the genome.

The genome had a variable length. The replication ‘machinery’ could slip or stall as it scanned the parent genome, changing the length of the offspring by either skipping a gene when it slips or duplicating a gene when it stalls. Also, point mutations were possible; these could change either one of the inputs or the output of the gene. The replication machine worked by starting from the first gene and continuing to replicate until it passes the last gene. At each step, the probability of slipping was 1%. If the replication machine slipped, it would move to the next gene without copying. There was also a 1% chance of stalling. If the replication machine stalled, it would copy the gene but not move to the next gene. It is therefore possible to skip multiple genes or stall and copy a gene multiple times. If the replication machine neither slips or stalls, the gene is copied once and the replication machine moves to the next gene. Point mutations act independently of slipping and stalling (so it is possible to stall and copy a gene with a point mutation simultaneously). They also act independently
on the inputs and output of the gene’s NAND gate. If an input or output undergoes point mutation, it is replaced randomly by another state variable. There is a uniform probability of picking any new state variable present in the system. The probability of an input or output undergoing a mutation was 1%. These mutation probabilities apply to each step of replication, and so are per gene rather than per genome.

Given that the genome had a variable length, it was necessary to decide how to handle the number of variables the NAND gates of the genes could refer to. Point mutations could replace an input or output with a number from 1 up to the genome length (applied after changes from slipping or stalling). This has the effect that if the genome increased in size, the number of potential variables in the organism’s phenotype increased. If the genome size decreased, point mutations would gradually remove input/output variables beyond the genome size. Therefore, genes and the variables in the phenotypic state are not the same thing. This arrangement is demonstrated by the table in figure A.2.1a. Each row is a gene, giving a NAND function. Each gene is applied to a set of state variables, as shown in the table in figure A.2.1b. The highest input/output in this table is 14, and therefore there are 14 state variables in the organism’s phenotype. Since there are 15 genes, mutations could replace the input/output variables that the genes reference with any number from 1 to 15. If a mutation introduced a ‘15’, the number of state variables would be increased to 15. If, however, the 14 was replaced by a 10, the number of state variables would be decreased to 10. The number of genes is used as the measure of genome length as well as network size; this is because each gene adds an interaction to the network, and so the number of state variables does not directly reflect the number of connections in the network.

The emerging network for the example given here is shown in figure A.2.1c. This figure is shown using standard diagrammatic notation for Boolean logic circuits, but the details are less important than to see how genes interact. For example, genes a
and \( f \) are connected because gene \( a \) changes state variable 5 in its output, and gene \( f \) has state variable 5 as an input. Since each gene is applied in turn, the effects of some genes on the state are masked by the effects of earlier genes. Also, only the first six state variables affect fitness (see section A.2.4). Therefore, not all of the genotype effects the phenotype of the organism, although a mutation which knocks out one gene could expose a gene earlier in the sequence.

This figure only shows the results for the first six state variables. These are the six which the fitness depends on. All types of mutation had a 1\% chance per gene per genome. In this system, a larger genome implies a more interconnected network, because there are more interactions between the 6 input/output values.

### A.2.3 Systematic Bias in Mutations

While mutations may or may not have asymmetric effects under selection (for example, deleterious mutations may be more common than beneficial mutations), care should be taken that they do not inherently affect the genotype in some biased way (e.g. a bias towards increasing genome size) unless specifically intended. This can easily be checked by running a small set of preliminary simulations with the fitness function set to return a constant value (i.e. it is independent of genotype or phenotype). We call this the flat fitness function, since it gives a flat fitness landscape. If the genome size increases or decreases from its start value in a statistically significant way, then there is a bias in how mutations affect genome size. This is critical for studies where the network topology is of interest. Soyer (2007) found that if mutations leading to novel interactions were rare compared to removal of existing interactions, then modular network topology could evolve. Of course, it could be argued that new interactions are rarer than deletions, and so the bias should be included in the model. It is therefore necessary to decide whether mutational processes should be varied to examine their
A.2. METHODS

<table>
<thead>
<tr>
<th>Gene</th>
<th>Input</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>9</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>b</td>
<td>7</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>c</td>
<td>10</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>e</td>
<td>10</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>f</td>
<td>10</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>g</td>
<td>7</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>h</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>i</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>j</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>k</td>
<td>2</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>l</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>m</td>
<td>8</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>n</td>
<td>5</td>
<td>5</td>
<td>14</td>
</tr>
<tr>
<td>o</td>
<td>8</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>variable</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(b)

Figure A.2.1: An example genotype, phenotype and the emerging network. This organism had evolve to solve the ‘zeroing’ fitness function, described in section A.2.4. (a) Shows a genome in table form. Each row is a gene which is applied sequentially taking two phenotypic state variables as input and changing one state variable as output. The phenotypic state variables are shown in (b). (c) Shows the network which emerges from this genotype. The connections in the network results by gene which are shown by the corresponding gene letters. Circles show the final output to the six variables which fitness depends. Orange lines show where the variable the genes affect have been set to 1. Black lines are where the variable is set to 0. The first six variables have been set to zero, indicating maximal fitness.
effect, or kept constant and unbiased so that they can be excluded as a determining factor of topology and genome size.

This type of bias is only relevant in a model where genome size is not fixed. However, there can be other kinds of bias in mutation. Another example is if the genome is represented by real numbers. A first attempt at creating a mutational effect on real numbers might multiply them by a random number between, say 0.5 and 1.5. This seems reasonable, but is in fact biased towards lower values. Multiplying by 0.5 halves the number, whereas multiply by 1.5 increases it by only 50%. Checking this with a null fitness test would show the real numbers in the genome decreasing systematically with time.

A.2.4 The Fitness Function

The fitness function chosen for a specific GA depends substantially on the research problem to be addressed. We explored several fitness functions to see how they affected genome size. For simplicity, two fitness functions were then chosen for further study which differently affected the genome size.

One was the ‘invert’ fitness function. This is where the gene network must invert the state of the first six phenotypic state variables; i.e. change 0’s to 1’s, and 1’s to 0’s. The other was ‘zero’, where the first six phenotypic state variables should be set to all 0’s.

The proportion of the six values which match the expected output is averaged over every possible starting combination of the first six state variables. Since these variables can either be 0 or 1, there are \(2^6 = 64\) such combinations, e.g. 000000, 000001, 000010... and so on. The maximum possible fitness is 1.0, where all 6 values are correct for all possible starting combination. The minimum fitness is 0.0, where all values are wrong in every case. Examples of the correct operation of these fitness functions are given...
A.2. METHODS

in table A.2, where the starting combination of the first six variables is 001100. Note that any remaining variables, that the organism has are always set to 0, and do not affect the fitness. Therefore, they can be used by the organism in any way to achieve fitness.

Table A.2 also lists the ‘Flat’ fitness function, which was used to test for neutral effects on genome size, as explained in A.3.2. In this fitness function, all of the variables, including the first six, are ignored. The fitness is always constant.

<table>
<thead>
<tr>
<th>Fitness Function</th>
<th>Example of correct operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>001100 should be changed to 000000</td>
</tr>
<tr>
<td>Invert</td>
<td>001100 should be changed to 110011</td>
</tr>
<tr>
<td>Flat</td>
<td>N/A; fitness always the same</td>
</tr>
</tbody>
</table>

Table A.2: Fitness functions used for the model of network evolution in our study

A.2.5 Calculation of Mutational Robustness

In this study, we test to see whether large networks have a higher mutational robustness. By mutational robustness, we mean that the average effect of a mutation per gene on the six phenotypic variables which contribute to fitness is lower. To calculate this, we delete each gene individually, and inspect the phenotype to see how many of the six variables is affected by the deletion. How many of the six variables change determines how much fitness is reduced, and so averaging overall possible gene deletions gives a measure of robustness. This is similar to the definition of robustness used in (Elena et al., 2007), where the fitness effects of every mutation on the genetic background of an individual is calculated.
A.3 Results

A.3.1 Comparison of Truncation and Tournament Selection

To test how the selection model can affect the outcome of the genetic algorithm (that is, the number of generations required to find a solution), in the context of simulating network evolution, we ran the same evolutionary experiment with both tournament and truncation selection. The threshold of truncation, i.e. what percentage of the population reproduce, varied between 1% and 50%. These results are shown in figure A.3.1a. An analysis of variance (ANOVA) confirms that the effect of the selection model is statistically significant, when comparing all three truncation selection models and the tournament selection model together (number of generations ~ selection model, F=4.2, df=3, df.D=9997, p=0.002). A Tukey post-hoc test shows that it takes significantly more generations to find a genome with the maximum fitness when truncation selection has a threshold of 1%, compared to either 50% truncation and tournament selection (p<0.03 in each case). The effect is also large; roughly double the number of generations is needed between 1% threshold selection and the other models. The number of generations required did not differ significantly between the other models of selection.

However, truncation selection also affects population genetic diversity. We show this by comparing the number of distinct genotypes at 100 generations for each of the above models. Genotypes are considered distinct unless their genomes are identical (the same sequence of genes with the same input and output values). Diversity is then simply the total number of such distinct genotypes in the population of 1000 individuals. The results are shown in figure A.3.1b. With all three truncation thresholds, truncation resulted in lower diversity than tournament selection. This was confirmed with an ANOVA on the three truncation selection models and the tournament selection model (distinct genotypes ~ selection model, F=10.8, df=3, df.D=9997, p<0.001) and
A.3. RESULTS

A pairwise Tukey post-hoc test ($p<0.001$ for each truncation threshold compared to tournament selection). A Tukey test showed that the size of the effect, between tournament and 50% truncation selection, was approximately 100 more distinct genotypes in tournament selection. This is a difference of 10% of the population. Between tournament selection and the truncation selection with a 1% or threshold, the size of the effect was approximately 170 more distinct genotypes. These are clearly substantial differences in diversity.

As noted in the methods, truncation selection has previously been used to model the artificial selection of breeders, as well as simulations of network evolution. Given that it also results in lower diversity and requires more generations to reach fit individuals, we suggest that truncation selection should only be used for models of artificial selection where it is biologically appropriate.

Figure A.3.1: Comparison of genetic algorithm selection models. Bars show mean for 100 replicates, error bars are 95% confidence intervals.
Figure A.3.2: Figure showing that mutations are not biased towards increasing or decreasing genome size under the null model (i.e. fitness function gives constant fitness regardless of genotype). Points show means at each generation, while red lines show upper and lower 95% confidence limits.
A.3.2 Confirmation of Unbiased Mutations

To test whether the model design is biased towards increasing or decreasing genome size, we ran the simulations for 100 generations with a dummy fitness function which always gave a fitness of zero, regardless of genotype. There were 100 replicates. The trajectories are shown in figure A.3.2. A regression of the 100 replicates over 100 generations showed that the slope was non-significant ($F=0.42$, $R^2 = 4 \times 10^{-5}$, df=9998, $p=0.515$). Note that similar results are obtained if the fitness function always gives a fitness of one – it does not matter what the fitness value is for the confirmation of unbiased mutations as long as the fitness is independent of genotype. Therefore, mutations did not have an inherent bias towards larger or smaller genomes. An earlier design of the model of evolution did tend towards smaller networks because mutation rates were per genome, rather than per gene. This clearly could have had an important impact on the resulting network sizes, and therefore on the results.

A.3.3 Genome Size and Elitism

First, to test for the benefits of elitism in reducing the time required to run simulations, we compare the time required to solve the two fitness functions – the inverting fitness function and the zeroing fitness function – with and without elitism. The results are shown in figure A.3.3. Although the effect is small, the 'zero' fitness function requires significantly fewer generations to evolve to maximal fitness with elitism than without ($t=4.4$, df=189, $p$-value<0.001). Fewer generations naturally means a reduction in the time required to run the simulations. The size of the effect is quite small in this case, probably because the model is relatively simple, but would be expected to be larger with more complex scenarios.

Given the potential benefit of using elitism to reduce simulation time, rather than
Figure A.3.3: Elitism tends to lower the number of generations required for an individual to reach a desired fitness – in this case, maximum fitness. There benefit is non-significant for the ‘invert’ fitness function, but is for the ‘zero’ fitness function (t-test, t = 4.4, df = 189, p-value < 0.001). Error bars showing 95% confidence intervals.
A.3. RESULTS

![Graphs showing genome size of fittest individual and average genome size of population](image)

(a) Fittest individual’s genome size at fitness 1.0

(b) Fittest individuals genome size, after 100 generations

(c) Fittest individuals genome size of population, when fittest organism has fitness 1.0

(d) Average genome size of population, fixed number of generations

Figure A.3.4: Genome size, measured in various ways: for the fittest individual or averaged over the population, and at a fixed time point or when at least one individual reaches a threshold fitness (in this case maximum fitness). Error bars are 95% confidence intervals.
### APPENDIX A. METHODOLOGY: THE APPLICATION OF GENETIC ALGORITHMS IN EVOLUTIONARY SYSTEMS BIOLOGY

(a) Tukey post-hoc test of ANOVA (F-value=245, df=3, df.D=396, p-value < 0.001) for genome length of fittest individual, taken when it has fitness 1.0

<table>
<thead>
<tr>
<th></th>
<th>elitism/invert</th>
<th>elitism/zero</th>
<th>noelitism/invert</th>
<th>noelitism/zero</th>
</tr>
</thead>
<tbody>
<tr>
<td>elitism/invert</td>
<td>6.21 (***)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>elitism/zero</td>
<td>6.48 (***)</td>
<td>0.27</td>
<td>6.36 (***)</td>
<td>6.63 (***)</td>
</tr>
</tbody>
</table>

(b) Tukey post-hoc test of ANOVA (F-value=374, df=3, df.D=396, p-value < 0.001) for genome length of fittest individual, taken at generation 100

<table>
<thead>
<tr>
<th></th>
<th>elitism/invert</th>
<th>elitism/zero</th>
<th>noelitism/invert</th>
<th>noelitism/zero</th>
</tr>
</thead>
<tbody>
<tr>
<td>elitism/invert</td>
<td>6.2 (***)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>elitism/zero</td>
<td>0.15</td>
<td>6.4 (***)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>noelitism/invert</td>
<td>12.5 (***)</td>
<td>6.3 (***)</td>
<td>12.7 (***)</td>
<td></td>
</tr>
</tbody>
</table>

(c) Tukey post-hoc test of ANOVA (F-value=220, df=3, df.D=396, p-value < 0.001) of average genome length in population when fittest organism has fitness 1.0

<table>
<thead>
<tr>
<th></th>
<th>elitism/invert</th>
<th>elitism/zero</th>
<th>noelitism/invert</th>
<th>noelitism/zero</th>
</tr>
</thead>
<tbody>
<tr>
<td>elitism/invert</td>
<td>11.4 (***)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>elitism/zero</td>
<td>0.06</td>
<td>11.5 (***)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>noelitism/invert</td>
<td>12.0 (***)</td>
<td>-0.55</td>
<td>12.0 (***)</td>
<td></td>
</tr>
</tbody>
</table>

(d) Tukey post-hoc test of ANOVA (F-value=640, df=3, df.D=396, p-value < 0.001) of average genome length in population, taken at generation 100

<table>
<thead>
<tr>
<th></th>
<th>elitism/invert</th>
<th>elitism/zero</th>
<th>noelitism/invert</th>
<th>noelitism/zero</th>
</tr>
</thead>
<tbody>
<tr>
<td>elitism/invert</td>
<td>11.4 (***)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>elitism/zero</td>
<td>0.06</td>
<td>11.5 (***)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>noelitism/invert</td>
<td>12.0 (***)</td>
<td>-0.55</td>
<td>12.0 (***)</td>
<td></td>
</tr>
</tbody>
</table>

Table A.3: Tukey post-hoc tests for various approaches to analysing genome size, each based upon one-way ANOVA with all possible combinations of elitism and fitness function in a single factor. When analysing at the population level or at the time the fittest individual first reaches maximal fitness, elitism does not significantly affect the genome size for the same fitness function. However, analyzing network size for the fittest individual at a fixed generation number is affected by elitism. Note that the only significant comparison between elitism and non-elitism methods for the same fitness function are found in figure b; elitism has affected the results in this case.
A.3. RESULTS

selection without elitism, we now compare the effect of the two fitness functions on genome size, with and without elitism. The elite is just one individual in our study. The results are shown in figure A.3.4. We can analyse the genome size of the fittest individual (i.e. the dominant organism), or take the average genome size of the population. We can also compare the genome size once the fittest individual has reached a given fitness, or at a fixed generation time. ANOVA and Tukey tests were performed for each scenario, and are shown A.3, and correspond to the sub-figures of figure A.3.4. These confirm that, in all these measures, the invert and zero fitness functions differ in their effect on genome size. Elitism also didn’t affect the result significantly if either the population average of genome size is used (sub-figure d) or the genome size is taken at the time the fitness of the fittest individual first reaches the maximum (sub-figure a), or both (sub-figure c).

However, when a fixed number of generations was used, elitism affected the result significantly (see Tukey test in table A.3b). In the case of the ‘zero’ fitness function, the average genome size of the population continued to increase after the fittest individual had reached maximum fitness. But because the fittest individual did not undergo mutation and had reached maximum fitness and so was no longer being replaced, the genome size was no longer changing. If elitism is being used, it is better to avoid analysing the dominant organisms at a fixed generation time that exceeds the upper limit of fitness (it is common for fitness to have a maximum value in genetic algorithms).

A.3.4 Genome Size is Under Selection

Referring back to the results from the flat fitness function in section A.3.2, we know that that there is no inherent bias in the effect of mutation on genome size. However, both fitness functions result in genome sizes which are significantly different from their start value (figure A.3.4). The genome was reduced in the case of the invert fitness
function (one-sampled t-test, $t = -16.2$, df = 199, p-value $< 0.001$) and increased in the case of the zero fitness function (one-sampled t-test, $t = 20.0714$, df = 199, p-value $< 0.001$), taking results with and without elitism together. These effects are therefore in different directions, with genomes growing under the zero fitness function and shrinking under the invert fitness function.

Without elitism, the genome size of the fittest individual continues to increase under the zero fitness function, even after the fitness of the fittest is maximal. A t-test was performed to compare the genome sizes of the fittest organism without elitism at maximum fitness and at generation 100. Note that maximum fitness of the fittest individual is on average reached around generation 36 (see figure A.3.3). This t-test was significant ($t = -11.3$, df = 150, $p<0.001$), confirming that even at maximum fitness the genome size of the dominant organism tends to increase under this fitness function.

Therefore we can conclude that larger networks are being selected for in the case of the zero fitness function. Why? This fitness function requires each value to be set to zero, regardless of how it is initially set. By having redundant connections in the gene network, it is possible for a value to be set to zero multiple times. If a mutation deletes one gene, the output value will still be set to zero by another gene. This is a form of mutational robustness. To check this, we calculated the number of possible point mutations which reduced the fitness of the organism, both for the dominant organism when maximum fitness is first reached, and at 100 generations. A paired t-test shows that proportionately fewer mutations reduced fitness in the later 100 generation fittest genotype ($t = -12.1$, df = 99, $p<0.001$) than in the genotype that first obtained maximum fitness, with a mean difference of 17.7% of mutations. These results are summarized in figure A.3.5.

In contrast, the invert fitness function cannot be made robust in this way. Each input value must be inverted precisely once in this function. Multiple connections to
Figure A.3.5: Robustness to mutations at maximum fitness and 100 generations, for the two fitness functions.
the first six state variables would be deleterious, or at least cannot offer redundant forms of the invert function. 'Code bloat' could still exist in the form of genes, as well as variables in the phenotype, which are not connected to the six variables on which the fitness depends. However, this is not the case. The formation of code bloat is subject to selection, even when the fitness of the organism is not explicitly penalized for large genome or network size.

A.4 Discussion

Much of the genetic algorithm (GA) literature is in the discipline of computational science, where GA methodologies have been developed and studied in the context of developing efficient machine learning and problem solving tools. The systems biology community has naturally made use of genetic algorithms to simulate evolution (examples in Decard and Sauro 2004; Paladugu et al. 2006; Parter et al. 2007) and so we have sought to examine how these methodologies fit into modeling instead of problem solving. This is important because GA have not been developed with biological relevance in mind; methodologies borrow terminology and ideas from biology but may become something quite different; we gave the example of niching genetic algorithms which do not really correspond to ecological niches, in section 1.3.2.

As part of our examination of GA methods for biological simulations, we studied how selection and fitness criteria can affect genome size to test the importance of GA structure and to provide a methodological framework for simulating the evolution of networks with genetic algorithms. We showed how the model of selection can affect the timescale required to adapt to the fitness function (figure A.3.1a) as well as the genetic diversity of the population (figure A.3.1b), and therefore cannot be neglected as a detail of the genetic algorithm. Before going on to test for the effect of fitness
function and elitism on genome size, we confirmed that mutations do not systematically bias genome size under a flat fitness function (figure A.3.2).

We then tested the effect of two different fitness functions on genome size, with and without elitism. Elitism, which is only a computational technique rather than a biologically meaningful mode of selection, biases the analysis of the dominant organism once fitness is maximized (figure A.3.4).

Without the artifact of elitism, the genome of the dominant organism can continue to change even after fitness is maximized. In the case of one of the two fitness functions, larger genomes were selected for, conferring greater mutational robustness. This continued after maximum fitness was reached.

This mutational robustness was likely to be due to redundant solutions to the fitness function, and selected for, since without any differential selection network size did not significantly change over time. Code bloat was also not seen with a fitness function where redundant gene interactions with the phenotypic traits associated with fitness (i.e. the first six variables) would be deleterious. This was the case even though genes could exist which did not interact with these phenotypic traits; variables could exist which did not directly affect fitness, and so code bloat was possible. Yet it was not observed. This may be because genes which in their genetic background are not involved in fitness could undergo a subsequent mutation which establishes an interaction with one of the phenotypic traits in fitness. If that interaction is deleterious, then such non-functional genes may be selected out. Therefore, code bloat, whether non-functional or redundant in its genetic background, might be selected for or selected against depending upon how mutations in the code bloat interact with the phenotype. We can also firmly conclude that code bloat is not an artifact of the genetic algorithm itself, and that an explicit penalty for genome size is not always necessary for code bloat to be under negative selection.
APPENDIX A. METHODOLOGY: THE APPLICATION OF GENETIC ALGORITHMS IN EVOLUTIONARY SYSTEMS BIOLOGY

Mutational robustness is known to be lower with a large population and small mutation rates, even when the genome is fixed in length (Elena et al., 2007; Elena and Sanjuán, 2008), and so using large populations and small mutation rates may be helpful if the evolution of small genomes/networks is desirable. Large populations and small mutation rates can be more realistic (Drake et al., 1998). The main disadvantage is that larger models are more computationally expensive.

We can formulate a set of guidelines for modeling evolution with genetic algorithms based upon the results of this methodology study:

1. Confirm that a genetic algorithm technique is biologically meaningful before using it in a model. While inspired by biology, genetic algorithms are designed for optimisation of engineering problems.

2. Use truncation selection to model artificial selection and tournament selection (or other) to model natural selection.

3. Avoid using elitism if possible, but if elitism is desirable for computational efficiency, end the simulation when the fittest individual has reached an upper limit of fitness rather than at a fixed number of generations. Alternatively, include the whole population in the network analysis.

4. Check for systematic effects of mutation on the genome using a flat fitness function.

5. All aspects of the genetic algorithm can potentially affect the results, therefore ensure others can repeat your work by publishing details of the genetic algorithm used.

It is hoped that, by demonstrating a methodological framework for simulating the evolution of networks with genetic algorithms, modelers can avoid potential pitfalls
that can make genetic algorithms perform poorly or produce erroneous results.
APPENDIX A. METHODOLOGY: THE APPLICATION OF GENETIC ALGORITHMS IN EVOLUTIONARY SYSTEMS BIOLOGY
Appendix B

Kinetics for Two Simple Robust Oscillators

B.1 Kinetics of a robust two-protein oscillator

These are the kinetics of the robust oscillator shown in figure 2.3.4b. \([B_p]\) has an inhibitory effect on the phosphorylation of \([A]\) (the formation of \([A_p]\) ). In the kinetic model used in this, a protein could inhibit (or activate) multiple reactions, but each reaction can be affected by at most one inhibitor and one activator, a simplification described in section 2.2.1. For this reason, the kinetic parameters were named after the reaction they affect, rather than the effector, so as to be uniquely identified. Hence the kinetic parameter for the inhibition of the formation of \([A_p]\) by \([B_p]\) is \(k_{A_p}^I\). Effectors are in bold, to differentiate them more easily from the substrates and products of reactions.
APPENDIX B. KINETICS FOR TWO SIMPLE ROBUST OSCILLATORS

\[
\begin{align*}
\frac{d[A]}{dt} &= + \frac{k_{cat}^A \cdot [B_p] \cdot [A_p]}{K_m^A + [A_p]} - \frac{k_{cat}^A \cdot [B] \cdot [A]}{K_m^A + K_m^A \cdot \frac{[B_p]}{k_f} + [A]} \\
\frac{d[A_p]}{dt} &= - \frac{k_{cat}^A \cdot [B_p] \cdot [A_p]}{K_m^A + [A_p]} + \frac{k_{cat}^A \cdot [B] \cdot [A]}{K_m^A + K_m^A \cdot \frac{[B_p]}{k_f} + [A]} \\
\frac{d[B]}{dt} &= + \frac{k_{cat}^B \cdot [A] \cdot [B_p]}{K_m^B + K_m^B \cdot \frac{[B]}{k_f} + [B_p]} - \frac{[B_p] \cdot [B_p]}{K_m^B + [B]} \\
\frac{d[B_p]}{dt} &= - \frac{k_{cat}^B \cdot [A] \cdot [B_p]}{K_m^B + K_m^B \cdot \frac{[B]}{k_f} + [B_p]} + \frac{[B_p] \cdot [B_p]}{K_m^B + [B]}
\end{align*}
\]

Kinetic Parameters:

\[
\begin{array}{|c|c|c|c|}
\hline
\hline
K_m & 0.50 & 1.1 & 0.050 & 0.040 \\
k_{cat} & 0.24 & 1.5 & 3.6 & 1.4 \\
k_f & - & 0.055 & 3.6 & - \\
\hline
\end{array}
\]

B.2 Kinetics of a robust three-protein oscillator

These are the kinetics for oscillator shown in figure 2.3.4c.
B.2. KINETICS OF A ROBUST THREE-PROTEIN OSCILLATOR

\[
\frac{d[A]}{dt} = + \frac{k_{Acat}^d \cdot [B] \cdot [A_p]}{K_{Am}^d + [A_p]} - \frac{k_{Ap}^d \cdot [A]}{K_{Am}^p + [A]}
\]

\[
\frac{d[A_p]}{dt} = - \frac{k_{Acat}^d \cdot [B] \cdot [A_p]}{K_{Am}^d + [A_p]} + \frac{k_{Ap}^d \cdot [A]}{K_{Am}^p + [A]}
\]

\[
\frac{d[B]}{dt} = + \frac{k_{Bcat}^d \cdot [C] \cdot [B_p]}{K_{Bm}^d + [B_p]} - \frac{k_{Bp}^d \cdot [C_p] \cdot [B]}{K_{Bm}^p + [B]}
\]

\[
\frac{d[B_p]}{dt} = - \frac{k_{Bcat}^d \cdot [C] \cdot [B_p]}{K_{Bm}^d + [B_p]} + \frac{k_{Bp}^d \cdot [C_p] \cdot [B]}{K_{Bm}^p + [B]}
\]

\[
\frac{d[C]}{dt} = + \frac{k_{Ccat}^d \cdot [A_p] \cdot [C_p]}{K_{Cm}^d + [C_p]} - \frac{k_{Cp}^d \cdot [C_p] \cdot [C]}{K_{Cm}^p + [C]}
\]

\[
\frac{d[C_p]}{dt} = - \frac{k_{Ccat}^d \cdot [A_p] \cdot [C_p]}{K_{Cm}^d + [C_p]} + \frac{k_{Cp}^d \cdot [C_p] \cdot [C]}{K_{Cm}^p + [C]}
\]
### Kinetic Parameters:

<table>
<thead>
<tr>
<th></th>
<th>[A]</th>
<th>[Aₚ]</th>
<th>[B]</th>
<th>[Bₚ]</th>
<th>[C]</th>
<th>[Cₚ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kₘ</td>
<td>0.036</td>
<td>0.49</td>
<td>0.13</td>
<td>0.063</td>
<td>0.87</td>
<td>0.052</td>
</tr>
<tr>
<td>kₑ Thông số</td>
<td>3.1</td>
<td>0.10</td>
<td>0.018</td>
<td>0.13</td>
<td>23</td>
<td>1.54</td>
</tr>
</tbody>
</table>
Appendix C

Avida Configuration

# This file includes all the basic run-time defines for Avida.
# For more information, see doc/config.html

VERSION_ID 2.9.0  # Do not change this value.

### GENERAL_GROUP ###
# General Settings
ANALYZE_MODE 0  # 0 = Disabled
    # 1 = Enabled
    # 2 = Interactive
VIEW_MODE 1  # Initial viewer screen
CLONE_FILE  # data/clone1  # Clone file to load
VERBOSITY 1  # 0 = No output at all
    # 1 = Normal output
    # 2 = Verbose output, detailing progress
    # 3 = High level of details, as available
    # 4 = Print Debug Information, as applicable

### ARCH_GROUP ###
# Architecture Variables
WORLD_X 60  # Width of the Avida world
WORLD_Y 60  # Height of the Avida world
APPENDIX C. AVIDA CONFIGURATION

WORLD_Z 1 # Depth of the Avida world
WORLD_GEOMETRY 2 # 1 = Bounded Grid
# 2 = Torus
# 3 = Clique
# 4 = Hexagonal grid
# 5 = Lattice

RANDOM_SEED 0 # Random number seed (0 for based on time)

HARDWARE_TYPE 0 # 0 = Original CPUs
# 1 = New SMT CPUs
# 2 = Transitional SMT
# 3 = Experimental CPU
# 4 = Gene Expression CPU

SPECULATIVE 1 # Enable speculative execution

TRACE_EXECUTION 0 # Trace the execution of all organisms in the population (default=off, SLOW!)

BCAST_HOPS 1 # Number of hops to broadcast an alarm

ALARM_SELF 0 # Does sending an alarm move sender IP to alarm label?
# 0 = no
# 1 = yes

IO_EXPIRE 1 # Is the expiration functionality of 'expire' I/O instructions enabled?

### CONFIG_FILE_GROUP ###
# Configuration Files

DATA_DIR data # Directory in which config files are found
INST_SET instset classic.cfg # File containing instruction set
INST_SET_FORMAT 0 # Instruction set file format.
# 0 = Default
# 1 = New Style

EVENT_FILE events.cfg # File containing list of events during run
ANALYZE_FILE analyze.cfg # File used for analysis mode
ENVIRONMENT_FILE environment.cfg # File that describes the environment

START_CREATURE default classic.org #default classic.org # Organism to seed the soup

### DEME_GROUP ###
# Demes and Germlines

NUM_DEMES 1 # Number of independent groups in the population (default=1).
DEMES_USE_GERMLINE 0
    # Whether demes use a distinct germline (default=0).

DEMES_PREVENT_STERILE 0
    # Whether to prevent sterile demes from replicating (default=0 or no).

DEMES_RESET_RESOURCES 0
    # Reset resources in demes on replication.
    # 0 = reset both demes
    # 1 = reset target deme
    # 2 = deme resources remain unchanged

DEMES_REPLICATE_SIZE 1
    # Number of identical organisms to create or copy from the source deme to the target deme (default=1).

LOG_DEMES_REPLICATE 0
    # Log deme replications. 0/1 (off/on)

DEMES_REPLICATE_LOG_START 0
    # Update at which to start logging deme replications

DEMES_PROB_ORG_TRANSFER 0.0
    # Probability of an organism being transferred from the source deme to the target deme (default=0.0).

DEMES_ORGANISM_SELECTION 0
    # How organisms are selected for transfer from source to target during deme replication.
    # 0=random with replacement (default).
    # 1=sequential.

DEMES_ORGANISM_PLACEMENT 0
    # How organisms are placed during deme replication.
    # 0=cell-array middle (default).
    # 1=deme center.
    # 2=random placement.
    # 3=sequential.

DEMES_ORGANISM_FACING 0
    # How organisms are facing during deme replication.
    # 0=unchanged (default).
    # 1=northwest.
    # 2=random.

DEMES_MAX_AGE 500
    # The maximum age of a deme (in updates) to be used for age-based replication (default=500).

DEMES_MAX_BIRTHS 100
    # The maximum number of births that can occur within a deme; used with birth-count replication (default=100).
APPENDIX C. AVIDA CONFIGURATION

DEMES_MIM_EVENTS_KILLED_RATIO 0.7  # Minimum ratio of events killed required for event period to be a success.
DEMES_MIM_SUCCESSFUL_EVENT_PERIODS 1  # Minimum number of consecutive event periods that must be a success.
GERMINE_COPY_MUT 0.0075  # Prob. of copy mutations occurring during germline replication (default = 0.0075).
GERMINE_INS_MUT 0.05  # Prob. of an insertion mutation occurring during germline replication (default = 0.05).
GERMINE_DEL_MUT 0.05  # Prob. of a deletion mutation occurring during germline replication (default = 0.05).
DEMES_REPLICATE_CPU_CYCLES 0.0 unused  # Replicate a deme immediately after it has used this number of cpu cycles, normalized by number of orgs in deme (0 = OFF).
DEMES_REPLICATE_TIME 0.0 unused  # Replicate a deme immediately after it has used this number of cpu cycles, normalized by number of orgs in deme and organism merit (0 = OFF).
DEMES_REPLICATE_BIRTHS 0  # Replicate a deme immediately after it has produced this many offspring (0 = OFF).
DEMES_REPLICATE_ORGS 0  # Replicate a deme immediately once it reaches a certain number of organisms (0 = OFF).
DEMES_REPLICATION_ONLY RESETS 0  # Kin selection mode. Deme replication really:
DEMES_MIGRATION_RATE 0.0  # Probability of an offspring being born in a different deme.
DEMES_MIGRATION_METHOD 0  # How do we choose what demes an org may land in when it migrates?
DEMES_NUM_X 0  # 0=all other demes
DEMES_SEED_METHOD 0  # 1=eight adjacent neighbors
DEMES_NUM_Y 0  # 2=two adjacent demes in list
DEMES_NUM_Z 0  # Simulated number of demes in X dimension.
DEMES_SEED_METHOD 0  # 0=maintain old consistency
DEMES_DIVIDE_METHOD 0

# 1=new method using genotypes
# Deme divide method. Only works with

DEMES_SEED_METHOD 1

# 0=replace and target demes
# 1= replace target deme, reset source deme to
# founders

# 2=replace target deme, leave source deme
# unchanged

DEMES_DEFAULT_GERMLINE_PROPENSITY 0.0

# Default germline propensity of organisms in
deme.

# For use with DEMES_DIVIDE_METHOD 2.

DEMES_FOUNDER_GERMLINE_PROPENSITY -1.0

# Default germline propensity of founder
organisms in deme.

# For use with DEMES_DIVIDE_METHOD 2.

# <0 = OFF

DEMES_PREFEIER_EMPTY 0

# Give empty demes preference as targets of
deme replication?

### REPRODUCTION_GROUP ###

# Birth and Death

BIRTH_METHOD 4

# Which organism should be replaced on birth?
# 0 = Random organism in neighborhood
# 1 = Oldest in neighborhood
# 2 = Largest Age/Merit in neighborhood
# 3 = None (use only empty cells in neighborhood)
# 4 = Random from population (Mass Action)
# 5 = Oldest in entire population
# 6 = Random within deme
# 7 = Organism faced by parent
# 8 = Next grid cell (id+1)
# 9 = Largest energy used in entire population
# 10 = Largest energy used in neighborhood

PREFFER_EMPTY 1

# Give empty cells preference in offspring placement?

ALLOW_PARENT 1

# Allow births to replace the parent organism?

DEATH_METHOD 0

# 0 = Never die of old age.

# 1 = Die when inst executed = AGE_LIMIT (+deviation)
# 2 = Die when inst executed = length * AGE_LIMIT (+dev)

AGE_LIMIT 20

# Modifies DEATH_METHOD

AGE_DEVIATION 2

# Creates a distribution around AGE_LIMIT
APPENDIX C. AVIDA CONFIGURATION

ALLOC_METHOD 0  # (Original CPU Only)
# 0 = Allocated space is set to default instruction.
# 1 = Set to section of dead genome (Necrophilia)
# 2 = Allocated space is set to random instruction.

DIVIDE_METHOD 1  # 0 = Divide leaves state of mother untouched.
# 1 = Divide resets state of mother
#   (after the divide, we have 2 children)
# 2 = Divide resets state of current thread only
#   (does not touch possible parasite threads)
# 3 = Divide resets mother stats, but not state.
# 4 = 3 + child inherits mother registers and stack values.

EPIGENETIC_METHOD 0  # Inheritance of state information other than genome
# 0 = none
# 1 = offspring inherits registers and stacks of first thread
# 1 = parent maintains registers and stacks of first thread
# 1 = offspring and parent keep state information

INJECT_METHOD 0  # 0 = Leaves the parasite thread state untouched.
# 1 = Resets the calling thread state on inject

GENERATION_INC_METHOD 1  # 0 = Only the generation of the child is increased on divide.
# 1 = Both the generation of the mother and child are increased on divide (good with DIVIDE_METHOD 1).

RESET_INPUTS_ON_DIVIDE 0  # Reset environment inputs of parent upon successful divide.

REPRO_METHOD 1  # Replace existing organism: 1=yes

### RECOMBINATION_GROUP ###
# Sexual Recombination and Modularity
RECOMBINATION_PROB 1.0  # probability of recombination in div-sex
MAX_BIRTH_WAIT_TIME -1  # Updates incipient orgs can wait for crossover
MODULE_NUM 0  # number of modules in the genome
CONT_REC_REGS 1  # are (modular) recombination regions continuous
CORESPOND_REC_REGS 1  # are (modular) recombination regions swapped randomly
# or with corresponding positions?
TWO_FOLD_COST SEX 0  # 1 = only one recombined offspring is born.
# 2 = both offspring are born
SAME_LENGTH SEX 0  # 0 = recombine with any genome
# 1 = only recombine w/ same length
### DIVIDE_GROUP ###

# Divide Restrictions

CHILD_SIZE_RANGE 2.0  # Maximal differential between child and parent sizes.
MIN_COPIED_LINES 0.5  # Code fraction which must be copied before divide.
MIN_EXE_LINES 0.5  # Code fraction which must be executed before divide.
MIN GENOME_SIZE 0  # Minimum number of instructions allowed in a genome. 0 = OFF
MAX GENOME_SIZE 0  # Maximum number of instructions allowed in a genome. 0 = OFF

REQUIRE_ALLOCATE 1  # (Original CPU Only) Require allocate before divide?
REQUIRED_TASK -1  # Task ID required for successful divide.
IMMUNITY_TASK -1  # Task providing immunity from the required task.
REQUIRED_REACTION -1  # Reaction ID required for successful divide.
REQUIRED_BONUS 0.0  # Required bonus to divide.
Implicit_Repro_Bonus 0  # Call Inst_Repro to divide upon achieving this bonus. 0 = OFF

Implicit_Repro_CPU_Cycles 0  # Call Inst_Repro after this many cpu cycles. 0 = OFF
Implicit_Repro_Time 0  # Call Inst_Repro after this time used. 0 = OFF
Implicit_Repro_End 0  # Call Inst_Repro after executing the last instruction in the genome.
Implicit_Repro_Energy 0.0  # Call Inst_Repro if organism accumulates this amount of energy.

### MUTATION_GROUP ###

# Mutations

POINT_MUT_PROB 0.0  # Mutation rate (per-location per update)
COPY_MUT_PROB 0.0075  # Mutation rate (per copy)
COPY_INS_PROB 0.0  # Insertion rate (per copy)
COPY_DEL_PROB 0.0  # Deletion rate (per copy)
COPY_UNIFORM_PROB 0.0  # Uniform mutation probability (per copy)
  # Randomly applies any of the three classes of mutations (ins, del, point).
COPY_SLIP_PROB 0.0  # Slip rate (per copy)
DIV_MUT_PROB 0.0  # Mutation rate (per site, applied on divide)
DIV_INS_PROB 0.0  # Insertion rate (per site, applied on divide)
DIV_DEL_PROB 0.0  # Deletion rate (per site, applied on divide)
DIV_UNIFORM_PROB 0.0  # Uniform mutation probability (per site, applied on divide)
APPENDIX C. AVIDA CONFIGURATION

# - Randomly applies any of the three classes of mutations (ins, del, point).
DIV_SLIP_PROB 0.0  # Slip rate
DIVIDE_MUT_PROB 0.0  # Mutation rate (per divide)
DIVIDE_INS_PROB 0.05  # Insertion rate (per divide)
DIVIDE_DEL_PROB 0.05  # Deletion rate (per divide)
DIVIDE_UNIFORM_PROB 0.0  # Uniform mutation probability (per divide)
# - Randomly applies any of the three classes of mutations (ins, del, point).
DIVIDE_SLIP_PROB 0.0  # Slip rate (per divide) - creates large deletions/duplications
INJECT_INS_PROB 0.0  # Insertion rate (per site, applied on inject)
INJECT_DEL_PROB 0.0  # Deletion rate (per site, applied on inject)
INJECT_MUT_PROB 0.0  # Mutation rate (per site, applied on inject)
SLIP_FILL_MODE 0  # Fill insertions from slip mutations with 0=duplication, 1=duplication, 2=random, 3=scrambled
PARENT_MUT_PROB 0.0  # Per-site, in parent, on divide
SPECIAL_MUT_LINE 1  # If this is >= 0, ONLY this line is mutated
META_COPY_MUT 0.0  # Prob. of copy mutation rate changing (per gen)
META_STD_DEV 0.0  # Standard deviation of meta mutation size.
MUT_RATE_SOURCE 1  # 1 = Mutation rates determined by environment.
# 2 = Mutation rates inherited from parent.
MIGRATION_RATE 0.0  # Uniform probability of offspring migrating to a new deme.

### REVERSION_GROUP ###
# Mutation Reversion
# These slow down avida a lot, and should be set to 0.0 normally.
REVERT_FATAL 0.0  # Should any mutations be reverted on birth?
REVERT_DETREMENTAL 0.0  # 0.0 to 1.0; Probability of reversion.
REVERT_NEUTRAL 0.0  #
REVERT_BENEFICIAL 0.0  #
STERILIZE_FATAL 0.0  # Should any mutations clear (kill) the organism?
STERILIZE_DETREMENTAL 0.0  #
STERILIZE_NEUTRAL 0.0  #
STERILIZE_BENEFICIAL 0.0  #
FAIL_IMPLPLICIT 0  # Should copies that failed *not* due to mutations
# be eliminated?
NEUTRAL_MAX 0.0  # The percent beneficial change from parent fitness
# to be considered neutral.

236
NEUTRAL_MIN 0.0  # The percent deleterious change from parent fitness to be considered neutral.

### TIME_GROUP ###
# Time Slicing
AVE_TIME_SLICE 30  # Ave number of insts per org per update
SLICING_METHOD 1  # 0 = CONSTANT: all organisms get default ...
  # 1 = PROBABILISTIC: Run _prob_ proportional to merit.
  # 2 = INTEGRATED: Perfectly integrated deterministic.
  # 3 = Demeprobabalistic, each deme gets the same number of
  # CPU cycles, which are awarded probabilistically within each deme.
  # 4 = ProbDemeProbabalistic, each deme gets CPU cycles proportional to its living population size, which are
  # awarded probabilistically within each deme.
  # 5 = CONSTANT BURST: all organisms get default, in
SLICING_BURST_SIZE 1  # Sets the scheduler burst size, when supported.
BASE_MERIT_METHOD 0 # 4  # 0 = Constant (merit independent of size)
  # 1 = Merit proportional to copied size
  # 2 = Merit prop. to executed size
  # 3 = Merit prop. to full size
  # 4 = Merit prop. to min of executed or copied size
  # 5 = Merit prop. to sqrt of the minimum size
  # 6 = Merit prop. to num times MERIT_BONUS_INST is in
BASE_CONST_MERIT 100  # Base merit when BASE_MERIT_METHOD set to 0
DEFAULT_BONUS 1.0  # Initial bonus before any tasks
MERIT_DEFAULT_BONUS 0  # Scale the merit of an offspring by this default bonus
# rather than the accumulated bonus of the parent? 0 = off
MERIT_BONUS_INST 0  # in BASE_MERIT_METHOD 6, this sets which instruction
  # counts
  # (-1 = none, 0 = First in INST_SET.)
MERIT_BONUS_EFFECT 0  # in BASE_MERIT_METHOD 6, this sets how much merit is
  # earned
  # per instruction (-1 = penalty, 0 = no effect.)
FITNESS_METHOD 0  # 0 = default, >1 = experimental
FITNESS_COEFF_1 1.0  # 1st FITNESS_METHOD parameter
FITNESS_COEFF_2 1.0  # 2nd FITNESS_METHOD parameter
APPENDIX C. AVIDA CONFIGURATION

FITNESS_Valley 0
# in BASE_MERIT_METHOD 6, this creates valleys from
# FITNESS_Valley_START to FITNESS_Valley_STOP
# 0 = off, 1 = on

FITNESS_Valley_START 0
# if FITNESS_Valley = 1, orgs with num_key_instructions
# from FITNESS_Valley_START to FITNESS_Valley_STOP
# get fitness 1 (lowest)

FITNESS_Valley_STOP 0
# if FITNESS_Valley = 1, orgs with num_key_instructions
# from FITNESS_Valley_START to FITNESS_Valley_STOP
# get fitness 1 (lowest)

MAX_CPU_THREADS 1
# Number of Threads a CPU can spawn

THREAD_SLICING_METHOD 0
# Formula for and organism's thread slicing
# (num_threads−1) * THREAD_SLICING_METHOD + 1
# 0 = One thread executed per time slice.
# 1 = All threads executed each time slice.

NO_CPU_CYCLE_TIME 0
# Don't count each CPU cycle as part of gestation time

MAX_LABEL_EXECUTE_SIZE 1
# Max nops marked as executed when labels are used

MERIT_GIVEN 0.0
# Fraction of merit donated with 'donate' command

MERIT_RECEIVED 0.0
# Multiplier of merit given with 'donate' command

MAXDonate_KIN_DIST -1
# Limit on distance of relation for donate; -1=no max

MAXDonate_EDIT_DIST -1
# Limit on genetic (edit) distance for donate; -1=no max

MIN_GBDonate_THRESHOLD -1
# threshold green beard donates only to orgs above this
# donation attempt threshold; -1=no thresh

DONATE_THRESH_QUANTA 10
# The size of steps between quanta donate thresholds

MAX_DONATES 1000000
# Limit on number of donates organisms are allowed.

PRECALC_PHENOTYPE 0
# 0 = Disabled
# 1 = Assign precalculated merit at birth (unlimited
# resources only)
# 2 = Assign precalculated gestation time
# 3 = Assign precalculated merit AND gestation time.
# Fitness will be evaluated for organism based on these
# settings.

FASTFORWARD_UPDATES 0
# Fast-forward if the average generation has not changed
# in this many updates. (0 = off)

FASTFORWARD_NUM_ORGS 0
# Fast-forward if population is equal to this

### GENEALOGY_GROUP ###
# Genealogy

TRACK_MAIN_LINEAGE 1
# Keep all ancestors of the active population?
# 0=no, 1=yes, 2=yes,w/sexual population

238
THRESHOLD 3  # Number of organisms in a genotype needed for it
    # to be considered viable.
GENOTYPE_PRINT 0  # 0/1 (off/on) Print out all threshold genotypes?
GENOTYPE_PRINT_DOM 0  # Print out a genotype if it stays dominant for
    # this many updates. (0 = off)
SPECIES_THRESHOLD 2  # max failure count for organisms to be same species
SPECIES_RECORDING 0  # 1 = full, 2 = limited search (parent only)
SPECIES_PRINT 0  # 0/1 (off/on) Print out all species?
TEST_CPU_TIME_MOD 20  # Time allocated in test CPUs (multiple of length)

### LOG_GROUP ###
# Log Files
LOG_CREATURES 0  # 0/1 (off/on) toggle to print file.
LOG_GENOTYPES 0  # 0 = off, 1 = print ALL, 2 = print threshold ONLY.
LOG_THRESHOLD 0  # 0/1 (off/on) toggle to print file.
LOG_SPECIES 0  # 0/1 (off/on) toggle to print file.

### LINEAGE_GROUP ###
# Lineage
# NOTE: This should probably be called "Clade"
# This one can slow down aida a lot. It is used to get an idea of how
# often an advantageous mutation arises, and where it goes afterwards.
# Lineage creation options are. Works only when LOG_LINEAGES is set to 1.
# 0 = manual creation (on inject, use successive integers as lineage labels).
# 1 = when a child's (potential) fitness is higher than that of its parent.
# 2 = when a child's (potential) fitness is higher than max in population.
# 3 = when a child's (potential) fitness is higher than max in dom. lineage
# and the child is in the dominant lineage, or (2)
# 4 = when a child's (potential) fitness is higher than max in dom. lineage
# (and that of its own lineage)
# 5 = same as child's (potential) fitness is higher than that of the
# currently dominant organism, and also than that of any organism
# currently in the same lineage.
# 6 = when a child's (potential) fitness is higher than any organism
# currently in the same lineage.
# 7 = when a child's (potential) fitness is higher than that of any
# organism in its line of descent
LOG_LINEAGES 0  #
LINEAGE_CREATION_METHOD 0  #
APPENDIX C. AVIDA CONFIGURATION

### ORGANISM_NETWORK_GROUP ###

# Organism Network Communication

NET_ENABLED 0 # Enable Network Communication Support
NET_DROP_PROB 0.0 # Message drop rate
NET_MUT_PROB 0.0 # Message corruption probability
NET_MUT_TYPE 0 # Type of message corruption. 0 = Random Single Bit, 1 = Always Flip Last

NET_STYLE 0 # Communication Style. 0 = Random Next, 1 = Receiver Facing

### ORGANISM_MESSAGING_GROUP ###

# Organism Message-Based Communication

MESSAGE_TYPE 0 # Messaging Style. 0 = Receiver Facing, 1 = Broadcast
MESSAGE_BCAST_RADIUS 1 # Broadcast message radius (cells)
ORGANISMS_REMEMBER_MESSAGES 1 # Does an organism remember all messages it has sent or received? 0 = false, 1 = true (default)
MESSAGE_QUEUE_SIZE -1 # Maximum number of unretrieved messages an organism can store (-1 for no limit is the default)
MESSAGE_QUEUE_BEHAVIOR_WHEN_FULL 0 # 0 = Drop incoming message (default), 1 = Drop oldest unretrieved message

### BUY_SELL_GROUP ###

# Buying and Selling Parameters

SAVE_RECEIVED 0 # Enable storage of all inputs bought from other orgs
BUY_PRICE 0 # price offered by organisms attempting to buy
SELL_PRICE 0 # price offered by organisms attempting to sell

### HOARD_RESOURCE_GROUP ###

# Resource Hoarding Parameters

USE_RESOURCE_BINS 0 # Enable resource bin use. This serves as a guard on most resource hoarding code.

ABSORB_RESOURCE_FRACTION 0.025 # Fraction of available environmental resource an organism absorbs with the collect instruction.

MULTI_ABSORB_TYPE 0 # What to do if collect is called on a range of resources.

# 0 = absorb a random resource in the range
# 1 = absorb the first resource in the range
# 2 = absorb the last resource in the range
# 3 = absorb \texttt{ABSORB\_RESOURCE\_FRACTION} / (# of
resources in range) of each resource in the range

\texttt{MAX\_TOTAL\_STORED} = -1
# Maximum total amount of all resources an organism
# can store.

\texttt{USE\_STORED\_FRACTION} = 1.0
# The fraction of stored resource to use.

\texttt{ENV\_FRAC\_TRESHOLD} = 1.0
# The fraction of available environmental resource to
# compare available stored resource to when deciding whether to use stored resource.

\texttt{RETURN\_STORED\_ON\_DEATH} = 1
# Return an organism's stored resources to the world
# when it dies?

### \texttt{ANALYZE\_GROUP} ###

\# Analysis Settings

\texttt{MAX\_CONCURRENCY} = -1
# Maximum number of analyze threads, -1 == use all available.

\texttt{ANALYZE\_OPTION\_1}  
# String variable accessible from analysis scripts

\texttt{ANALYZE\_OPTION\_2}  
# String variable accessible from analysis scripts

### \texttt{ENERGY\_GROUP} ###

\# Energy Settings

\texttt{ENERGY\_ENABLED} = 0
# Enable Energy Model. 0/1 (off/on)

\texttt{ENERGY\_GIVEN\_ON\_INJECT} = 0.0
# Energy given to organism upon
# injection.

\texttt{ENERGY\_GIVEN\_AT\_BIRTH} = 0.0
# Energy given to offspring upon birth.

\texttt{FRAC\_PARENT\_ENERGY\_GIVEN\_TO\_ORG\_AT\_BIRTH} = 0.5
# Fraction of parent's energy given to
# offspring organism.

\texttt{FRAC\_PARENT\_ENERGY\_GIVEN\_TO\_DEME\_AT\_BIRTH} = 0.5
# Fraction of parent's energy given to
# offspring dene.

\texttt{FRAC\_ENERGY\_DECAY\_AT\_ORG\_BIRTH} = 0.0
# Fraction of energy lost due to decay
during organism reproduction.

\texttt{FRAC\_ENERGY\_DECAY\_AT\_DEME\_BIRTH} = 0.0
# Fraction of energy lost due to decay
during dene reproduction.

\texttt{NUM\_INST\_EXEC\_BEFORE\_0\_ENERGY} = 0
# Number of instructions executed
# before energy is exhausted.

\texttt{ENERGY\_CAP} = -1.0
# Maximum amount of energy that can be
# stored in an organism. -1 means the cap is set to Max Double.

\texttt{APPLY\_ENERGY\_METHODOLOGY} = 0
# When should rewarded energy be
# applied to current energy?

# 0 = on divide
# 1 = on completion of task
APPENDIX C. AVIDA CONFIGURATION

FRAC_ENERGY_TRANSFER 0.0
   energy take by new resident
   # 2 = on sleep
   # Fraction of replaced organism's
   # Fraction of replaced organism's energy taken by new resident

LOG_SLEEP_TIMES 0
   off/on
   # Log sleep start and end times. 0/1 {
   # WARNING: may use lots of memory.

FRAC_ENERGY_RELINQUISH 1.0
   relinquish
   # Fraction of organisms energy to relinquish

ENERGY_PASSED_ON_DEME_REPLICATION_METHOD 0 dune
   # Who get energy passed from a parent
   # 0 = Energy divided among organisms injected to offspring dune
   # 1 = Energy divided among cells in offspring dune

INHERIT_EXE_RATE 0
   1=yes
   # Inherit energy rate from parent? 0=no

ATTACK_DECAY_RATE 0.0
   attack
   # Percent of cell's energy decayed by attack

ENERGY_THRESH_LOW .33
   # Threshold percent below which energy level is considered low. Requires ENERGY_CAP.

ENERGY_THRESH_HIGH .75
   # Threshold percent above which energy level is considered high. Requires ENERGY_CAP.

### ENERGY_SHARING_GROUP ###

# Energy Sharing Settings

ENERGY_SHARING_METHOD 0
   # Method for sharing energy. 0=receiver must actively receive/request, 1=energy pushed on receiver

ENERGY_SHARING_PCT 0.0
   # Percent of energy to share

ENERGY_SHARING_INCREMENT 0.01
   # Amount to change percent energy shared

ENERGY_SHARING_LOSS 0.0
   # Percent of shared energy lost in transfer

### SECOND_PASS_GROUP ###

# Tracking metrics known after the running experiment previously

TRACK_CLADES 0
   # Enable tracking of coalescence clades

TRACK_CLADES_IDS coalescence.ids
   # File storing coalescence IDs

### GX_GROUP ###

# Gene Expression CPU Settings

242
MAX_PROGRAMIDS 16  # Maximum number of programids an organism can create.
MAX_PROGRAMID_AGE 2000  # Max number of CPU cycles a programid executes before it is removed.
IMPLICIT_GENE_EXPRESSION 0  # Create executable programids from the genome without explicit allocation and copying?
IMPLICIT_BG_PROMOTER_RATE 0.0  # Relative rate of non-promoter sites creating programids.
IMPLICIT_TURNOVER_RATE 0.0  # Number of programids recycled per CPU cycle. 0 = OFF
IMPLICIT_MAX_PROGRAMID_LENGTH 0  # Creation of an executable programid terminates after this many instructions. 0 = disabled

### PROMOTER_GROUP ###
# Promoters
PROMOTERS_ENABLED 0  # Use the promoter/terminator execution scheme.
# Certain instructions must also be included.
PROMOTER_INST_MAX 0  # Maximum number of instructions to execute before terminating. 0 = off
PROMOTER_PROCESSIVITY 1.0  # Chance of not terminating after each cpu cycle.
PROMOTER_PROCESSIVITY_INST 1.0  # Chance of not terminating after each instruction.
PROMOTER_TO_REGISTER 0  # Place a promoter's base bit code in register BX when starting execution from it?
TERMINATION_RESETS 0  # Does termination reset the thread's state?
NO_ACTIVE_PROMOTER_EFFECT 0  # What happens when there are no active promoters?
# 0 = Start execution at the beginning of the genome.
# 1 = Kill the organism.
# 2 = Stop the organism from executing any further instructions.
PROMOTER_CODE_SIZE 24  # Size of a promoter code in bits. (Maximum value is 32)
PROMOTER_EXE_LENGTH 3  # Length of promoter windows used to determine execution.
PROMOTER_EXE_THRESHOLD 2  # Minimum number of bits that must be set in a promoter window to allow execution.
INST_CODE_LENGTH 3  # Instruction binary code length (number of bits)
INST_CODE_DEFAULT_TYPE 0  # Default value of instruction binary code value.
# 0 = All zeros
# 1 = Based off the instruction number
APPENDIX C. AVIDA CONFIGURATION

CONSTITUTIVE_REGULATION 0  # Sense a new regulation value before each CPU cycle?

### COLORS_GROUP ###
# Output colors for when data files are printed in HTML mode.
# There are two sets of these; the first are for lineages,
# and the second are for mutation tests.
COLOR_DIFF CCCCFF  # Color to flag stat that has changed since parent.
COLOR_SAME FFFFFF  # Color to flag stat that has NOT changed since parent.
COLOR_NEG2 FF0000  # Color to flag stat that is significantly worse than parent.
COLOR_NEG1 FFCCCC  # Color to flag stat that is minorly worse than parent.
COLOR_POS1 CFFCCC  # Color to flag stat that is minorly better than parent.
COLOR_POS2 00FF00  # Color to flag stat that is significantly better than parent.
COLOR_MUT_POS 00FF00  # Color to flag stat that has changed since parent.
COLOR_MUT_NET FFFF00  # Color to flag stat that has changed since parent.
COLOR_MUT_NEG FFF00  # Color to flag stat that has changed since parent.
COLOR_MUT_LETHAL FF0000  # Color to flag stat that has changed since parent.

### BIOMIMETIC_GROUP ###
# Biomimetic Features Settings
BIOMIMETIC_REFRACTORY_PERIOD 0.0  # Number of updates affected by refractory period
BIOMIMETIC_MOVEMENT_STEP 1  # Number of cells to move Avidian on move
    instruction
BIOMIMETIC_MOVEMENT_LOG 0  # Log detailed movement information [WARNING: large
data file]
BIOMIMETIC_MOVEMENT_FACTOR 1.0  # Scale merit bonus due to movement (m<1.0 applies a
cost)
BIOMIMETIC_EVAL_ON_MOVEMENT 0  # Force task evaluation on each movement step
BIOMIMETIC_K 0  # Carrying capacity in number of organisms

### PHEROMONE_GROUP ###
# Pheromone Settings
PHEROMONE_ENABLED 0  # Enable pheromone usage. 0/1 (off/on)
PHEROMONE_AMOUNT 1.0  # Amount of pheromone to add per drop
PHEROMONE_DROP_MODE 0  # Where to drop pheromone
    # 0 = Half amount at src, half at dest
    # 1 = All at source
    # 2 = All at dest
EXPLOIT_EXPLOR_PROB 0.00  # Probability of random exploration
# instead of pheromone trail following
LOG_PHEROMONE 0 # Log pheromone drops. 0/1 (off/on)

PHEROMONE_LOG_START 0 # Update at which to start logging pheromone drops
EXPLOIT_LOG_START 0 # Update at which to start logging exploit moves
EXPLORE_LOG_START 0 # Update at which to start logging explore moves
MOVETARGET_LOG_START 0 # Update at which to start logging movetarget moves

LOG.Inject 0 # Log injection of organisms. 0/1 (off/on)
INJECT_LOG_START 0 # Update at which to start logging injection of organisms

### SYNCHRONIZATION_GROUP ###
# Synchronization settings
SYNC_FITNESS_WINDOW 100 # Number of updates over which to calculate fitness (default = 100).
SYNC_FLASH_LOSSRATE 0.0 # P() to lose a flash send (0.0 = off).
SYNC_TEST_FLASH_ARRIVAL -1 # CPU cycle at which an organism will receive a flash (off = -1, default = -1, analyze mode only.)
Bibliography

URL http://dx.doi.org/10.1038/nrg1771


URL http://dx.doi.org/10.1016/j.ymgme.2009.10.005

URL http://dx.doi.org/10.1155/JBB.2005.172

URL http://dx.doi.org/10.1073/pnas.0505231102

URL http://dx.doi.org/10.1038/nature08480


URL http://www.esp.org/foundations/genetics/classical/holdings/b/wb-methods-08.pdf


URL http://dx.doi.org/10.1172/JCI12557


URL http://dx.doi.org/10.1093/nar/gki180


URL http://dx.doi.org/10.1101/gr.6305707
URL http://dx.doi.org/10.1093/nar/gkp1024

URL http://dx.doi.org/10.1093/bioinformatics/btn051

URL http://dx.doi.org/10.1073/pnas.0603038103

URL http://dx.doi.org/10.1055/s-2008-1061497


URL http://dx.doi.org/10.1101/gad.1945410

Cheng, P., Yang, Y., Liu, Y., Jun 2001. Interlocked feedback loops contribute to the robustness of the neurospora circadian clock. Proc Natl Acad Sci U S A 98 (13),
URL http://dx.doi.org/10.1073/pnas.0705396104

URL http://dx.doi.org/10.1371/journal.pcbi.0030015

URL http://dx.doi.org/10.1038/msb4100128

URL http://dx.doi.org/10.1371/journal.ppat.0020136

URL http://dx.doi.org/10.1016/j.neurobiolaging.2007.11.027

URL http://dx.doi.org/10.1371/journal.pgen.0040035

URL http://dx.doi.org/10.1073/pnas.0334340100

URL http://dx.doi.org/10.1038/35037572


URL http://dx.doi.org/10.1371/journal.pcbi.1000112

URL http://dx.doi.org/10.1002/cbic.200400178


URL http://dx.doi.org/10.1074/jbc.R600018200

URL http://dx.doi.org/10.1073/pnas.0404656101


URL http://dx.doi.org/10.1038/84379

BIBLIOGRAPHY

Heredity 100 (5), 478–483.
URL http://dx.doi.org/10.1038/sj.hdy.6801088


URL http://dx.doi.org/10.1038/37108


URL http://dx.doi.org/10.1186/1471-2148-8-284

URL http://dx.doi.org/10.1111/j.1558-5646.2007.00064.x

URL http://dx.doi.org/10.1002/cfg.173

Feist, A. M., Henry, C. S., Reed, J. L., Krummenacker, M., Joyce, A. R., Karp, P. D., Broadbelt, L. J., Hatzimanikatis, V., Palsson, B. O., 2007. A genome-scale metabolic reconstruction for Escherichia coli k-12 mg1655 that accounts for 1260...
orfs and thermodynamic information. Mol Syst Biol 3, 121.
URL http://dx.doi.org/10.1038/msb4100155

URL http://dx.doi.org/10.1111/j.1610-0387.2009.07137.x

URL http://dx.doi.org/10.1101/gr.3832305


URL http://dx.doi.org/10.1038/35076523

URL http://dx.doi.org/10.1073/pnas.0304532101

coli sm-3-5. J Bacteriol 190 (20), 6779–6794.
URL http://dx.doi.org/10.1128/JB.00661-08

action: enzyme functional annotations in biological databases. Nat Chem Biol 5 (8),
521–525.
URL http://dx.doi.org/10.1038/nchembio0809-521

pathway duplication in the metabolic network of yeast and digital organisms. J R
Soc Interface 6 (41), 1233–1245.
URL http://dx.doi.org/10.1098/rsif.2008.0514

mutation load and inbreeding depression. Genetics 185 (3), 939–959.
URL http://dx.doi.org/10.1534/genetics.110.116368

Glocker, M. O., Guthke, R., Kekow, J., Thiesen, H.-J., Jan 2006. Rheumatoid arthritis,
a complex multifactorial disease: on the way toward individualized medicine. Med
URL http://dx.doi.org/10.1002/med.20045

URL http://www.gnu.org/software/glpk/glpk.html

Learning, 1st Edition. Addison-Wesley Professional.
path=ASIN/0201157675


URL http://dx.doi.org/10.1111/j.1525-142X.2006.05077.x

URL http://dx.doi.org/10.1007/s00239-008-9166-5

URL http://dx.doi.org/10.1556/AMicr.50.2003.4.1


URL http://dx.doi.org/10.1006/tpbi.2000.1508

URL http://dx.doi.org/10.1111/j.1365-2958.2004.04386.x

URL http://dx.doi.org/10.1186/1471-2105-11-379

URL http://dx.doi.org/10.1093/bioinformatics/btl485


URL http://dx.doi.org/10.1038/msb4100174

URL http://dx.doi.org/10.1162/10645460473563577


BIBLIOGRAPHY

URL http://dx.doi.org/10.1093/nar/gki039


URL http://dx.doi.org/10.1093/nar/gkn734

URL http://dx.doi.org/10.1111/j.1525-142X.2009.00347.x

URL http://dx.doi.org/10.1038/nature04228


BIBLIOGRAPHY

URL http://dx.doi.org/10.1371/journal.pcbi.0020059


URL http://dx.doi.org/10.1038/msb.2008.52


URL http://dx.doi.org/10.1093/bioinformatics/bth167

URL http://dx.doi.org/10.1073/pnas.0705455104

URL http://dx.doi.org/10.1007/s10709-008-9310-6


Mrázek, J., Jul 2006. Analysis of distribution indicates diverse functions of simple
sequence repeats in mycoplasma genomes. Mol Biol Evol 23 (7), 1370–1385.
URL http://dx.doi.org/10.1093/molbev/msk023

URL http://dx.doi.org/10.1073/pnas.0702412104

URL http://dx.doi.org/10.1093/bioinformatics/btn077


URL http://www.pnas.org/content/107/8/3930.abstract


URL http://dx.doi.org/10.1006/jmbi.2000.4042

Novère, N. L., Sep 2008. Neurological disease: are systems approaches the way forward?
Pharmacopsychiatry 41 Suppl 1, S28–S31.
URL http://dx.doi.org/10.1055/s-2008-1080913

URL http://dx.doi.org/10.1162/106454604773563612


mutations in *Escherichia coli*. Evolution 59 (11), 2343–2352.


URL http://dx.doi.org/10.1146/annurev.ge.26.120192.000443

URL http://dx.doi.org/10.1186/1471-2148-7-169

URL http://dx.doi.org/10.1371/journal.pcbi.1000206

Penrose, L. S., Sep 1951. Measurement of pleiotropic effects in phenylketonuria. Ann Eugen 16 (2), 134–141, unfortunately, the Annals of Eugenics didn’t become the Annals of Human Genetics until 1954; at least it was Penrose who instigated the name change.

URL http://dx.doi.org/10.1186/gb-2007-8-4-r51

URL http://www.genetics.org

URL http://dx.doi.org/10.1038/nrg2278

URL http://dx.doi.org/10.1098/rstb.2009.0241

URL http://dx.doi.org/10.1002/14651858.CD001304.pub2
URL http://dx.doi.org/10.1088/1478-3975/6/1/016007

URL http://dx.doi.org/10.1126/science.1063143

URL http://dx.doi.org/10.1186/1471-2105-9-499


URL http://dx.doi.org/10.1186/1471-2105-7-118

URL http://dx.doi.org/10.1073/pnas.1003113107


URL http://dx.doi.org/10.1016/j.bbrc.2009.11.058


URL http://dx.doi.org/10.1186/1471-2148-7-205

URL http://dx.doi.org/10.1073/pnas.162041399
BIBLIOGRAPHY


URL http://dx.doi.org/10.1006/jmbi.2001.5226

URL http://doi.acm.org/10.1145/362814.362819

URL http://dx.doi.org/10.1371/journal.pgen.1000344


URL http://dx.doi.org/10.1126/science.1156951

URL http://www.jstor.org/stable/2462954

URL http://dx.doi.org/10.1038/35018085

URL http://dx.doi.org/10.1002/jez.10144

URL http://dx.doi.org/10.1016/j.febslet.2005.01.063

URL http://dx.doi.org/10.1098/rspb.2007.1137

URL http://dx.doi.org/10.1038/nrg2267


URL http://dx.doi.org/10.1371/journal.pcbi.0030198


URL http://dx.doi.org/10.1038/35085569


URL http://dx.doi.org/10.1088/1478-3975/3/3/003


URL http://dx.doi.org/10.1186/1471-2105-7-386