Modelling Biological Form in Evolution

by

Rebecca Cotton-Barratt, MESc., MSc.

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Department of Complexity Science

October 2013

Contents

Acknowledgements

I would like to express my deep thanks and gratitude to my supervisor, Prof. Markus Kirkilionis, for allowing me to pursue my interest in evolutionary theory and introducing me to networks, adaptive dynamics, and C++. I am also grateful to Prof. Hans Metz for his valuable insight and experience.

Thank you to everyone at the Complexity Science Doctoral Training Centre who provided comments and interesting conversation, in particular Chris, Anas, James, and Leigh. I would also like to thank Jen for her amazing all round competence and support. Also, my thanks go to my husband, Owen, for his unfailing love, and to Louise, Conrad, Dette, and Katy for their motivation, distraction, and copious quantities of food. Finally, to my Mum, who made everything possible and to whom I owe so much.

This project was funded by the Engineering and Physical Sciences Research Council through the Centre for Complexity Science Doctoral Training Centre.

Declarations

Parts of Chapter 3 were based on a project report written as part of an MSc. project in Complexity Science – these are sections 3.5.1, 3.5.3 and the beginnings of 3.2.4 and 3.2.5.

Parts of Chapter 4 were published in Proceedings of the European Conference on Complex Systems 2012, September 2013, under the title "Modelling Biological Form", authored with Markus Kirkilionis. However, all work used in this thesis was written by the author.

Abstract

How are processes working at the individual level, the species level and the macro-ecological level connected? This thesis explores the theoretical and structural constraints on biological evolution. It does this by developing an evolutionary program to model biological form. This development was necessary as the existing models of evolution are poorly suited to modelling morphological constraint.

The model of biological form developed in this thesis uses graphs to abstractly represent organisms and the relationships of their internal structure. We show that by increasing the number of degrees of freedom, or by increasing the ruggedness of the fitness landscape, higher levels of diversity are supported - particularly when there is strong directional selection.

We explore whether meta-regulation is bounded in the model by using an analytical framework. We show that there is no analytical steady state, but that one can be induced in the model by selection effects. We find that a mixed strategy between increasing object complexity and increasing hierarchical complexity maximises the average degree of a vertex. This agrees with the evolutionary history of meta-regulation.

We claim that the macro-ecological response to environmental perturbation is determined by both the characteristic time scale of mutation and the time scale of the environmental change. We show that for high amplitude changes the system can adapt provide the mutation time scale is smaller than the environmental change. We also show that low amplitude environmental changes cause rapid turnovers in species' diversity.

Finally, we show that mass extinctions can be the result of species' interactions and background rates of extinction, and do not need large external perturbations to occur. This, combined with the results above, suggests that many of the trends seen over geologically long time periods can be explained as a result of the interacting processes at the individual and species level.

Glossary

Copy numbers A way of keeping track of the number of copies of a species or sections of DNA in a genome.

Emergence The process by which complex systems develop from simple interactions.

Evolutionary pattern formation Pattern formation is the process during the development of an organism where tissues develop more complex functions. Evolutionary pattern formation considers the historical causes or rules behind such patterns.

Evolvability The potential ability of an organism to adapt to change, or the potential changes an organism can adapt to.

Exaptation The change in the function of a trait overly evolutionary time.

Heterotrophic An organism which obtains energy from other organisms.

Ma Millions of years (mega annum).

Macroecology Patterns in ecology considered over geological time scales.

Macroevolution Patterns in evolution considered over geological time scales.

Morphological Relating to the shape and form of organisms.

Morphospace The space of all possible morphological forms.

Polarity An early stage of embryo development where the embryo divides into two hemispheres.

Recombination The process through which genetic diversity is increased by cross-over of chromosomes during meiosis.

Chapter 1

Introduction

Gould [1985] proposes three distinct tiers for evolutionary processes operating over different time scales:

1. Evolutionary events of ecological moment, *e.g.* the results of the struggles of individuals.

2. Evolutionary trends within lineages and clades in geological time, *e.g.* patterns in the evolution of species that occur over millions of years such as punctuated equilibrium.

3. Mass extinctions – large, infrequent events that overturn many of the previously dominant species.

The first tier is the population dynamics tier, which operates according to the survival of the fittest due to competitive interactions between individuals. The modern evolutionary synthesis aimed to create a consistent explanation of evolutionary phenomena, derived solely from these competitive individual interactions that form the first tier. Gould [1985] argues that survival of the fittest implies continued adaptation and optimisation, so we should expect progression in fitness over long time periods. However, we don't see any clear direction of progression [Gould, 1985]. So the behaviour in the second and third tiers is not what we would predict from the first tier, despite the fact that their behaviour emerges from it. This is called the paradox of the first tier. This paradox impacts not only discussions of complexity in organisms [McShea, 1996], but also macroecological trends [Butterfield, 2007].

Gould [1985] proposes two solutions for this paradox. Either

1. First tier does regulate, but the nature of the process is undiscovered. However, there is no expectation of progress.

2. Distinct processes at second and third tier reverse or obscure the first tier accumulations.

This thesis is particularly concerned with linking the first, second and third tiers of evolutionary processes as described in Chapter 2 and defined by Gould [1985]. We approach this paradox both theoretically, in Chapter 3, analytically in Chapters 6 and 8, and by mathematical modelling in Chapters 4, 5, and 7. We propose that first tier interactions control third tier processes, and that the trends seen in the third tier are, in fact, an expected consequence of this control, thus supporting the first of the solutions proposed by Gould [1985].

1.1 Chapter 2

Chapter 2 surveys the existing literature, and explains the background to the aims and methods of this thesis. This covers the theory of evolution and open questions, existing models of evolution which this thesis builds on, and network dynamics which are used in the models developed in Chapters 4, 5, 6, 7.

1.2 Chapter 3

This chapter considers constraints that may be imposed implicitly and explicitly to an evolutionary system. Whilst optimisation is theoretically possible in evolving systems, real world systems rarely, if ever, maximally optimise due to the historically contingent evolutionary constraints applied to them. However, models very rarely include such contingency explicitly. The degree of constraint in such systems, and its effect on the outcomes, is explored following Leimar [2002]. We propose that constraint could be a first tier process that is a driver of evolutionary trends.

In the second section we consider the paradox of evolvability. We propose that if we consider biological evolutionary processes as a non-equilibrium physical system then, after Wissner-Gross and Freer [2013], this seems to negate the so-called paradox of evolvability, *i.e.* that evolvability should not be a persistent feature of organisms as selection is short-sighted in its maximisation of fitness, since non-equilibrium physical systems can delay immediate maximisation to instead maximise entropy production for between now and a future time. This suggests a mechanism for linking first and third tier processes. We explore this claim further in Chapter 6.

This chapter concludes with a discussion on a possible framework for evolution, combining the observations made from the model comparison analysis with the consideration of constraint and evolvability.

1.3 Chapter 4

In this chapter we describe the model that is used in Chapters 5,6, and 7. In the basic model species are represented by their body plans, represented as hierarchical directed graphs. This hierarchical depiction is an elegant way of intuitively displaying the different levels of biological organisation, which could be called the informational content of the body plan. Note that this depiction is essentially the developed information content, it does not include any of the development machinery (nor the genotype information content) for translating the genotype into the phenotype, which could be called the development information.

Each vertex within a species' body plan graph is assigned to a level. Levels represent the vertex's hierarchical position within the species graph. This is a convenient mathematical abstraction, as it has a basis in biology via the levels of organisation typically seen in organisms (for example the proteins which cause cells to specialise which can then form conglomerates such as organs) and has a direct relation to the levels of complexity defined in [McShea, 1996]. Hierarchical structures are evolved through the mutation of individual vertices, which may copy individual vertices or subgraphs, or create new vertices or subgraphs.

1.4 Chapter 5

In this chapter we approach the gap between first and second tiers from the first tier, by asking what the effect of different fitness landscapes is on the overall diversity of the system. Darwin proposed a gradualistic model of evolution, where speciation occurs at a steady rate Darwin [1859]. Punctuated equilibrium was proposed as an alternative model, where speciation occurs in bursts interspersed with periods of stasis Eldredge and Gould [1972]. Although both of these patterns have been shown to occur in the fossil record, it is unclear what the mechanism behind them is. Since mutation occurs at a constant rate, we would predict that a gradualistic trend should be shown in the model, rather than a punctuated equilibrium trend. If we can show the output is dependent on the fitness landscape, this would suggest a causal mechanism for why some lineages show punctuated equilibrium trends, whilst others show gradualistic trends.

We propose that as the changes in fitness occurring now are very small due to the increase in interactions between traits, the replacement of species is not swift and so the apparent biodiversity is large. As the degrees of freedom in a system increase, species are able to to be arbitrarily close to one another, thus slowing the dynamics of replacement.

To translate this prediction into a testable hypothesis we use the model described in Chapter 4. In this context, we predict that models with higher numbers of degrees of freedom should have higher diversity, all other things being equal.

We find that the model exhibits both rapid evolutionary innovation (characteristic of punctuated equilibrium) but also gradualistic change. This is enhanced by the ruggedness of the fitness landscape, and by the presence of a cost to species' structure in the single peak fitness landscape. However, we do not find that the ruggedness or lack of ruggedness induces a particular mode of speciation.

We also find that, in the single peak fitness landscape, increasing the number of degrees of freedom in the species under selection increased also increased the diversity of the system. This agrees with our initial prediction, and we propose that this is because with a higher combination of traits, more species can coexist at arbitrarily close points in the landscape. This slows the rate of competition between species, resulting in an artificial inflation of the diversity of the system.

1.5 Chapter 6

In Chapter 5 we approached the paradox of the three tiers by examining the connection between the first and second tiers (population dynamics and evolutionary trends respectively). In this chapter we consider the second tier and, specifically, the role of meta-regulation in constraining such evolutionary trends. We propose that meta-regulation is one of the more important regulators in evolutionary trends, being responsible for constraint as well as aiding evolvability and altering the tempo of evolutionary processes (as argued in Chapter 3). In Chapter 6 we use the results from Chapter 5 to quantify the variation seen in the average outgoing degree to model meta-regulation rules.

We can model meta-regulation by considering the parameters of the model described in Chapter 4. Increasing object complexity is represented by copying nodes (with probability $1-p$) and increasing hierarchical complexity is represented by creating nodes on new levels (with probability p). β now represents the scope of the regulation – high β represents many genes controlled by few regulatory genes, whilst low β represents direct regulation.

We consider the following two questions:

Firstly, what are the values of p and β which maximise the change in average outgoing degree over time? This provides information regarding the trade-off between increasing object vs. hierarchical complexity.

Secondly, we ask under what conditions is the evolution of meta-regulation promoted? Specifically we are interested in whether there is an analytical steady state towards which a system will tend. In biological systems we have yet to see a limit on the levels of meta-regulation, particularly if the definition is extended to include societal structures (*e.g.* ant colonies) [Vinicius, 2010]. However, we can see that there are jumps in evolution, associated with the development of meta-regulation [Maynard Smith and Szathmáry, 1997; Vinicius, 2010]. This "punctuated equilibrium" has been explained by exploding combinatorial possibilities [Solé et al., 2003], however we propose that such jumps arise naturally under a selection regime which favours moderate connectivity and has four possible mutation outcomes. This would support observations regarding the nature of meta-regulation [Maynard Smith and Szathmáry, 1997] - that is, that there are periods of rapid innovation followed by periods of quiescence.

To answer these questions we use the duplication-divergence framework developed in Vázquez et al. [2003] and Solé and Valverde [2008]. Average outgoing degree is considered, rather than average degree, so as to emphasise the meta-regulation aspects of node degree (*i.e.* the number of nodes controlling other nodes, rather than the number of nodes being controlled).

We find that for sparse or small graphs, the average outgoing degree is maximised when $p = \beta = 1$. However, the growth rate is also maximised when $\beta \neq 1$. Under these conditions a mixed strategy is produced - where small dense sub-graphs are sparsely connected to each other. The model does not have a steady state for average outgoing degree, suggesting that either hierarchical or object complexity (or both) is unbounded.

1.6 Chapter 7

Continuing the theme of uniting the first, second and third tiers of evolution by exploring the gap between first, second and third tiers we investigate in Chapter 7 how environmental variation over different time scales affects diversity. In the previous chapter we looked at intrinsic ways hierarchical complexity could be evolved, whereas in this chapter we look at fluctuations in external resources and their impact on the diversity of organisms evolved within the model described in Chaper 4. We ask the question: what role do external environments play in third tier processes?

We hypothesise that the relationship between external environments and third tier processes is dependent on the relative time scales of, on the one hand, change in the external environment and, on the other, mutation rate. Specifically, we hypothesise that

1. When the characteristic time scale of the mutation rate is smaller than the characteristic time scale of the environmental change, we will observe third tier processes tracking the environmental change.

2. When the characteristic time scale of the mutation rate is larger than the characteristic time scale of the environmental change, we will not observe third tier processes tracking the environmental change.

To answer this we introduce fluctuating resource pools to our model as an environmental change with a clearly defined characteristic time scale. We look at the number of species as a representative third tier variable. We also contrast to a simple version of the model which includes resource pools, but where these are static.

The results from model suggest that, as expected, the impact of the environmental change is determined by the characteristic mutation rate of the system relative to the fluctuations and the amplitude of such fluctuations. Thus, systems can only respond to environmental changes on longer wavelengths ($\tau \geqslant$ mutation rate). Furthermore, we find that the impact of low amplitude wavelengths results in rapid extinctions of multiple species across all time scales. High amplitude changes induce overall system responses in the diversity of the system at meso- and macro-time scales. We attribute this behaviour to low amplitude fluctuations enabling lingering species, which pushes the system to an unsteady state. This unsteady state eventually causes a rapid extinction of multiple species, an overturning of the species present. This apparent cascading, concurrent death of multiple species is interesting as it suggests that the dynamics of both population and resources combine to produce systems which evolve towards a precarious state.

1.7 Chapter 8

In this chapter a null hypothesis for mass extinctions is developed – namely that mass extinctions can be explained by a series of random, unfortunate, but crucially minor, events related to the background rate of extinction. The background rate of extinction is directly driven by competition between species - a first tier process. Mass extinctions are a clear third tier process.

Although the prevailing view is that mass extinctions are driven by large exogenous events (i.e. completely separate from first and second tier processes), we look at data from all known volcanic events and bolide impacts and compare this to the historical extinction record, finding little observable link. If large external environmental perturbations aren't correlated with mass extinctions, an alternative cause must be proposed. In this chapter we develop a model to test whether it is possible for mass extinctions to be caused solely by the background rate of extinction, thus directly linking the first and third tiers.

We show that this model is sufficient to create mass extinctions, using biologically realistic parameters, and thus large exogenous events are not required. The model depends strongly on the number of connections needed to be a highly connected node, and assumes a scale-free network, however we propose that this links processes on the second tier (diversification and species interaction) to those on the third tier (mass extinctions).

Chapter 2

Literature Review

2.1 Introduction

Although the theory of evolution by natural selection is well-established in scientific circles, debate over the fine details has still not been assuaged. As with any unifying theory, and particularly in a field as natural as biology, special cases abound and the complexity of the theory grows as more discoveries are made. Although much of the theoretical evolutionary debate has quietened since the 1970s, evolutionary modelling has made great leaps forward with the advent of computational methods.

The structure of this literature review is to firstly give an overview of the fundamental questions in evolutionary theory, namely the relative roles of adaptation, constraint, and complexity. Naturally, the new synthesis is touched upon, but from a historical basis, considering its impact upon the current generation of evolutionary models. This leads into a discussion of mathematical evolutionary models, beginning with an overview of closed and open problems by Schuster [2010]. One of the criticisms levelled at the new synthesis is the lack of environmental feedback [Durinx and Metz, 2005]. Indeed, although one of the purposes of the new synthesis was to bring the concepts of population dynamics and gene frequency models together with macro-evolutionary trends, many of these emergent phenomena are still unaccounted for.

The mathematical modelling of evolution is in a golden period, and as connections are made between increasingly disparate disciplines, the future looks even brighter. The nature of this thesis is inter-disciplinary, having its roots in biology, mathematics, and palaeontology. As one of its main aims is to examine hierarchical structures in evolution, the next section of the literature review considers network models, particularly those with applications to evolution.

2.2 Evolutionary Theory

2.2.1 Adaptation

The idea of a unified evolutionary theory is flawed almost from its conception, due to a lack of data. This is not the lack of geological data, discussed in the Origin of Species as the "incompleteness of the geological record"; it is a lack of comparative data at the grandest scale. As far as we know life has only evolved once on Earth. As such, the many explanations put forward for an organism's evolutionary trajectory are often not falsifiable [Popper, 1968]. One of the strongest advocates for this perspective-inspired way of thinking was Stephen Jay Gould, and nowhere is his argument made more strongly than in "The Spandrels of San Marco and the Panglossian Paradigm" [Gould and Lewontin, 1979].

The arguments in Gould and Lewontin [1979] centre on the adaptationist paradigm, which at its heart is the optimisation principle of natural selection. This paradigm, and its effect on evolutionary modelling, is discussed in more depth in the following chapter. Gould and Lewontin [1979] warns against the atomisation of traits into perfectly optimised functional structures, an important point which has recently influenced discussion on constraint in evolution (*e.g.* Wagner and Stadler [2003]; Hansen [2003]). Such concerns have also permeated into the genotype-centred view of evolution, where covariance matrices are used to estimate the degree of connectedness (see for example Metz [2011]). Such a solution falls into the second warning of Gould and Lewontin [1979], that such trade-offs are introduced to explain the maladaptation or lack of apparent optimisation. Thus organisms are "interpreted as best compromises among competing demands" [Gould and Lewontin, 1979]. This "solution" allows the paradigm of adaptation to exist in the presence and absence of evidence, which invalidates the falsifiability of the paradigm.

As an alternative view to the optimisation (or immediate adaptation) of evolutionary change, Gould and Lewontin [1979] proposes five possible processes. Some of these fall into already proposed processes of evolution, such as the neutral theory of evolution [Jukes and Kimura, 1984], which has no adaptation or selection. A subset of this is where there is no selection or adaptation in the trait being studied, which can be seen in the growth laws of organisms [Thompson et al., 1942]. There is the possibility of decoupling of adaptation and selection [Gould and Lewontin, 1979], an example of which is when the range of possible forms (the morphospace) is highly constrained by external physical laws (such as in unicellular organisms). It is also possible for there to be adaptation and selection, but the differences in adaptations may not be due to a selective reason. This comes almost naturally from the idea of the many-to-one genotype-phenotype mapping (in-depth discussions of this can be found in Stadler et al. [2001]; Schuster [2010]). Finally, adaptation may be a secondary utilization of parts historically present [Gould and Lewontin, 1979]. This cannibalization is seen at all scales in biology, from protein pathways [Solé et al., 2003] to whole organs [Gould and Vrba, 1982].

Evolutionary inertia, *i.e.* the lack of fundamental change in organisms, can be explained in terms of phyletic inertia (past adaptations slow down or impede adaptation) [Gould and Lewontin, 1979]. Developmental inertia is embodied in the idea of Baupläne – where early set developmental processes and pathways constrain future change [Gould and Lewontin, 1979]. Although this identifies the possibility of developmental constraint, the paper offers little in the way of quantification of such concepts. Indeed, without such quantification the same criticisms Gould levels at the adaptationist paradigm could be levelled at the evolutionary processes proposed in Gould and Lewontin [1979].

We consider adaptation further in Chapter 7, in which we claim that the relative time scales of environmental change (which organisms must adapt to) are critical drivers of third tier processes.

2.2.2 Constraint

A mathematical attempt to quantify the effect of constraint was carried out by Leimar [2002]. However, rather than consider explicitly the structural constraint discussed above, Leimar [2002] considered a Darwinian demon, capable of deciding which mutations appear but not the subsequent effects of natural selection. This results in a system analogous to morphospace, where what is possible matters more than what is probable. The study limits itself to small effect mutations, as large effect mutations would simply increase the effect of the Darwinian demon [Leimar, 2002]. Correspondingly, the study cannot explore the relative importance of size of effect in mutational jumps.

In the $1d$ case, mutational effects can only influence the rate of mutation change, rather than the direction [Leimar, 2002]. In the multidimensional case, any small increment Δx which points uphill is favoured by selection, but this uphill requirement decreases in higher dimensions [Leimar, 2002]. The trait combination which has the steepest slope will be selected for, however this may not result in all traits increasing in fitness due to pleiotropy (where a single mutation affects several traits of an organism).

The global properties of the fitness landscape constrain the evolution, rather than the local structure [Leimar, 2002]. This is in agreement with the case for epistatic interactions causing ruggedness in fitness landscapes made in Solé et al. [2003]. Leimar [2002] also discusses the fixedness of fitness landscapes. Thus if there are no inescapable traps, as might be expected in a shifting landscape, then adaptive walks can range widely over the space [Leimar, 2002]. Of course, this is naturally a limitation of the choice to only consider small effect mutations, as large effect mutations are less susceptible to such traps.

In evolutionary game theory, uninvadibility has played a greater role than convergence stability [Leimar, 2002]. However, there exist perturbations which encourage populations to move away from an evolutionary stable strategy (ESS) [Leimar, 2002]. Therefore, convergent stability must play an important role in determining the long term outcomes of the evolutionary process [Leimar, 2002].

Leimar [2002] concludes that a Darwinian demon could potentially have an unlimited effect on evolution, by utilising the potential invadability of one of a pair of mutations which have no net effect on overall fitness. Thus the important effect of mutations is in the degree of correlation or connectedness that develops between traits [Leimar, 2002].

How does this fit with the framework discussed in Gould and Lewontin [1979]? The importance Leimar [2002] places on convergent stability can be considered as an attempt to formalise architectural constraint (albeit for a particular subset of examples). The conclusion that a Darwinian demon could have a potentially unlimited effect on the outcome of evolutionary change reinforces the suggestion in Gould and Lewontin [1979] for a separation of adaptation and selection. However, the fundamental question of how to quantify architectural constraint remains open.

In Chapter 3 we claim that constraint is a driver of evolutionary trends, and in Chapter 6 explore this claim further.

2.2.3 Macroevolution

Another theoretical consideration is the discreteness of the levels of evolution. As discussed in the first section Gould [1985] proposes three tiers for evolutionary outcomes:

1. Evolutionary events of ecological moment.

2. Evolutionary trends within lineages and clades in geological time.

3. Mass extinctions.

The first tier can be thought of as the population dynamics tier, and indeed the tier which the new evolutionary synthesis aimed to merge with the second tier.

As such, the modern synthesis could be argued to presuppose that all evolutionary theory derives solely from the first tier [Gould, 1985].

Although Darwin stated that “from so simple a beginning endless forms most beautiful and most wonderful have been, and are being, evolved”, it is unclear that the forms are endless. One of the most surprising observations in biology is that, once the superficial differences in species are accounted for, the base disparity in animals is not only restrictive but unchanging. The 30 bodyplans [Raff, 1996] that evolved amongst others in the Cambrian Explosion are still very much present in today’s organisms, and it is unclear what progress at that level is, and whether any has taken place. Given that natural selection supposes that competition is the main driver in evolutionary progress, such that more adapted forms persist, the proposed paradox of the first tier is the absence of a clear direction of progression [Gould, 1985]. Such a paradox impacts not only discussions of complexity in organisms [McShea, 1996], but also macroecological trends [Butterfield, 2007].

Gould [1985] proposes two solutions for this paradox. Either

1. First tier does regulate, but the nature of the process is undiscovered. However, there is no expectation of progress.

2. Distinct processes at second and third tier reverse or obscure the first tier accumulations.

This latter suggestion is of vast importance to evolutionary modellers, because if true, then no current evolutionary model can predict the outcome of long term evolution based simply on competition and survival at the individual level. Gould [1985] proposed that the development of a hierarchical Darwinian view of evolution should be “a major agenda for evolutionary theory before the millennium”. However, such an agenda has not been acted upon, nor has there been any development of such a theory.

In Erwin [2000], the same questions are asked: do macroevolutionary processes simply reflect the accumulation of microevolutionary processes, or are macroevolutionary processes emergent phenomena? How important is repatterning and redeployment in developmental processes, and how can the macroevolutionary potential of developmental innovations be measured? Erwin [2000] emphasises that there is a non-random origination of evolutionary novelties over time, because innovations require ecological opportunity, developmental possibility and the right environmental setting. Innovations in this context are defined as major changes in the organism, which may involve early developmental processes.

Discontinuities impart a hierarchical structure to evolution and may even neutralise microevolution [Gould, 1985; Erwin, 2000]. The relevant question is not whether macroevolution distinct from microevolution, but the relative frequency and impact of processes at the various levels of the hierarchy [Erwin, 2000]. As with the effect of architectural constraint, there is a lack of quantification of these processes. One reason for this lack of quantification is the unanimous uptake by evolutionary modellers of the new synthesis.

In Chapter 3 and in Chapter 8 we consider macroevolutionary trends and claim that third tier mass extinctions are a result of competitive interactions at the population level - that is, a direct outcome of first tier processes.

2.2.4 Complexity

Returning again to the paradox of the first tier Gould [1985], it is possible to find the idea of progress not only permeating many discussions of evolutionary trends, but also evolutionary complexity [McShea, 1996]. Arguments put forward for increasing complexity over time include the hypothesis that as diversity increases and niches become more specialised, and by extension complex, then the organisms themselves become more complex [McShea, 1996; Waddington, 1969]. This argument shows no awareness of the processes of the third tier (mass extinctions), in which it should be expected that more complex organisms (if more specialised) have a lower chance of survival during extreme environmental perturbation. However, if complexity is increasing across all species, families and genera, then for each successive mass extinction a higher proportion of species should be made extinct, as complexity even in the simplest organisms increases. This trend is not seen in the fossil record, and although that may be due to the severity of the environmental perturbations, it seems impossible to test such a hypothesis.

Although intuitively progress is a more nebulous concept than complexity, it turns out that complexity in organisms is also ill-defined. McShea [1996] identifies four distinct structural components that make up organismal complexity:

1. Object complexity which is the number of different physical parts of a system.

2. Process complexity which is the number of different interactions between them.

3. Hierarchical structure which is the number of levels of nestedness (or the number of levels of organisation).

4. Non-hierarchical complexity which is the number of parts of interactions at a given scale.

Any empirical evidence for supporting these trends requires these four components to be measurable in some way [McShea, 1996]. However, it is unclear even with such measures whether overall organismal complexity (composed of the above components) has been increasing over time [McShea, 1996]. At best, there are two possible processes that describe the pattern seen in the fossil record – diffusive or passive, where there is simply a lower bound on the complexity of the simplest organism, or driven where complexity increases over time [McShea, 1996].

Defining complexity in this way is inclusive of randomness, and avoids the pitfalls of functional definitions referenced in the above discussion. However, in this definition complexity is not entropy [McShea, 1996]. This separation of complexity from entropy is particularly unsatisfying because Kolmogorov complexity describes genotype complexity so well. Lloyd and Pagels [1988] argued that complexity should be related to the generating process. However, such a measure of genotype complexity ignores the complexity of the developmental process, and defining the one in terms of the other potentially hides any further insight [McShea, 1996].

In McShea [2000] a definition of functional complexity is introduced, however it suffers from the problems of adaptation versus exaptation, that is the change in the function of a trait over time. It does, however, define a "part" of an organism as internally highly connected, but isolated externally from surrounding parts. This, of course, is isomorphic to the notion of community used in network science. This definition, along with the structural definition above, suggests networks as a natural medium in which to explore questions of constraint, complexity, and contingency.

One of the most interesting questions raised by the McShea [1996] paper is whether there are limits to complexity, and if so what these limits are. In McShea [2005], a process for increasing the internal variance of an organism is proposed, whereby the accumulation of mutations results in further differentiation of an organism. This process of differentiation is analogous to the duplication-divergence model of protein networks discussed later and in more depth in chapter 6. Although in McShea [2005], this increase should only be seen in non-hierarchical object complexity, it would be very interesting to see it extended to hierarchical object complexity.

All the notions of increasing complexity treat evolution as an essentially non-reversible process. This is interesting, in particular because we know it to be false, at least at the adult phenotype level [Galis et al., 2001]. It seems that mutations within the developmental process can lead to the effective removal of otherwise non-functional "parts". If the developmental process can have such an effect, there is surely a strong argument for it to be included in models of evolution, either explicitly or by proxy.

It is possible that a deeper understanding of the genotype-phenotype map can shed light on this division, but it is equally likely that a mapping from Kolmogorov complexity measures to structural measures might shed some light on both the developmental process and the genotype-phenotype map.

In Chapter 6 we claim that meta-regulation is strongly selected for, and explore the trade-off between hierarchical and object complexity.

2.2.5 New synthesis

The role of development has been understated in both the new synthesis and evolutionary models. The lack of a coherent understanding of the process and whether there are top-down or bottom-up effects on evolution has stymied the attempts of modellers to create integrated models of evolution. Part of this lack of understanding has come from development's neglect in the hands of the new synthesis [Amundson, 2005b]. The confusion surrounding both development and structuralism is neatly summarised by Amundson [2005b]: "Structuralists seem (to the adaptationists) to be saying that past history is causally affecting present-day populations without explaining how this can be accomplished." Of course, proposing such outcomes or constraints without a coherent process is speculative in the extreme, and this is where a need for explorative modelling is great.

Amundson [2005b] claims that typology and essentialism, that is characters conforming to types and species defined by sets of attributes, (which underlie the idea structural constraint) are simply evolutionary processes which cannot be reduced to populational processes. In this he is following in the footsteps of Gould and Lewontin [1979] and Erwin [2000]. Any claims over the realism of typology makes a statement over the permanence of characters, a test of which could be the rate of decay of characters associated with the bodyplan compared with characters not associated [Amundson, 2005b]. However, the notion of characters sits uneasily aside this integrated notion of typology.

Homologous characters are shared between two taxa as they were present in their last common ancestor. But the identification of homologous characters not only runs afoul of the problem of functional equivalence, but also the identification of "parts" or characters. A further problem is that homology, though widely used in phylogenetics, is not reducible to any particular developmental process Amundson [2005b]. Wagner and Stadler [2003] proposed that a developmental account of homology should include their conservation through time, their individuality, and their origin. Such an account requires an understanding of the networks and hierarchical relations of induction in the developing embryo.

The questions of why organisms have distinct parts and why it is so easy to spot correspondences amongst them are as yet unanswered. Amundson [2005b] argues that these may be answered by some combination of the bodyplan, which is constantly maintained through ontogeny, and modularity, where parts are semi-independent. It is perhaps also related to the human capacity to generalise. Note, however, that a structurally similar phenotypic outcome can be maintained (and identified as homologous) despite the developmental processes responsible being independently modified. This is seemingly analogous to the many-to-one mapping of genotype to phenotype.

The thirty basic Baupläne [Raff, 1996], are causally involved in the evolutionary process as they require a specific causal explanation [Amundson, 2005b]. Their potential impact on evolutionary processes are two-fold – they are modified and maintained, but can influence phenotypic variations available. This constraining aspect of their nature is particularly important from a modelling perspective, as it changes which characters we expect to see as evolution progresses. Understanding this constraint would enable us to make testable predictions of the evolutionary process.

2.3 Models of Evolution

2.3.1 Neutral networks

Now that the theoretical backdrop has been established, the next step is to consider what steps in modelling have been undertaken to solve these problems. Modelling is a very powerful tool which can be used to test hypotheses and isolate the key underlying processes in complex systems. Schuster [2010] gives an overview of the recent advances in modelling, in particular insights into the genotype-phenotype map. One of the key discussions in the paper concerns error threshold rates as applied to mutation in genetic code. The error threshold limits the chain length for faithful reproduction [Schuster, 2010]. There is potentially an analogy here between the trade-off between information content and faithful reproduction with the trade-off between evolvability and robustness. In particular, such an analogy could be easily modelled.

Schuster [2010] then goes on to discuss neutral networks. Neutral networks potentially exist in any many-to-one mapping, as they describe the ability to move through one space without incurring penalties in the other space. Thus, the redundancy offered by a many-to-one mapping allows this, as does (it should be noted) a completely flat fitness landscape, since ultimately the genotype-phenotype map is used in these discussions as a proxy for the fitnesses of genotypes. In the RNA-protein structure mapping, the amount of redundancy determines whether the neutral network formed by the mapping are well-connected or split into components [Schuster, 2010]. The critical connectivity that decides whether the network is well-connected is depends only on the number of letters in the genetic alphabet [Schuster, 2010].

If a neutral network exists, evolution can be modelled as a diffusive process [Kimura, 1968]. In this case the phenotype is constant [Schuster, 2010]. Alternatively, several phenotypes can appear with the same replication rate, and a random walk occurs between these structures [Schuster, 2010]. However, as yet a comprehensive stochastic modelling approach to evolution has not yet been developed [Schuster, 2010]. Schuster [2010] also notes that the predictive power of the genotype-phenotype map is very poor. Unfortunately, this analysis also ignores the potential structural constraints that occur if scaled to the genotype-phenotype map. In particular, the RNA structure is well-understood in terms of its kinetics, thus the phenotype in this case is limited only by physical constraints. The system is memoryless, but it is unclear (as discussed previously) whether evolution at the level of organisms is so forgetful.

We explore this mapping in Chapter 3 and in Chapter 5, where we consider the relationship between diversity and degrees of freedom in the system.

2.3.2 Environment

This type of modelling ignores the effect of environment except in the population dynamics phrasing, where the fitness proxy is simply reproduction rates. In contrast, the Tangled Nature proposed in Christensen et al. [2002] derives its dynamics solely from interactions with the environment. In this model, the interaction strength between individuals (represented by their genetic sequence) determines the possibility of the organism to succeed [Christensen et al., 2002]. If speciation is kept constant, independent of the strength of coupling, then the population is diffusive across genome space [Christensen et al., 2002]. This is expected not only from the behaviour of aggregation models, but it is also an example of the phenomenon noted above, where evolution is diffusive in the genotype if there exists a many-to-one mapping, or in the case where the fitness landscape is flat.

A physical environment, as opposed to the environment created by other individuals, is represented by a death rate and associated carrying capacity [Christensen et al., 2002]. The results of this model are surprising – in the asexual reproduction case, the system passes through a bottleneck even if diversity is initially high [Christensen et al., 2002]. There is the observance of quasi-ESSes, and the time to reach stationary states increases exponentially with increasing genome length [Christensen et al., 2002]. For sexual reproduction, as reproduction probability is determined by Hamming distance, then similarity is favoured [Christensen et al., 2002]. However, if the physical environment is exceedingly harsh, low diversity is favoured over high diversity [Christensen et al., 2002]. These observations have important repercussions, not only for evolutionary modelling, but also the integration of population scale dynamics with macro-ecological dynamics – a possible uniting of the first and third tiers.

The main concept that is missing from the model described in Christensen et al. [2002] is a separation of time scales. In Metz [2011], this separation of time scales is emphasised in the framework of mathematical modelling. Fitness at the micro-evolution level comprises a quantitative measure of an individual's competitive success, and thus depends on the other individuals and the environment [Metz, 2011]. Thus, the population state space is the positive measures (since positive fitness is success) over the individuals' state space, and the state space of a community is the product of the state spaces of the comprising species, plus the state spaces of the dynamics of any external resources [Metz, 2011].

In the framing of adaptive dynamics, evolutionarily steady strategies (ESS) are formed when all mutants of the residents have negative fitness [Metz et al., 1996]. It can be proved that an invading type replaces its progenitor if the latter is not too close to an ESS or bifurcation point of the community dynamics, and the mutational step was not too large [Metz et al., 1996]. However, this has not yet been extended to more general structured populations [Metz, 2011]. Indeed, several open problems are proposed in Metz [2011]:

1. How do the outcomes of a specific eco-evolutionary model depend on possible forms of the mutational covariance matrix?

2. How does the mutational covariance matrix vary with the traits?

Both of these concern mutational covariance matrices, which are the mathematical modellers proxy for structural constraint (and indeed, other forms of constraint). Thus the first problem asks for a mathematical modelling approach akin to that seen in Leimar [2002], and whether there can be measurable effects of constraint. The second asks whether it is possible to predict the covariance behaviour of a particular trait, given its mutational covariance matrix. We do not address such questions in this work.

In Chapter 7 we look at the effect of varying time scales of environmental change and claim that the characteristic time scale of the change determines the evolutionary response of the system. In Chapter 5 we find that, in a single peak fitness landscape, diversity is increased by increasing the degrees of freedom in the system. We hypothesise that this is because the increased degrees of freedom allow organisms to cluster arbitrarily close to the fitness peak.

2.3.3 Evolutionary limits

If models are a way of testing possibilities, then mathematics provides the logical framework within which the model is formed. In Chaitin [2012], evolution is taken to its most abstract form, as the fitness of algorithmic programs is considered. In this model, mutations are treated as point mutations in the program, a program A is k bits away from B if the probability of moving from $A \rightarrow B$ in a single step is $\frac{1}{2^k}$ [Chaitin, 2012]. Note that this is in bit framework, so each position in the program is either a 1 or a 0. It is also possible to define the program-size complexity, namely the relative information content of B given A, which is also the mutation distance is $-log_2$(probability of going from $A \rightarrow B$ in a single mutation).

Chaitin [2012] then uses the Busy Beaver function, defined as the largest integer that can be named in N bits, to create a fitness definition for the algorithms. Note that in doing this Chaitin [2012] must avoid Turing's Halting Problem by utilising an oracle which avoids algorithms which do not halt and mutations which do not work. This is unfortunate, for it abstracts away all the potentially important detail, leaving evolution as a simple optimisation process.

Nevertheless, even within this optimisation framework, the results of this abstract model are intriguing. There are three possible ways to optimise. The intelligent design principle, with an omnipotent, omniscient process, who evolves the best of all possible worlds, which reaches $BB(N)$ in time N [Chaitin, 2012]. Secondly, there is the brainless exhaustive search, which tries every possibility until reaching a dead-end, at which point it reverts to the last previous choice; which reaches $BB(N)$ in time 2^N [Chaitin, 2012]. Finally, cumulative random evolution, which accumulates the best mutations over time but mutation proceeds in a random fashion, reaches $BB(N)$ in time N^2 to N^3 [Chaitin, 2012]. What is surprising is that cumulative random evolution does substantially better than the brainless exhaustive search [Chaitin, 2012]. What is also interesting is that it is restrictions on the course of random walks which improve the outcome. Although constraint in evolution is often used in a negative context, it is possible that structural constraint in fact enables evolution far more than it restricts.

2.4 Network Models

One of the most interesting dynamics in studies of genetics are the regulatory effects that some genes have on other genes. Such regulating genes produce spatio-temporal differences in gene expression, particularly during developmental stages [Erwin and Davidson, 2009]. A natural way to represent these genetic interactions is via a directed graph, or network. Bodyplans, as discussed previously, are fundamental structures which in some way represent the outcome of these interacting genes.

The structure of such networks is highly modular, with independent subunits representing different phenotypic effects [Erwin and Davidson, 2009]. Such a hierarchical structuring of a network is vitally important in analysing what the effects of mutations are at different points in the genome [Erwin and Davidson, 2009]. What is also important is that the probability of successful mutations varies across the hierarchy, with phyletically "deep" genes (those high in the hierarchy) having few, if any, non-deleterious mutations [Erwin and Davidson, 2009]. This, in fact, provides the evidence for the existence of deep-seated structures which are transmitted through time. Such structures, referred to as kernels, show a structural constraint which is invoked by network topology [Erwin and Davidson, 2009].

This hierarchical structure also describes the divergence of orders and classes, when mutations occur in intermediate positions of the hierarchy, whilst species-specific mutations occur on the periphery [Erwin and Davidson, 2009]. However, what this also implies is the relative size effects of mutations in different locations – in particular, that some mutations may potentially have large phenotypic effects [Erwin and Davidson, 2009]. Over evolutionary time, the range of possible mutations has contracted by changing the probability of evolution in different parts of the network [Erwin, 2012]. This "meta-regulation" of mutation has no obvious responsible process, though its evolution is presumably selected for. However, if such a process is relatively irreversible, it is unclear whether to describe it as selected for, rather than just an inescapable evolutionary trap.

In Solé et al. [2003], genetic network analysis is combined with adaptive walks on rugged fitness landscapes to explore the apparently rapid origins of bodyplans. The fundamental constraints imposed by early developmental dynamics meant swift exploration of the space of possible bodyplans was followed by an establishment of lower taxonomic groups as the possibility of finding fitter mutants decreased [Kauffman and Levin, 1987]. This agrees with the experimentally based arguments presented in Erwin and Davidson [2009], and also proposes a mechanism by which such a state could be achieved (the ruggedness of the fitness landscape). This ruggedness is produced by traits interacting. A natural question to ask, therefore, is whether there is an interaction threshold beyond which increasing the number interactions results in disadvantageous ruggedness. An obvious way to test this would be via modelling the interactions as a network, and identifying peak connectivity. It would be interesting to see whether the results thus obtained match the critical connectivity threshold for neutral genotype networks described in Schuster [2010].

In Chapter 6 we use a framework developed by Vázquez et al. [2003] and Solé and Valverde [2008] to claim that meta-regulation is strongly selected for.

2.5 Conclusions

In this chapter a summary was given of the background and current state of affairs in the modelling of biological form. Several questions were posed, which this thesis hopes to answer, at least in part. The first section on evolutionary theory summarised the historical background behind structural constraint [Gould and Lewontin, 1979], the possibility for emergent evolutionary phenomena which could not be extrapolated from micro-evolutionary processes [Gould, 1985], and the problem with defining organismal complexity [McShea, 1996, 2000, 2005]. Additionally the role of developmental processes and the concept of a bodyplan was considered.

Many of the questions posed in this theoretical section (the role of optimisation, how to quantify structural constraint, and how do the evolutionary processes at one scale influence those on another) could be illuminated by *in silico* modelling of evolution. The second section in this chapter considered current evolutionary models, which rarely seem to address the concerns of the theorists. The predictive power of the genotype-phenotype map, and the lack of a comprehensive stochastic modelling approach are identified as concerns [Schuster, 2010], as well as an incomplete understanding of mutational covariance matrices [Metz, 2011]. Also highlighted was the importance of interactions between species in defining evolutionary phenomena, and the effect of a physical environment [Christensen et al., 2002]. However, the questions still remain as to whether constraint can be measured, and if so, in what form.

Finally, the possibility of network modelling was considered in an evolutionary context. Network models of genetic regulatory networks were discussed, proposing a non-uniform mutation probability distribution, as well as the evolution of meta-regulation of mutation probabilities [Erwin and Davidson, 2009]. The effect of genetic regulatory networks was considered in the context of evolving bodyplans and the rapid exploration of morphospace during the Cambrian explosion [Solé et al., 2003]. However, this asked more questions than it answered, in particular regarding the possibility of a critical connectivity in networks related to the ruggedness of the fitness landscape.

Chapter 3

Evolution as Theory

3.1 Introduction

The benefit of reducing real world systems to logical formalism is that the key processes can be identified and explored. However, the relative lack of comparative analysis between different models of evolution results in models being chosen seemingly at random, without comprehensive consideration of the effect model choice has on output. Too often, established models are viewed as black boxes and their efficacy or applicability to the problem unknown. This has been particularly problematic in the field of phylogenetics, where models of evolutionary change implicitly alter the reconstructions of ancestral states [Pagel and Harvey, 2008]. This chapter presents a theoretical framework for considering evolutionary models, as well as individual case studies of theoretical considerations.

In particular, this chapter considers constraints that may be imposed implicitly and explicitly to such a system. It may be seen that whilst optimisation is theoretically possible in evolving systems, real world systems rarely, if ever, optimise due to the historically contingent evolutionary constraints applied to them. However, models very rarely include such contingency explicitly. The degree of constraint in such systems, and its effect on the outcomes, is explored following Leimar [2002]. We propose that constraint could be a first tier process that is a driver of evolutionary trends.

In the second section we consider the paradox of evolvability. We propose that if we consider biological evolutionary processes as a non-equilibrium physical system then, after Wissner-Gross and Freer [2013], this seems to negate the so-called paradox of evolvability, *i.e.* that evolvability should not be a persistent feature of organisms as selection is short-sighted in its maximisation of fitness, since non-equilibrium physical systems can delay immediate maximisation to instead maximise entropy production for between now and a future time. This suggests a mechanism for linking first and third tier processes.

This chapter concludes with a discussion on a possible framework for evolution, combining the observations made from the model comparison analysis with the consideration of constraint and evolvability.

3.2 Constraint

3.2.1 Introduction

As noted in Chapter 2, there are still many open questions regarding biological constraint – in particular, how to quantify architectural constraint and how to quantify constraint in general. Many evolutionary models consider constraint as any possible effect that could defer the optimisation process, through covariance matrices. In this section, this definition is considered with reference to previous literature and then a theoretical framework for quantifying fitness is developed.

3.2.2 Definitions of constraint

Björklund [1996] uses the definition given by Stearns [1982], that a constraint is any factor which slows the population from attaining immediate access to the nearest adaptive peak on a fitness landscape. We define a fitness landscape as the resultant space after taking the reproductive success of a species in a given environment. Species may be defined either by phenotype or genotype. However, it is clear that this definition of constraint fails to take into account that constraints need not have a negative effect on evolutionary progress – that in some cases constraints accelerate progress towards an adaptive peak (see for example Gould [1989]; Gerhart and Kirschner [2007].

This narrowness of view is frequently found in papers pertaining to evolutionary constraints. Jukes and Kimura [1984] assume that the rate of fixation is maximised when there are no adaptive constraints, effectively meaning there exists no selection. This assumes that there is no correlation between mutation points and other loci, thus assuming that the genome is a string of independent characters, or that correlation exists but all such correlated mutations are neutral. Galis et al. [2001] also use constraint as a term for the limiting of possible evolutionary trait trajectories, which implicitly accepts that constraint cannot increase the number of possibilities (since unconstrained evolution assumes all evolutionary trajectories are viable).

Such considerations of constraint have also been undertaken in morphology – specifically in the investigation of morphospaces (*e.g.* Niklas [1999]; McGhee [2007]). The relative over- or under-occupation of morphospace could potentially shed light on constraining effects. However, as argued in Pie and Weitz [2005] such observations require a null model in order to have any meaning. Is the null model in constraint that evolution would proceed optimally, after the modern synthesis which determines that there must be overall progress? Certainly, this appears to be the case from the definitions so far considered.

Gould [1989] defines constraint as the sources of changes that do not arise through the action of stated causes within a favoured theory. Leaving aside the somewhat dubious nature of this definition, Gould [1989] then goes on to propose that formal constraint (arising from the formal rules of structure, defined by morphologists such as D'Arcy Wentworth-Thompson and Cuvier) and phylogenetic constraint (arising from historical contingencies of phylogeny) interact to form functional active adaptation.

In Galis et al. [2001], experimental evidence is given for the impact of constraint on possible mutations – namely that the lack of variation seen in digit numbers in amniotes compared with the large variation seen in amphibians with aquatic larvae can be traced back to the modularity of the developing embryo. Limb development in amphibians with aquatic larvae is decoupled from much of the development, and thus mutations have low pleiotropic effects compared with amniotes [Galis et al., 2001]. In particular, any reductions in digit number occur as additional steps in the development process, rather than as deletion processes earlier in development [Galis et al., 2001]. Such a definition clearly affects evolvability, arguably in a negative way. The link between constraint and evolvability is explored in Section 3.3.

3.2.3 Fitness constraints

The forms that constraints may take is determined by the form of evolution that takes place. Consider constraints in a genetic algorithm, where evolution is an act of optimisation on a fitness landscape. Implicit in this model is the assumption that evolution acts to locally maximise fitness; thus we can define constraint to be anything that causes the evolutionary trajectory of the system to deviate from this maximum. In an initial thought experiment, consider a species that has a single real valued trait that it needs to maximise in order to be fitter. The species therefore exists in a single peak fitness landscape. Any mutation that the species has will either increase or decrease the value of this trait. Therefore as $t \to \infty$ the population will undertake to climb this peak. The rate at which it does so may be constrained, which may occur due to the underlying probability distribution of the different mutations. This, of course, is dependent on the genotype-phenotype mapping. Such a framing implicitly assumes that the atomisation of traits is not only permitted, but complete, that is that the single trait that is being considered is the only trait affected by evolution.

Let us extend this scenario to a species with two real valued traits. Now, the possibility exists for there to be multiple peaks (depending on what functions the fitness landscape is defined on). However, in the case where there exists only a single peak in the fitness landscape (such that only one combination of the two traits achieves maximum fitness), there exists only a potential constraint on the rate of progress. In the case where there exist multiple peaks, constraint (as defined by Galis et al. [2001] as the limiting of possible evolutionary trait trajectories) naturally arises since the trajectory is forced to choose one particular path.[1] Assuming a fixed fitness landscape, by climbing mount A the species can never reach mount B. Therefore, in the case of the evolutionary optimisation model, where the fitness landscape is static, constraint is directly related to the ruggedness of the fitness landscape. We consider this explicitly in Chapter 5, following the model defined in Chapter 4.

For completeness we now consider the two species' case. This is identical to the one species' example if we assume that the fitness landscape is independent of the environment formed by the two species. This is an over-simplification, and so instead we look at the case where the environment (and by extension the fitness landscape) depends on both species. The mathematical framework to do this was established by Metz et al. [1992], and has proven useful in analysing co-evolutionary scenarios (*e.g.* Meszéna et al. [1997]; Geritz et al. [2004]). Constraint in such a system exists by having multiple ESSes, but there should also exist a covariance matrix which defines how one trait changes with respect to another trait [Metz, 2011]. At present, it is not known how to reliably construct such a matrix [Metz, 2011]. Is there an alternative solution?

Consider a demon similar to Leimar's Darwinian Demon [Leimar, 2002], which can ignore any and all constraints in the evolutionary system. As a result selection will result in the maximal fitness possible for organisms. In such a regime, one has to define mutation carefully. If we imagine the species existing as a point in genotype space, then mutation rate effectively increases the radius of possible combinations that the species can reach in one time step. Of course, if the demon

[1]Unless it is a case where a branching point occurs and thus both paths are taken.

can ignore constraints then the genotype space will be fully connected. By extension, the fitness landscape will also be fully connected except that the demon cannot violate the hill-climbing exercise. It is assumed that the demon has no omnipotence, and cannot make short-term negative decisions to get a bigger fitness pay-off later. It is not clear whether this is a valid assumption as some thermodynamically closed systems have been speculated to maximise larger outcomes despite not resulting in the highest short-term pay-off [Wissner-Gross and Freer, 2013].

Fitness landscapes can exist for both genotype and phenotype. However, the former is ill-defined without (implicit) reference to the phenotype, since fitness requires some notion of the environment and genotypes interact with the environment via their phenotype. Although the precise form of genotype-phenotype mapping is still unknown, it is known that the relationship is of a many-to-one mapping [Wagner and Stadler, 2003]. That is, there exist many possible genotypes for a particular phenotype. Therefore, the space of possible genotypes is of a much higher dimension than the space of possible phenotypes (morphospace).

If g is a genotype in the space of possible genotypes, and $\Phi(g)$ is the corresponding phenotype then the fitness of a genotype $\Theta(g)$ is given as

$$\Theta(g) := \Theta(\Phi(g)). \tag{3.1}$$

Note that we assume that there does exist some mapping from genotype to phenotype. Since the dimensionality of genotype space is higher than morphospace, the morphospace fitness landscape is also of low dimensionality. Critically the dimensionality of the fitness landscape of the genotype is lower than the dimensionality of the genotype space. The optimisation restriction still applies to any path traced in either fitness landscape. However, it is now not necessarily the case that the demon can maximise the fitness gradient in the morphospace fitness landscape, since change can only be instantiated within the genotype space. There is then a difference in response, which leads to a dissociation in the optimal trajectory in genotype space versus the optimal trajectory in morphospace. There are two causes for this dissociation. Firstly, the maximum fitness gradient in morphospace may not correspond to the maximum fitness gradient in genotype space. Secondly, two points that are close in phenotype space may not necessarily be close in genotype space, and may not even be reachable in a path through genotype space.

One implication of this definition of dissociation is that increasing the dimensionality of morphospace increases the degree of dissociation, since the system now has more degrees of freedom. Erwin [1994] and Solé et al. [2003] both proposed that the increasing dimensionality of morphospace explains the pattern of the Cambrian explosion. Such a proposal would then suggest that constraint has increased over time, and in particular across the pre-Cambrian–Cambrian boundary. Arguably such an explanation could potentially explain the macroecological patterns of diversification and extinction seen after the rise of animals, discussed in Butterfield [2007]. Thus this dissociative property of the genotype-phenotype map has the potential to link the first tier, second tier, and third tier phenomena described in Gould [1985].

Note that so far there have been no claims on the fixity or otherwise of the fitness landscape; the property of dissociation remains whether the fitness landscape is ever-changing or rigid. However, if the fitness landscape is dynamic, it is arguable that unless there is some bias in the dynamics, on average the degree of dissociation will remain relatively constant. It can also be noted that under this definition, historical contingencies of phylogeny (as defined in Gould [1989]) are taken care of, since the degree of dissociation is increased with increasing structural constraint. Distributions of mutational probabilities are also captured within this definition.

The definition of constraint defined above, that is anything that causes the evolutionary trajectory to deviate the local maximum, implies that if there were zero dissociation then no constraint would occur. Is this true? For there to be no dissociation, there would have to exist a local one-to-one mapping between genotype space, phenotype space, and (by extension) the fitness landscape. Thus a single change in genotype space would correspond to a single change in phenotype space which would either increase or decrease fitness. Because the phenotype is an expression of the genotype, and there is a local one-to-one mapping, then every genotype must locally map to a phenotype. Now the question becomes whether or not every local peak in the fitness landscape in such a scenario can be reached via consecutive maximally positive fitness steps in mutation. This obviously depends on the mutation process. If there is a uniform mutational probability distribution over all genotypes (that is, any genotype may be reached by any other genotype), then there is no constraint.

If this is not the case, what causes the probability distribution to deviate from the uniform? Heritable mutations are caused by errors in copying. Therefore, for a uniform probability the error rate must be uniform across the genotype. This can only be true where there is no specific meta-regulation to prevent this, or where the kinetics of the mutation are unfavourable. The universal transition/transversion bias has recently been questioned Keller et al. [2007]. We propose that, because of this, constraint exists throughout biological systems and that, even in the most simple evolutionary case, meta-regulation, and therefore constraint, is selected for. This agrees with the results in the Chapter 6. As such, constraint is a vital component of evolution, which must be included explicitly in predictive models of evolution.

3.2.4 Conclusion

In this section, the position of constraint in evolutionary theory was established and discussed. A definition was given that considered constraint as the difference caused by the genotype to phenotype mappings. An exploration of this theoretical framework showed that even in the most favourable case, constraint would still be present due to meta-regulation of the genotype. As constraint is selected for, it seems likely that constraint affects all biological evolutionary processes, either via the meta-regulation process or the dissociative property of the genotype-phenotype map. As such, the idea of evolution as a simple optimisation process is inaccurate, and will never be able to have predictive powers.

3.3 Evolvability

3.3.1 Introduction

In the previous section, the role of constraint in evolution was considered. One facet of evolutionary processes is constraint, another is evolvability. Both can be deemed meta-traits of traits – that is, they describe the patterns of behaviour of traits over evolutionary time. In this section, evolvability is studied with respect to evolutionary modelling, and what implications the various definitions and problems of evolvability have for questions posed in Gould [1985] and Erwin [2000].

Hansen [2003] defines evolvability as the ability to respond to a selective challenge. However, whilst this is a useful measure for comparative questions (*e.g.* is trait x more evolvable, given a set of environments, than trait y?), it seems a broad definition which poses more questions than it answers. For instance, does evolvability only relate to an individual selection pressure or can it be extended to a set of selection pressures? Is there a time frame over which the evolvability is measured? Indeed, the measurement of evolvability is somewhat difficult, as there are two components which contribute to any measure: the capacity to evolve given a set of environments (often referred to as robustness in biological literature) and the degree of success once evolved, given a set of environments [Reisinger et al., 2005]. How success is measured and, in particular, over what time frame, is still a question.

Further questions regarding evolvability are asked in Palmer and Feldman

[2012], in particular what unit of selection possesses evolvability, what time frame of variation should be considered (*i.e.* standing or new), and why evolvability is not predictive of evolutionary outcomes. It is then argued that survivability, or the longevity of traits, is more important and fundamental in traits than evolvability [Palmer and Feldman, 2012]. In particular, it should be noted that there has been found no correlation between heritability and evolvability, as defined within a genetic trait framework Hansen et al. [2011] This argument begins to link first tier processes with second and third tier processes, as shall be argued later. However, whilst survivability is another meta-trait of traits (as are evolvability and robustness), this section will focus on the definition of evolvability as stated above.

3.3.2 The paradox of evolvability

Evolvability can be phrased as a trade-off between independence of traits and maximal impact of genetic mutation [Hansen, 2003]. Tension arises as independence between traits allows selective pressures to impact more successfully, as the trait's ability to respond without causing negative effects in correlated traits is increased Hansen [2003]. However, a degree of correlation allows the trait to respond to a wider variety of selection pressures, thus both independence and correlation should be selected for. Hansen [2003] found that evolvability was maximised at intermediate levels of genetic integration (connectedness), using the Lande equation. A low degree of connectedness allows for a large amount of variation, which compensates for the downsides of correlation, whilst still allowing the benefits of correlation Hansen [2003]. As correlation increases, this compensation decreases and the dependencies become too fixed to allow much evolvability Hansen [2003]. A similar argument is made in Wagner [2008], where the antagonistic relationship between evolvability and robustness is discussed, framed as a paradox (the so-called "paradox of evolvability") between a species' ability to withstand mutations against their ability to generate enough variation to withstand a changing environment. Biological systems are mutationally robust if their structure or function persist after mutations in their parts, and evolvability is framed in terms of mutations in the system which produce heritable phenotypic variation [Wagner, 2008]. A random walk notion of evolvability does not depend on robustness as a result of the many-to-one mapping from genotype to phenotype [Wagner, 2008].

One way to calculate evolvability is to model evolutionary dependence via graphs, and to look at the relative frequency of graphs and sub-graphs after t time steps. From this you would expect smaller graphs to have a relatively high frequency, whilst more complex graphs should have a lower frequency, as seen in similar work undertaken by Solé et al. [2003]; Johnston et al. [2011]. This measure of evolvability is analogous to the definition identified by Brookfield [2009] as the ability to produce new mutations which can then be selected for adaptive advantage over long time periods. This is similar to entropic systems in physics where it has recently been found that for a non-equilibrium physical system, despite the natural tendency to maximise entropy instantaneously (analogous in evolutionary processes to locally maximising fitness), such systems can instead delay immediate maximisation to instead maximise entropy production for between now and a future time [Wissner-Gross and Freer, 2013]. This is then analogous to a biological system attaining a higher fitness peak, despite the initial fitness gradient promoting an alternative (ultimately lower) fitness peak. Indeed, if we consider biological evolutionary processes as such a system (and there is no reason not to), then this seems to negate the so-called paradox of evolvability – that evolvability should not be a persistent feature of organisms as selection is short-sighted in its maximisation of fitness.

We also propose that this paradox is negated by the macroecological trends that are seen throughout the Phanerozoic, in particular the mass extinctions that punctuate this era. If we apply the same evolvability arguments to the biological system as a whole, we obtain interesting analogies. In particular, instead of treating characters as independently evolving, we now consider species as independently evolving. Clearly, this is a false assumption, as co-evolution of predator-prey species is well-documented (*e.g.* [Van Der Laan and Hogeweg, 1995; Abrams and Matsuda, 1997]). However, co-evolutionary competition generally produces more specialised species, rather than generalists [Lawlor and Smith, 1976]. Thus, competition has the effect of reducing evolvability in species. However, as argued in Hansen [2003], pleiotropy and covariance (which we can think of as symbiotic co-evolution) may increase evolvability.

We would expect this balance to be preserved over short time, but not over long time, for as diversity increases, competition increases, leading to reduced evolvability. As such, the system as a whole, as well as each species, is poorly adapted to a sudden environmental perturbation. Clearly some evolvability it maintained in such circumstances, as speciation occurs – but we would expect those species with low evolvability to go extinct. However such an environmental perturbation is caused, a mass extinction results. Such a result suggests that the third tier phenomena [Gould, 1985] are to be expected from processes occurring at the level of the first tier. We explore the claim that internal drivers such as competition (first tier processes) can be responsible for mass extinctions (third tier processes) in Chapter 8.

3.3.3 Conclusion

In this section, definitions of evolvability were considered, as well as possible empirical measures. The "paradox" of evolvability was examined and found to be misleading. However, the observation that evolvability is not particularly heritable was proposed as a fundamental link between first tier phenomena and third tier phenomena. This line of reasoning is continued in Chapter 7, where fluctuating resources are added to the evolutionary model introduced in the following chapter. The next section introduces a theoretical basis for the model introduced in Chapter 4, and builds on the theoretical considerations discussed in this section and Section 3.2.

3.4 A Meta Model for Evolvable Systems

3.4.1 Introduction

Scientists must be very careful about over-fitting models to data, particularly when the data is obtained solely from past, highly contingent, events. Ultimately Darwin's theory of natural selection is not useful if evolution is so contingent that it cannot be used to extrapolate from the past to the future. Conversely, if all that can be made are broad generalisations which fail to capture relevant intricacies, then its usefulness is again called into question.

Both constraint and evolvability are traits which can be evolved over time. Approaching questions which rely on ensemble averages of outcomes (*e.g.* will life always become more "complex", how much does constraint control evolution?) are very hard to tackle because life has only evolved once. The constraints that exist on current evolutionary processes have been evolved and are either evolutionarily advantageous, or are sufficiently entrenched in life processes that they cannot be altered.

In order to create "laws" of biology, there needs to be an identification of the constants. But without ensemble averaging, how can constants be discovered? One way of identifying which parameters are fixed and which are a product of the system is to examine predictive models that explicitly include or explicitly exclude the parameters. This gives a first approximation of the impact of changes in the parameters, and thus whether they are fixed internally or externally (within or outside the evolutionary system). This section considers the basic principles behind evolutionary theory, and asks whether any light can be shed on such parameters.

3.4.2 What is an evolvable system?

In *Gödel, Escher, Bach: an Eternal Golden Braid*, Hofstadter discusses the differences between meaningless and meaningful interpretations of symbols, and how formal systems attempt to embody an isomorphic mapping of reality.

> First, we can have a *meaningless* interpretation, one under which we fail to see any isomorphic connection between theorems of the system, and reality. Such interpretations abound – any random choice at all will do.
>
> ...
>
> The other kind of interpretation will be called *meaningful.* Under such an interpretation, theorems and truths correspond – that is, an isomorphism exists between theorems and some portion of reality. That is why it is good to distinguish between *interpretations* and *meanings.*

In mathematical modelling there are similar degrees of "meaningful" and "meaningless" mappings between the phenomena being studied and the model being built. Since mathematical models are attempts to formalise reality within a consistent system, the analogy appears appropriate. Typical methods of evaluating models include how well results fit empirical data, and the ability of the model to predict outside its initial scope (via both interpolation and extrapolation). Models of evolutionary systems are often limited by the latter method, as since life is presumed to have arisen only once on this planet models only have one set of data available. Generation of data is possible, but although the crudest mechanisms of evolution have been expounded, the fine details (particularly the mapping from phenotype to genotype) are still somewhat of a mystery, making *in silico* data suspect. The time scales involved make predictions *in vivo* even more difficult. Consequently, the evaluation of models of evolutionary systems is incomplete. However, it is only through such evaluation of mathematical models that we determine whether their mapping is "meaningful" or "meaningless".

An evolvable system is defined as a system which is capable of evolution. Note that evolution is being used here in the biological sense (as opposed to in other sciences where it can simply mean change over time). Of course, this requires a definition of what is meant by "capable of evolution". Typical definitions of evolvable systems (particularly in computer science) involve adaptability to internal and/or external stimuli – which necessarily requires some capacity for "self"-modification [Barringer et al., 2007].

Rhodes et al. [2010] states that the principle of evolution is

An evolving organism transforms itself in such a manner so as to maximize the contact with the complete environment, subject to reasonable control and understanding of the contacted environment.

Rhodes et al. [2010] then proceeds to construct a mathematical notion of this using finite state machines and Lagrangians. However, whilst this ascribes a driving force to evolutionary processes (in much the same way as entropy, or gravity, in physics), it says little about the prerequisite system (that is, the formal definition of what the force acts upon).

Since Rhodes' definition focuses on the predominantly biological side of evolution, it could be argued that a definition of life would aid formalisation. The various definitions of life given in Maynard Smith and Szathmáry [1997], covering the phenotypic definition of life ("a thing is alive if it has parts, or 'organs', which perform functions") and particularly the definition of life of Muller [1966] as any thing "having the properties of multiplication, variation and heredity", approach a definition of an evolvable system, since all living things are capable of evolution, and form an evolvable system. Maynard Smith and Szathmáry [1997] argue that chemical systems which appear to evolve (specifically, autocatalytic cycles after Gánti [1974]; Gánti and Koch [1979]) lack heritable mutation and so are not truly evolving systems.

Of course, it can be argued that some non-alive systems are capable of evolution (*e.g. viruses*). Holland [1995] describes a class of systems "complex adaptive systems" (or *cas*), which can be argued to have the capacity to evolve. These *cas* have four common properties and three common mechanisms:

1. Aggregation – the emergence of complex large-scale behaviours from interacting simpler (less complex) agents, combined with the ability to simplify such systems into categories.

2. Tagging – an important mechanism for aggregation and boundary formation, facilitating selective interaction.

3. Nonlinearity – interactions are not simply additive.

4. Flows – flows through *cas* vary over time; neither the the flows nor the networks are fixed in time. This includes *multiplier effects* and *recycling effects.*

5. Diversity – *cas* have multiple differing components.

6. Internal models – a mechanism that the *cas* uses to anticipate and predict events or scenarios, which then determines the agent's behaviour.

7. Building blocks – reusable and repeatable component parts of the world, and the ability to identify such parts.

Although there is no question that all *cas* display such properties, it is questionable as to whether the mechanisms are, in fact, mechanisms, or whether in some cases pattern and process are being confused. In particular, the internal models mechanism, which appears to imply some degree of being able to think "outside the box" so to speak – to refer to a higher level of abstraction when faced with a situation. Whilst this is not altogether implausible, it is not obvious that this is the actual process that occurs – as opposed to an abstraction of a process that produces the same results.

As an alternative, the modern evolutionary synthesis offers a commonality in its constituent disciplines. Reif et al. [2000] analysis identified five components of the Synthesis:

1. Mutation;
2. Selection;
3. Recombination in sexually reproducing populations;
4. Isolation;
5. Drift.

This is more biologically bounded than the definitions given above. In particular, it focuses on certain mechanisms that enable speciation (isolation) and variation (mutation and recombination) – both key requirements for Darwinian evolution.

A differing view to this is the causal completeness principle, proposed by evolutionary development (evo-devo) researchers [Amundson, 2005a]. It states that

> In order to achieve a modification in adult form, evolution must modify the embryological processes responsible for that form. Therefore an understanding of evolution requires an understanding of development.

This proposal, in connection with the paragraph above, offers a brief summary of the contrast between population genetics and morphology. However, whilst the above paragraph seeks to further constrain the mechanism (by specifying what units of selection must be able to do), the causal completeness principle offers a perspective of how the mechanism actually works.

Finally, consider Darwin's theory of natural selection. Darwin [1859] proposed that:

1. Organisms produce more offspring than can possibly survive.
2. A change in conditions results in better adapted individuals surviving.

Darwin formulated his ideas within a biological framework. It should be noted that the first is a property that inputs to the system must have, whilst the second is an actual mechanism of the system.

In both the *cas* case and the evolutionary synthesis case it is important to distinguish between mechanisms and properties. When defining a formal system it is ideally the case that the mechanisms are established, whilst the properties restrict the inputs to the system. In the following sections, a basic definition is given of the unit of selection and the environment, and a simple evolutionary function defined.

3.4.3 The unit of selection

The unit of selection is a term often used in evolutionary theory. In general, it denotes the abstract, but accepted, notion that individuals (by which is meant individual organisms) do not evolve, but over time populations or ontogenies do [Amundson, 2005a]. In fact, it would be far better to call it the unit of information storage, since that is the part of the unit that selection is acting upon.

Let the units of selection be binary strings of length L, and $\mathcal{U}$ to be the set of these strings.

Definition 1. $\mathcal{U} := \bigcup_{L \in \mathbb{N}} \{0, 1\}^L$.

Because any finite piece of information may be encoded as a binary string, it does not matter that in biology we find strings of the form $\{AGCT\}^L$ or $\{AGCU\}^L$. This sets up a correspondence such that these strings correspond to elements of $\mathcal{U}$. A similar definition of the unit of selection is used in evolutionary models such as genetic algorithms.

3.4.4 Environment

The other input to the system is a term for environmental factors – *i.e.* those things that can interact with and influence the evolutionary process, apart from which elements of $\mathcal{U}$ are present.

Definition 2. $\mathcal{E}$:= *the set of possible environments.*

We can decompose $\mathcal{E}$ into two parts:

$$\mathcal{E} = \mathcal{E}_m \times \mathcal{E}_i \tag{3.2}$$

$\mathcal{E}_m$ and $\mathcal{E}_i$, the former is the mutable environment set and the latter contains the immutable environment set. Immutable environmental interactions are unchanging on the time scales under consideration – they can be defined as the set of environment members which $\mathcal{U}$ cannot influence, but which can influence $\mathcal{U}$. Examples of members of $\mathcal{E}_i$ might be the weather, or physical constraints like gravity, or even further abstractions like the laws of thermodynamics. Mutable environments are changeable on the time scales under consideration, and may be changed by the course of the system. In this way, feedback loops are introduced into the evolutionary system.

3.4.5 A basic evolutionary function

Definition 3. *An* evolutionary function *for a system* $(\mathcal{U}, \mathcal{E})$ *is a function* $\Phi : \mathbb{N}^{\mathcal{U}} \times \mathcal{E}_i \times \mathcal{E}_m \to \mathbb{N}^{\mathcal{U}} \times \mathcal{E}_i \times \mathcal{E}_m$*, which leaves the* $\mathcal{E}_i$ *co-ordinate unchanged.*

Here $\mathbb{N}^{\mathcal{U}}$ tells you the population of each unit of selection simultaneously. While anything, in principle, may be encoded in the environmental part of this model it is understood that the co-ordinate in $\mathcal{E}$ should carry a relatively small amount of information compared to that in $\mathbb{N}^{\mathcal{U}}$. This is because as evolution is an accumulative process, all past adaptations to $\mathcal{E}$ are encoded with the unit of information storage.

Obviously, the definition in 3 is sufficiently vague that it provides very little constraint on models of evolution. Even so, this basic function can be used as a categorisation and comparison tool for current mathematical models of evolution. The mathematical models produced rely on a separation of time scales, where ecological time is compressed, and consideration is only given to evolutionary time scale phenomena (see Metz et al. [1996] for discussion). Provided environmental change is sufficiently slow (relative to change in the unit of selection), the evolutionary process enters a regime which allows bifurcations (effectively speciations) [Geritz et al., 2004]. We can consider the factors of $\mathcal{E}_m$ as those that change within time steps of the model, and those that change between time steps ($\mathcal{E}_m^t$ and $\mathcal{E}_m^\tau$ respectively). In the adaptive dynamics case, $\mathcal{E}_m^t = 1$, because such "fast" environmental factors are not considered, and $\mathcal{E}_m^\tau$ is more dominant. Compare this to genetic algorithms, where availability of resources and competition with other units of selection can be major drivers. In such models, $\mathcal{E}_m^t$ is the driving force behind selection. Thus there are characteristic time scales for each process. A model exploring such time scales is discussed in Chapter 7.

3.4.6 Branching processes

Earlier the unit of selection was defined to be a binary string of length L. However, there are additional properties that the string must have in order to be considered a unit of selection. There are three states that the unit of selection can occupy – birth, maintenance, death – and it progresses through these states during its lifetime. The first and last, birth and death respectively, occur because evolution is, fundamentally, a birth/death process. Deaths are a way of enacting selection by removing those less adapted to the environment and so, by default, rewarding those better adapted. A pure birth process, therefore, has no way of selecting between individuals for reproduction nor any way of removing individuals which have reproduced. As a result, the population will go to infinity having displayed a random walk through trait space and is therefore uninteresting, both from a theoretical and realistic point of view.

Birth-death processes are branching processes, as the process creates tree structures of descent. Phylogenetic trees are used to test evolutionary theory and reconstruct the sequence of evolutionary steps that led to a particular output. However, one criticism of the fossil record is that it is incomplete. Whilst it was argued in Eldredge and Gould [1972] that this incompleteness may be part of the process of evolution, rather than an inconvenience, there are many statistical methods of attempting to construct a "true" phylogenetic tree from data.

However, assume the complete genome sequences of everything that has ever lived are known and accurately dated. Is it possible to construct the "true" phylogenetic tree from this dataset? This depends on the characteristic time scale that genomes are dated to, since any relatedness measure must assume something about the nature of mutations (namely that there are very few (ideally one) between time steps). The question then becomes whether there are more mutations in a single birth than there are differences between that species and another species. Assuming this to be true results in a set of possible "true" trees, with likelihood contours (as in Billera et al. [2001]) surrounding the various mutational jumps. Therefore, until more is known about the constraints of mutation only approximations of the "true" phylogenetic tree can ever be known. Using only genotypic information and the rate of mutation, the uniqueness of parents cannot be established. This is even more problematic in the case of bacterial evolution, which features lateral gene transfer.

3.4.7 Conclusion

In this section, the theory of evolvable systems was discussed and a comparative consideration given to different definitions. A simple evolutionary function was developed, which highlighted the importance of characteristic time scales. This was then used, along with the observations made in 3.2, to establish that the "true" history of phylogeny can never be established, even in the best possible case. As such, assumptions must be made about the nature of the evolutionary process, and such assumptions must be defined as fixed parameters, which are external to the system, rather than internal parameters which may be subject to evolution.

3.5 Conclusion

Although evolution is often reduced to natural selection, there are many processes and intricacies which seem to prevent evolution from proceeding as an optimisation exercise. In the first section of this chapter, three models of evolutionary processes were reduced to their components, to assess what they could and could not explicitly model. This was followed by a theoretical treatment of the main meta-traits of traits – constraint and evolvability. Following the concept of constraint to its natural conclusion, it is found that constraint will always occur in any evolvable system, due to the nature of heritable mutations, meta-regulation, and the dissociative property of the genotype-phenotype map. This suggests that constraint should be explicitly modelled in all evolutionary models which aim to have a predictive effect. It is possible that such an observation will lead to more basic laws of evolution being discovered proposed.

Interesting conclusions also result from the discussion of the paradox of evolvability in Section 3.3. Evolvability has the potential to link first tier processes with third tier phenomena, by tying processes at the genotypic and phenotypic level to mass extinctions. As evolvability decreases over time, the ability to adapt to sudden shifts in environment is lowered, and the species become more constrained, until a mass extinction is inevitable. Such a result is due to the nature of characteristic time scales, which are properties of the unit of selection discussed in Section 3.4 and which are explored in Chapter 7. In this section, a formal definition of an evolvable system is discussed, along with the properties of the unit of selection and the environment. This leads to the development of a basic evolutionary function which ties characteristic time scales inherently to the unit of selection. Finally, observations are made about branching processes which suggest that even with complete information the "true" phylogenetic tree cannot be constructed without assuming properties of the evolutionary process. Among such properties are constraint and evolvability, which are still poorly understood and not explicitly modelled in evolutionary models.

Chapter 4

Evolutionary Model Description

4.1 Introduction

In this chapter we define the model that is used in Chapters 5,6, and 7. This thesis is particularly concerned with linking the first, second and third tiers of evolutionary processes as described in Chapter 2 and defined by Gould [1985]. We approach this paradox in this chapter by developing a model of hierarchical structures which evolve.

Morphological form has long been vital to taxonomy, and although advances in genetics have provided phylogenetic clues for extant organisms, palaeontologists must still use form as their main classification data. Charles Darwin's important observations on biological form and, in particular, modifications in form were critical to the development of his theory of natural selection. However, form has been under utilised in the mathematical modelling of evolution (*e.g.* the areas of population genetics and adaptive dynamics), having been relegated as the preserve of palaeontologists and therefore interesting only as a product of evolutionary processes [Amundson, 2005b]. Form has fared somewhat better in evolutionary pattern formation and self-assembly models [Solé et al., 2003; Johnston et al., 2011], and evolutionary pattern prediction in morphospace analysis [Pie and Weitz, 2005; Niklas, 1999; Costello et al., 2008]. However, whilst such models describe the results of selection on form, and may even hint at the underlying dynamics, they fail at closing the feedback loop of the constraint form imposes on evolution and vice versa.

Of course, it could be argued that if evolutionary processes can be modelled sufficiently without form there is no need to over-complicate models by incorporating it, especially since a lack of clear understanding of the genotype-phenotype map means large assumptions must be made to do so. Nevertheless, macroecological observations over deep time suggest that, since the initial radiation of the Cambrian explosion (approximately 550 Ma), life on earth has undergone an evolutionary shift in both tempo and mode linked to the rise of animals [Butterfield, 2007]. This "explosion" of diversification and niche construction has resulted in the meta-stable biosphere seen today [Butterfield, 2011]. There are two potential explanations for this shift in macroevolutionary and macroecological behaviour – the new heterotrophic dynamics of the system or the evolvability of modular forms. We propose in Chapter 5 that it is the increase in degrees of freedom (due to modularity) that allows this increase in diversity, and we propose in Chapters 7 and 8 underlying causes the changing dynamics.

In this chapter, a model is developed which explores the evolvability of forms within the framework of evolutionary dynamics. The model is constructed in such a way so as to explore structure and its effect on evolutionary processes. Hierarchical structures are found in all biological systems, from the trophic interactions between species (*e.g.* food webs [Dunne et al., 2002a]), to regulatory gene networks [Erwin and Davidson, 2009]. In general, it is possible to split such structures into endogenous and exogenous classes – that is, those structures within individuals and the form of those individuals. The primary structures this model aims to investigate are endogenous. Whilst much work has been done on hierarchical structures in genetics, relatively little work has been done on the phenotypic results of such structures [Erwin and Davidson, 2009]. Such phenotypic structures naturally fall within the classification of Baupläne or body plans, which despite having a genetic basis (through Homeobox genes) are first and foremost morphological (and therefore phenotypic) features.

Such body plans are modular structures which characterise the interactions between different components of an organism. Thus an organism can be reduced to its components and their interactions (as in the definition of complexity given in McShea [1996]). Both interactions and components are hierarchically organised within this organism. A natural way to represent this set of objects is via a directed graph – referred to as a system hierarchy graph from here on. This interpretation of phenotypes as graphs can be considered as a mapping between two spaces, one containing abstract mathematical objects, the other extant and extinct species.

The main reason to use the mapping chosen here is the observation of hierarchical structures in biological systems and, in particular, living organisms. Note that such structures may be hierarchically organised in space, time, or some other space (*e.g.* fitness), and that "organisation" in this sense is functional. There are three different meaningful ways to interpret this mapping, though there are certainly more which could be proposed:

1. Body organisation. Biomolecules are the lowest level (the building blocks of organisms), organelles and membranes as the second lowest level, cells, tissues and organs, and finally whole body parts on increasingly higher levels.

2. Developmental organisation. There is a strong time dependence in developmental pathways, which has a profound effect on evolutionary outcomes [Galis et al., 2001]. Thus lower level instructions (*e.g.* polarity) must be carried out before higher level instructions (*e.g.* number of fingers) for the embryo to be viable.

3. Behavioural organisation. Here the graph can be thought of as a series of nested if-then statements. As before, lower level statements must be carried out before higher level statements.

The system hierarchy graph could encode each of these individually, or any combination – as real organisms must. In the basic model species are represented by their body plans using the first interpretation. This hierarchical depiction is an elegant way of intuitively displaying the different levels of biological organisation, which could be called the informational content of the body plan. Note that this depiction is essentially the developed information content, it does not include any of the development machinery (nor the genotype information content) for translating the genotype into the phenotype, which could be called the development information.

Each vertex within a species' body plan graph is assigned to a level. Levels represent the vertex's hierarchical position within the species graph. This is a convenient mathematical abstraction, as it has a basis in biology via the levels of organisation typically seen in organisms (for example the proteins which cause cells to specialise which can then form conglomerates such as organs) and has a direct relation to the levels of complexity defined in [McShea, 1996]. Hierarchical structures are evolved through the mutation of individual vertices, which may copy individual vertices or subgraphs, or create new vertices or subgraphs.

The choice of modelled mutations is justified by using genotype–phenotype map arguments from [Jukes and Kimura, 1984; Schuster, 2010]. In protein interaction networks, new functionality evolves through duplication and edge rearrangement [Solé and Valverde, 2008; Pastor-Satorras et al., 2003]. Edge rearrangement can be considered a point mutation (*i.e.* in a graph matrix, a change from 0 to 1 or vice versa). Rather than model individual edge mutations we model duplication of nodes and the creation of new subgraphs from these nodes - which incorporates point mutations.

4.2 Definition of the model

Species (S_n) are represented by acyclic, directed graphs G, such that $G = G(V, E)$. The vertex set, $V(G)$ is defined on finitely many different levels, *i.e.* $V = V(V_0, \ldots, V_L), L \in \mathbb{N}$. V_L is the *system level*, *i.e.* the highest level in the system. The vertex in set V_L is denoted by v_L. The base level is defined as V_0, and closes the system hierarchy from below. A level V_i is lower than level V_j in the hierarchy established by $V(G)$ if $i < j$. This can be written as $V_i < V_j$. If a vertex v is in V_0 then it is known as a *basic building block* (of the system). There is no limit on the number of vertices per level, for levels lower than V_L. If all levels of G only contain finitely many vertices then G is called *finite*. An equivalent hierarchical set can be induced on the edge set $E = E(G)$. For $e \in E_i$, if e connects any vertex $v \in V_i, 0 < i \leq L$, to any other vertex $w \in V_j, i > j \geq 0$. Thus E_i is defined implicitly. It is required that a vertex $v \in V_i$ must be connected to a vertex $w \in V_{i-1}$. This orientates the graph G to be *downstream oriented.*

The different interpretations of system hierarchy graphs described above can be viewed as weights on the edges and vertices of the graph G, such that

$$\Phi_V : V(G) \to I_V \tag{4.1}$$

$$\Phi_E : E(G) \to I_E \tag{4.2}$$

where $I_V = (I_{V_0}, \ldots, I_{V_L})$ is the space of vertex interpretations, and $I_E = (I_{E_1}, \ldots, I_{E_L})$ is the space of edge interpretations. Thus for the first interpretation, it is enough to choose $I_{V_i} = \mathbb{N}, 0 \leq i \leq L$, where each natural number encodes a different "type" present at that level in the hierarchy. Note that in such a case, $E(G) = \emptyset$, since any edge has the interpretation "being composed of" and so need not be weighted. Integer types can have different interpretations, depending on their level in the hierarchy. Such an interpretation has the implicit assumption that only quantity adds to the function or fitness of an organism not, for example, physical location. However, physical location could be incorporated by using a hierarchical compartment model. Such models are not considered here. Note that individuals of species are considered only in the dynamical model, and are never modelled distinctly. Instead, we track the number of individuals of a species.

4.2.1 Mutation

Mutations which are detrimental to the fitness score of the species can never be established, and are therefore not modelled (following the arguments presented in

[Metz et al., 1996]). Since these species can never have higher fitness than the parent species (as we assign fitness benefits to individual vertices), we do not implement any mutation steps which involve explicit deletion of vertices or edges. This is in contrast to evolutionary models of protein interaction networks [Solé et al., 2003], where fitness is assigned to the overall structure.

Mutation is dependent directly on the number of vertices a species has, defined as $\hat{V}(S_i)$ – mutation is therefore modelled as an independent point process, affecting precisely one node in the species. The probability of any particular node being chosen for mutation is

$$\frac{1}{\sum_{n=1}^{N} \hat{V}(S_n)}, \tag{4.3}$$

which naturally results in the probability of a species (S_i) being chosen for mutation being

$$\frac{\hat{V}(S_i)}{\sum_{n=1}^{N} \hat{V}(S_n)}. \tag{4.4}$$

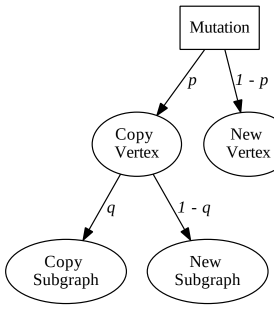


Figure 4.1: The mutation process has three possible outcomes – a vertex and its associated subgraph is copied, a vertex is copied but a new subgraph is created, or a new vertex is created with a random subgraph, at a level higher than the highest level of the vertices in its subgraph.

There are two initial possibilities for mutation – an existing vertex is copied or a new vertex is created, illustrated by Figure 4.1. These occur with probability p and $1 - p$ respectively. If a new vertex is created within a species, there is a probability η of it being connected to any other vertex, provided that that vertex is on a lower level. Therefore all subgraphs of a species' body plan are random graphs. At vertex creation a level is assigned to the vertex which represents its place in the hierarchical network. As levels increase, the hierarchical complexity of the species increases, and as vertices on a level increase the object complexity of the species

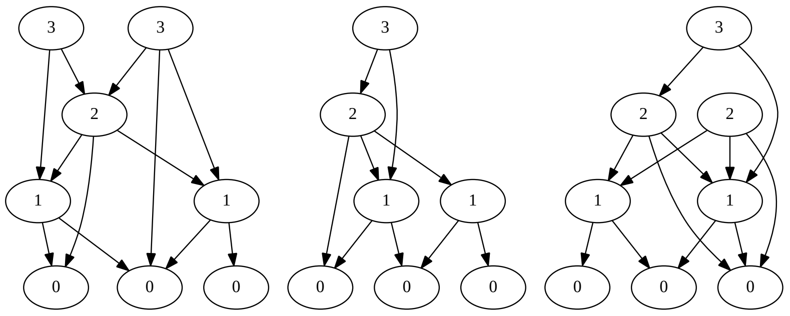


Figure 4.2: Examples of relatively simple species created by the model. The middle species could evolve into either the left or right species by copying a vertex and creating a new subgraph (left), or copying both vertex and subgraph (right).

increases (as defined in McShea [1996].

If an existing vertex is copied, there are two further possible outcomes. The vertex may be copied and its existing connections maintained (thus duplicating the subgraph's edges) or a new subgraph may be constructed (following the construction rules detailed above). These occur with probability q and $1 - q$ respectively.

4.2.2 Fitness

If v_{ijk} is a vertex i in level j of species k, then the vertex's addition to the overall fitness of the species is defined as a_{ijk}. Mutation interacts with benefit score of a vertex in the following way. The new vertex's benefit score is equal to the sum of all the benefit scores of its downstream connected vertices. Note that if the chosen vertex is of level 0, the vertex is simply copied and assigned a new benefit score (since level 0 vertices have no subgraphs of their own). Level 0 vertex benefit scores are drawn from a quasi-normal distribution, with $\mu = 0$, $\sigma = 0.2$. However, as any score must be greater than 0 (as harmful mutations are rejected), the modulus is taken of the normal distribution. Therefore, it is expected that mutations at this level cause much smaller changes in total fitness score than evolving more hierarchical structures. Thus, fitness is dependent on the structure of the species graph.

In the simplest form of the model, fitness is equal to the total summed vertex benefit scores of a species $\phi(S_k) = A(S_k) = \sum a_{ijk}$. Note that level 0 vertices (which contribute to the fitness scores of higher level vertices they are connected to) will contribute multiple times to the total fitness of the species, in direct proportion to the number of distinct subgraphs the vertex is a member of.

Competition

As discussed earlier, the environmental context of the individual is crucial in defining fitness. Normalisation of the fitness provides a way of forcing competitive interaction between species, as each species' fitness is defined relative to the fitness of other extant species.

Therefore, let the reproductive opportunity of a species S_k defined relative to the mean fitness of the population, that is

$$\phi(S_k) = \hat{A}(S_k) = \frac{A(S_k)}{S_k\bar{A}} \tag{4.5}$$

where $\bar{A} = \frac{\sum A(S_k)}{N(t)}$ and $N(t)$ is the number of species extant in the system at that particular time step. Under competition therefore species with better than average total ability have a fitness greater than 1, whilst those with lower than average total ability have a fitness lower than 1.

Structure Cost

We introduce a cost to structures, such that for $l > 0$ vertex costs are the sum of "downstream" connected vertex costs. However, the two are independently generated and normalised. Therefore, to calculate reproductive opportunity it becomes important to know the ratio of (normalised) benefit to cost, $\phi(S_k) = \frac{\hat{A}(S_k)}{\hat{M}(S_k)}$. This cost is actualised in Chapter 7, where a resource pool of components is introduced in the model, and species must compete for resources to maintain their structures.

Scaling laws

Allometric scaling laws in biology are well-known empirically, and well-studied theoretically. The most frequently used scaling law states that the metabolic rate of an organism scales as the mass of the organism to a power of between $\frac{2}{3}$ and $\frac{3}{4}$. Various hypotheses have been proposed to explain the observed difference in scaling exponent [Demetrius, 2006; West et al., 1997]. A good comparison of the two models can be found in [van der Meer, 2006]. In Dynamic Energy Budget models, maintenance is also proportional to effective organism size, this time in the form of volume [Nisbet et al., 2000]. Thus there is a clear link between the number of cells in an organism (the biomass) and associated biological processes. What is unclear is whether this relationship scales with other properties of organisms, such as object complexity or hierarchical complexity.

Given that the cost of a maintaining a structure scales with size, and that hierarchical complexity scales with size, assume that the vertex cost scales with hierarchical complexity. This is constructed by using the following metric: let m_{ijk} be the maintenance cost of vertex i in level j of species k. Then the scaled vertex cost $\hat{m}_{ijk}$ for a vertex of level j is

$$\hat{m}_{ijk} = m_{ijk}^{\gamma} \tag{4.6}$$

where γ is the scaling coefficient. Note that in the absence of level scaling, this is equivalent to $\gamma = 1$.

4.3 Further Model Development

In this chapter a model for exploring the evolution of hierarchical structures is developed. We use this model as the basis for Chapters 5,6, and 7.

In Chapter 5, we ask what the effect of different fitness landscapes is on the overall diversity of the system and to explore the effect of increasing the degrees of freedom on overall diversity. This links processes on the first tier with those on the second.

Chapter 6 looks at the second tier and the role of meta-regulation in constraining evolutionary trends. We consider the following three questions: are there values of the model parameters which maximise the change in average outgoing degree over time? How is the meta-regulation trade-off bounded by increasing object vs. hierarchical complexity? And finally we ask under what conditions the evolution of meta-regulation is promoted.

Chapter 7 extends this model further by implementing an external environment to asses the sensitivity of systems to large external perturbations and increased interaction between species, involving intra- and interspecific competition for resources.

Chapter 5

Linking the First and Second Tiers

5.1 Introduction

In Chapter 2, we described the interaction between first tier (population dynamics) and second tier (evolutionary trends) processes, as well as the lack of a coherent narrative explaining these interactions. In this chapter we approach this problem from the first tier, by asking what the effect of different fitness landscapes is on the overall diversity of the system.

Darwin proposed a gradualistic model of evolution, where speciation occurs at a steady rate Darwin [1859]. Punctuated equilibrium was proposed as an alternative model, where speciation occurs in bursts interspersed with periods of stasis Eldredge and Gould [1972]. Although both of these patterns have been shown to occur in the fossil record, it is unclear what the mechanism behind them is. Since mutation occurs at a constant rate, we would predict that a gradualistic trend should be shown in the model, rather than a punctuated equilibrium trend. If we can show the output is dependent on the fitness landscape, this would suggest a causal mechanism for why some lineages show punctuated equilibrium trends, whilst others show gradualistic trends.

We also suggest a mechanism for the change in macro-ecological behaviour since the Cambrian explosion, as noted in Butterfield [2007]. We propose that as the changes in fitness occurring now are very small due to the increase in interactions between traits replacement of species is not swift and so the apparent biodiversity is large. As the degrees of freedom in a system increase, species are able to to be arbitrarily close to one another, thus slowing the dynamics of replacement.

To translate this prediction into a testable conjecture we use the model defined in Chapter 4. In this context, we predict that models with higher numbers of degrees of freedom should have higher diversity, all other things being equal.

5.2 Single and Multiple Fitness Peaks

This section investigates the question of whether particular diversity profiles can be induced by first tier processes. It is known that increasing the interactions between traits increases the ruggedness of the fitness landscape [Kauffman and Levin, 1987]. We are interested in what effect the ruggedness of the fitness landscape is on the diversity profiles of the system. We predict that models with fitness landscapes with high ruggedness should result in high diversity, as increasing the number of traits increases the degrees of freedom of the system. To model this we compare the diversity profiles of a single fitness peak with multiple fitness peaks, with both cost and benefit of the species considered.

5.2.1 Single fitness peaks

A single peaked fitness landscape can be modelled by using a single value, and having the death rate be a function of the distance between the optimal fitness value, f^*, and the fitness of a species. For this model, the choice of fitness function is

$$f(A(S_n)) = \max((\mathrm{f}^*)^2 - (\mathrm{A}(\mathrm{S_n}) - \mathrm{f}^*)^2), 0), \tag{5.1}$$

with $A(S_n)$ as defined in Equation 4.5. This fitness function was chosen as it defines a simple relationship between the ability score of the species, $A(S_n)$, and the distance from the optimal fitness value.

The selection function is based on this fitness, such that

$$\chi = \frac{1 - \hat{f}(S_n)}{2}, \tag{5.2}$$

where $\hat{f}(S_n)$ is the normalised fitness of species n.

The model was run with $f^* = 5$, $\beta = p = q = 0.5$, and the average number of species of 500 runs recorded as a function of time. These results are plotted in Figure 5.1. It can be seen that the effect of implementing a cost to species' structures (shown in green) does provide a greater number of potential coexistence opportunities. In particular, it is notable in comparison with the normalisation only model (shown in red), which after initial exploration of the space, decays to precisely

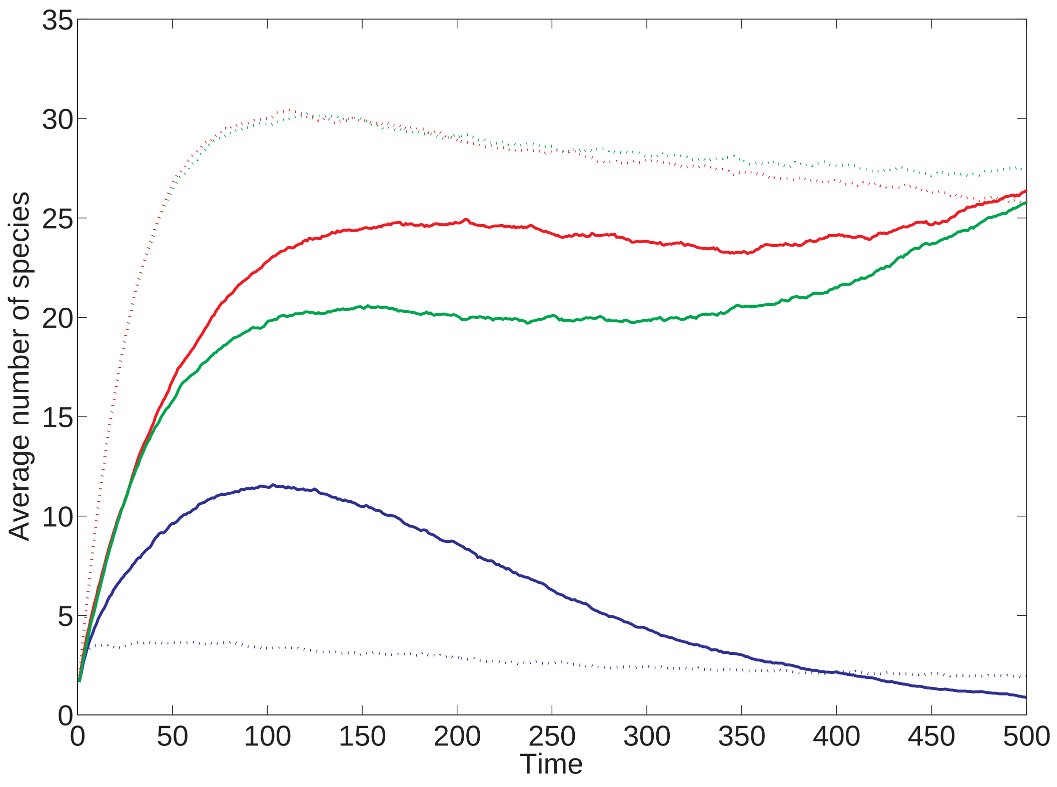


Figure 5.1: Comparison of the average number of species over time for algorithmic model, taken over 500 runs. Single fitness peak at $f^* = 5$. $\beta = p = q = 0.5$. Dashed lines comparison with model behaviour with fitness peak at $f \rightarrow \text{inf}$. Adding an extra degree of freedom in the form of a structure cost (red and green lines) cost allows for greater diversity both fitness landscapes. Red line has a structure cost has scaling $\gamma = 0.75$, whilst the green line has a scaling of $\gamma = 0.66$. Blue lines represent model without a structure cost.

one (optimal) species. This is due to a combination of factors – that there is only one variable, which is constrained in how it can evolve, resulting in a sole surviving species. Such a species need not be globally optimal, but must be locally optimal. In the case where there are two such variables (as when cost is implemented), attaining the single optimum value is harder, but simultaneously it is easier to get arbitrarily close to f^* due to higher degrees of freedom. As a result, a higher number of species can coexist. This result suggests a strong correlation between the degrees of freedom and the number of possible coexistent species. This supports our conjecture that increasing the number of degrees of freedom increases the available diversity of the system.

5.2.2 Multiple fitness peaks

We now consider whether our conjecture extends to a rugged fitness landscape. The two traits considered are the average degree and the average number of nodes per level. The fitness peaks are defined in the following way

$$f(S_n) = sin(< k(S_n) > +\sqrt{2}) + sin(\frac{v(S_n)}{L(S_n)} + \sqrt{3}), \tag{5.3}$$

where $< k(S_n >$ is the average degree of species n and $\frac{v(S_n)}{L(S_n)}$ is the average number of nodes per level. An example of the fitness landscape produced by Equation 5.3 is shown in Figure 5.2. The selection function is again defined as

$$\chi = \frac{1 - \hat{f}(S_n)}{2}, \tag{5.4}$$

where $\hat{f}(S_n)$ is the normalised fitness of species n.

Since multiple fitness peaks encourage both diversification and coexistence, we anticipate that the results from this model should show higher average numbers of species than seen in Figure 5.1. In Figure 5.3, it can be seen that this prediction holds true for the normalisation model. However, the addition of an extra degree of freedom, via cost attributed to species' structure, has no discernible effect on the diversity over time.

This does not support our conjecture that an additional degrees of freedom increases the diversity of the system absolutely, since models with a cost associated to species' structures should still produce higher levels of diversity than those without. This could be because the mutations arising in the model are too discrete and result in jumps to other peaks, rather than coexistence around the local peak. This would explain why all versions of the model have similar diversity outputs and

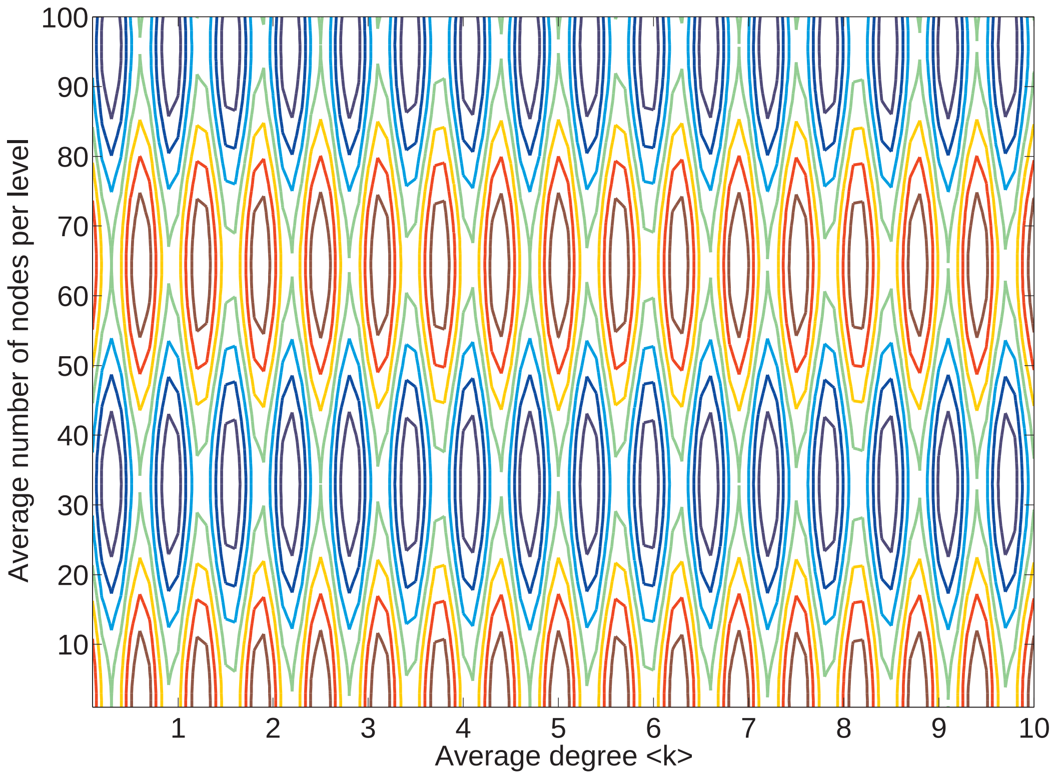


Figure 5.2: Fitness landscape produced by Equation 5.3. Blue represents regions of low fitness, red regions of high fitness.

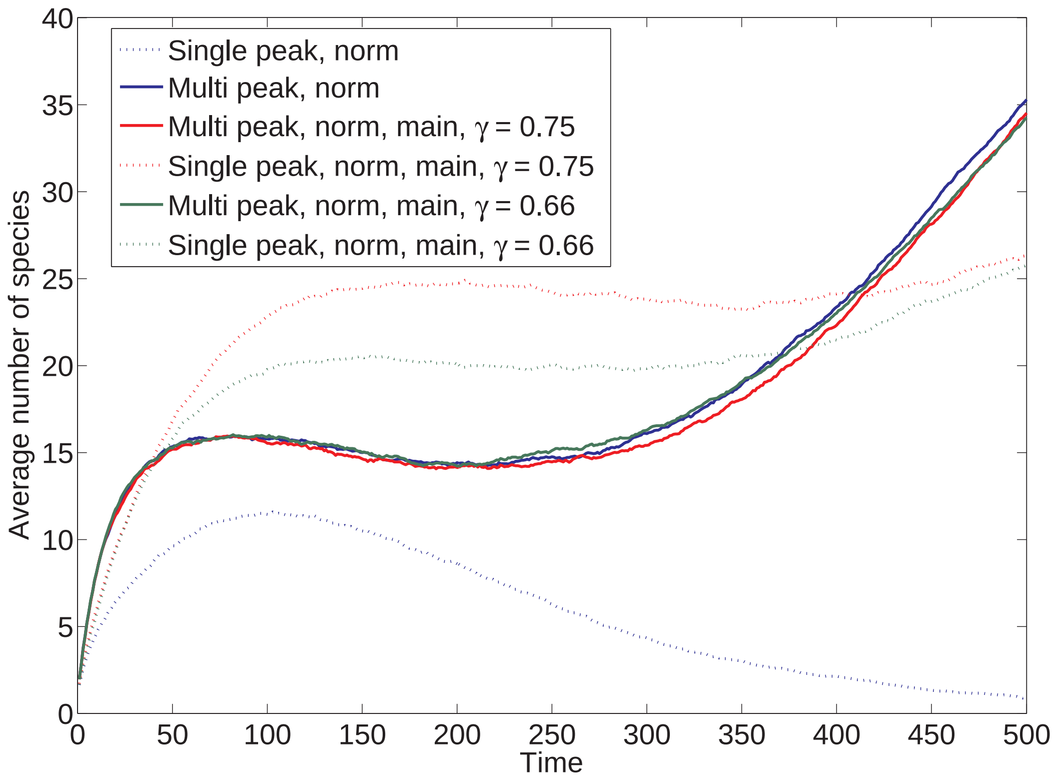


Figure 5.3: Comparison of the average number of species over time for algorithmic model. Fitness peaks given by Equation 5.3. $\beta = p = q = 0.5$. Averaged over 500 runs.

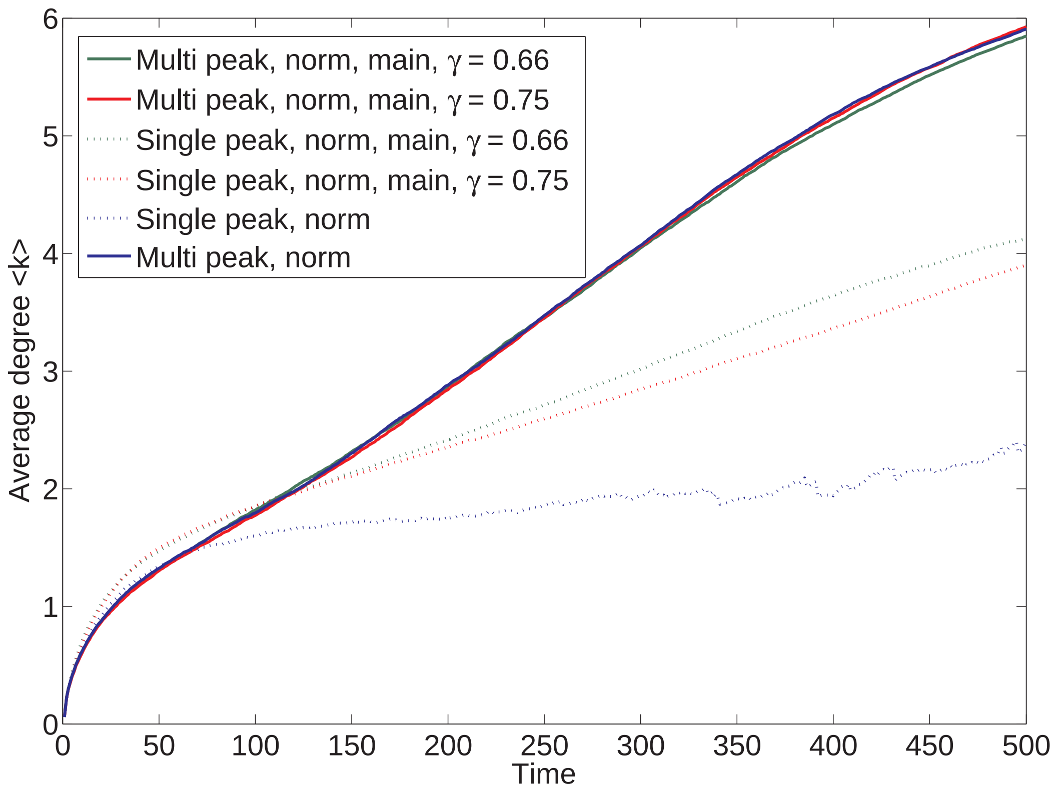


Figure 5.4: Comparison of the average degree over time for algorithmic model. Fitness peaks given by Equation 5.3. $\beta = p = q = 0.5$. Averaged over 500 runs.

why the rugged fitness landscape initially shows lower diversity than the single peak fitness landscape, as shown in Figure 5.4.

Finally, we asked whether different fitness landscapes produced different characteristic diversity profiles. We predicted that the model should show gradual change, with species being replaced by more fit species on a continuous scale. Figure 5.4 shows that, during the period of stable diversity in the rugged fitness landscape (time steps 50 to 300) one-for-one replacement of species is occurring, as the species in the system show overall increases in fitness. We then see rapid diversification of the system. This supports both gradualistic and punctuated change (that is, rapid speciation events over relatively short time periods) and suggests that both modes should be seen in the system. The presence of a rugged fitness landscape enhances this behaviour (compared with the single peak fitness landscape), suggesting that systems with higher degrees of freedom should have more diversification events.

5.3 Discussion

This chapter focussed on linking the first and second tiers, as described in Gould [1985] and discussed in Chapter 2. We asked whether different fitness landscapes could induce different diversity profiles in the model. We find that the model exhibits both rapid evolutionary innovation (characteristic of punctuated equilibrium) but also gradualistic change. This is enhanced by the ruggedness of the fitness landscape, and by the presence of a cost to species' structure in the single peak fitness landscape. However, we do not find that the ruggedness or lack of ruggedness induces a particular mode of speciation.

The single peak fitness landscape showed that as the number of degrees of freedom in the species under selection increased, the diversity also increased. This agrees with our initial prediction, and we propose that this is because with a higher combination of traits, more species can coexist at arbitrarily close points in the landscape. This slows the rate of competition between species, resulting in an artificial inflation of the diversity of the system.

In the multiple peak fitness landscape it was shown that although this system shows higher levels of diversity than the single peak fitness landscape, as would be expected, due to the colonisation of multiple peaks. However, in this case the additional degree of freedom did not have a discernible effect on the diversity of the system. This suggests that if the fitness landscape exhibits high degrees of ruggedness, the resultant diversity is high, due to interactions between traits. However, the addition of a cost to the species' structure made no discernible difference to the diversity of the system. This disagrees with our conjecture and suggests that either there either there is a maximum to the amount of diversity supported by additional degrees of freedom (with diminishing returns for each additional degree of freedom added) or that the mutations result in jumps to other local peaks, rather than coexistence around a local peak. However, our model predicts that where there is strong directional selection, increasing the degrees of freedom in the system will result in increased diversity.

Chapter 6

Meta-regulation in the Second Tier

6.1 Introduction

In Chapter 5 we approached the paradox of the three tiers by examining the connection between the first and second tiers (population dynamics and evolutionary trends respectively). In this chapter we consider the second tier and, specifically, the role of meta-regulation in constraining such evolutionary trends. We propose that meta-regulation is one of the more important regulators in evolutionary trends, being responsible for constraint as well as aiding evolvability and altering the tempo of evolutionary processes.

The evolution of meta-regulation is still a mystery [Vinicius, 2010]. Network analysis has been applied to a wide range of phenomena and structures, both biological and technical [Albert et al., 1999; Albert and Barabási, 2002; Newman, 2003; Mason and Verwoerd, 2007]. Many of these graphs are hierarchical in their organisation – that is, they have specific levels which influence interactions between nodes (and thus the overall topology of the graph). However, this external information is often discarded by modellers, such that only a directed network is used when interactions are important (for example, in regulatory networks [Babu et al., 2004], though see Ravasz et al. [2002] for an alternative view).

Meta-regulation can be framed as a trade-off between adding to object complexity versus increasing hierarchical complexity. Hierarchical structures are often combined with modularity and studies have focused in the biological sciences on network motifs – generally small, repeated subgraphs which occur frequently in natural networks [Barabási and Oltvai, 2004; Mason and Verwoerd, 2007]. Models such as the duplication-divergence model aim to construct algorithms from which such motifs arise [Vázquez et al., 2003]. However, it is still not clear what the bounds to this trade-off are, and when the trade-off becomes favourable or unfavourable [Vinicius, 2010].

The causal process between the two types of complexity is also not clear – increasingly hierarchical complexity allows for greater object complexity, which has the potential to obscure the relationship between the two (analagous to the pattern vs. process arguments referenced in Chapter 2). Similarly, it would be interesting to know whether a small set of algorithms is sufficient for genetic kernels to be formed, as discussed in [Erwin and Davidson, 2009]. In this chapter we use the results from Chapter 5 to quantify the variation seen in the average outgoing degree to model meta-regulation rules.

We can model meta-regulation by considering the parameters of the model described in Chapter 4. Increasing object complexity is represented by copying nodes (with probability $1-p$) and increasing hierarchical complexity is represented by creating nodes on new levels (with probability p). β now represents the scope of the regulation – high β represents many genes controlled by few regulatory genes, whilst low β represents direct regulation.

We consider the following two questions:

Firstly, what are the values of p and β which maximise the change in average outgoing degree over time? If meta-regulation is selected for we would expect a system to tend towards these values of p and β, as the probability of creating a new node (and thus increasing meta-regulation) is defined as $1-p$. For $p = 0.5$ there is a balance between increasing object complexity and increasing hierarchical complexity, whereas for $p < 0.5$ there is a bias towards increasing hierarchical complexity and for $p > 0.5$ there is a bias against this. We can interpret this as a trade-off between increasing object vs. hierarchical complexity. We would expect $\beta \neq 1$, as newly created nodes which control all downstream nodes would, in an evolutionary context, require perfect copying to replicate accurately. We would expect this to be too resource intensive. However, the more nodes the new node is able to regulate the greater the efficiency saving (that is, one node controlling several nodes is more efficient than one node controlling a single node). We would therefore expect the optimal strategy between creating more meta-regulation and increasing object complexity to be at neither extreme.

Secondly, we ask under what conditions is the evolution of meta-regulation promoted? Specifically we are interested in whether there is an analytical steady state towards which a system will tend. From the Chapter 5 results (particularly

Figures 5.2 and 5.6) we would expect not. In biological systems we have yet to see a limit on the levels of meta-regulation, particularly if the definition is extended to include societal structures (*e.g.* ant colonies) [Vinicius, 2010]. However, we can see that there are jumps in evolution, associated with the development of meta-regulation [Maynard Smith and Szathmáry, 1997; Vinicius, 2010]. This "punctuated equilibrium" has been explained by exploding combinatorial possibilities [Solé et al., 2003], however we propose that such jumps arise naturally under a selection regime which favours moderate connectivity and has four possible mutation outcomes. This would support observations regarding the nature of meta-regulation [Maynard Smith and Szathmáry, 1997] - that is, that there are periods of rapid innovation followed by periods of quiescence.

To answer these questions we use the duplication-divergence framework developed in Vázquez et al. [2003] and Solé and Valverde [2008]. Average outgoing degree is considered, rather than average degree, so as to emphasise the meta-regulation aspects of node degree (*i.e.* the number of nodes controlling other nodes, rather than the number of nodes being controlled).

6.2 Duplication-Divergence Model

The basis for the analysis in this chapter is the work found in Vázquez et al. [2003], following Solé and Valverde [2008]. In Vázquez et al. [2003], a simple duplication-divergence model was proposed and analytically solved. These models describe an algorithmic procedure where graph growth is by duplication of nodes and edge creation and deletion according to some probabilities, α and δ, respectively. This is inspired by the biological hypothesis that protein networks evolve through protein copying, the duplication allowing new protein functions to evolve without negatively impacting existing functions [Vázquez et al., 2003].

It is shown in Vázquez et al. [2003] that the average degree, $\langle k_n \rangle$, of a vertex n in an undirected graph evolves in discrete time (where time is the same total number of vertices in graph due to the mutation rule) as

$$\langle k_{n+1} \rangle = \frac{n\langle k_n \rangle + 2\alpha + (1-2\delta)\langle k_n \rangle}{n+1}. \tag{6.1}$$

These analytics were used to show that in the duplication-divergence model, the appearance of modularity is not selected for, but arises from the rules for graph growth [Solé and Valverde, 2008]. Modularity is found to be dependent on the value of δ, and a critical value of $\delta_c = \frac{1}{2}$ exists, which separates the graphs produced into sparsely or highly connected classes [Vázquez et al., 2003].

6.3 Extended Duplication Model

As stated previously, the impact of externally imposed hierarchical rules on such models has been under-developed. Therefore, the model of graph evolution considered here differs from a duplication-divergence regime in two main ways. Firstly, graphs are directed and hierarchically structured, such that there are restrictions on node connections. This is defined by saying that a node of level l can only have outward edges to nodes of level strictly less than l. As such, all graphs formed by this process are acyclic. Secondly, there are three different possibilities for mutation steps, which change the average outgoing degree of a node in the system. The model is described more completely in Chapter 4. This process models regulatory networks and other, more abstractly hierarchical biological networks, such as organs; and to consider the impact this has on the evolutionary process. Although the graphs we consider are theoretically biologically based, the findings could equally be applied to artificial hierarchical graphs.

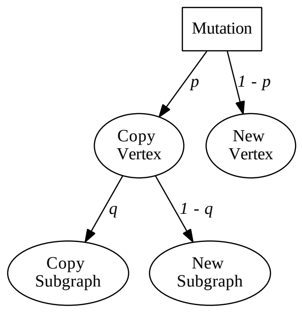


Figure 6.1: The mutation process has three possible outcomes – a vertex and its associated subgraph is copied, a vertex is copied but a new subgraph is created, or a new vertex is created with a random subgraph, at a level higher than the highest level of the vertices in its subgraph.

Figure 6.1 is a schema showing the pathways and associated probabilities associated with the mutation process. The first possible mutation step is to create a new node, on a new level, with a completely new subgraph. This happens in the model with probability p. A subgraph is generated by having a fixed probability of edge creation β between all other nodes in the species. This generates subgraphs with particular properties – for example $\beta = 0.5$ corresponds to the creation of a random subgraph. Such a mutation changes the average outgoing degree in the following way

$$\langle k_{n+1} \rangle = \frac{n\langle k_n \rangle + \beta n}{n+1}. \tag{6.2}$$

This assumes that a new node created in this way is expected to attach to βn other nodes in the species. In fact, this is an oversimplification, due to the condition imposed where nodes must attach to at least one other node, and the way this algorithm is implemented. The implementation chooses a node on level $l - 1$ to connect to, and then connects to the remaining nodes with a probability β. Thus Equation 6.2 becomes

$$\langle k_{n+1} \rangle = \frac{n\langle k_n \rangle + 1 + \beta(n-1)}{n+1}, \tag{6.3}$$

following the expectation of a binomial distribution. Obviously, the analytic equation is considering the case where there is no selection, and sequential progression of mutations is considered (*i.e.* a continuous mutation process where each mutant replaces instantaneously the parent, and there are no selection pressures).

Duplication and divergence mutations are also implemented, but these are considered as separate stages rather than the combined process described in Vázquez et al. [2003]. In a mutation event, if a new node is not created, a node in the species is copied instead. This happens with probability $1 - p$. Given this node duplication, the model then either duplicates the subgraph or creates a new subgraph (divergence) using the same rules as described for new node creation. This happens with probability q and $1 - q$ respectively. Copying both the subgraph and node changes the average outgoing degree of a node in the graph as

$$\langle k_{n+1} \rangle = \frac{n\langle k_n \rangle + \langle k_n \rangle}{n+1}, \tag{6.4}$$

which simplifies such that $\langle k_{n+1} \rangle = \langle k_n \rangle$. Thus copying a node and its subgraph doesn't change the average outgoing degree when averaged over long time. This makes intuitive sense, as there is an equal chance of any node being chosen, and thus over long time all nodes and subgraphs will be duplicated, resulting in no overall change in the average outgoing degree of a node. An extension considered later is to consider weight mutation probabilities, such that nodes of lower outgoing degree are more likely to mutate than nodes of high outgoing degree (mimicking the evolutionary process of module fixation, where some parts of genetic regulatory networks are easily modified but phylogenetically historical components less so [Erwin and Davidson, 2009]).

In the model as described in Chapter 4, one outcome of mutation copies a

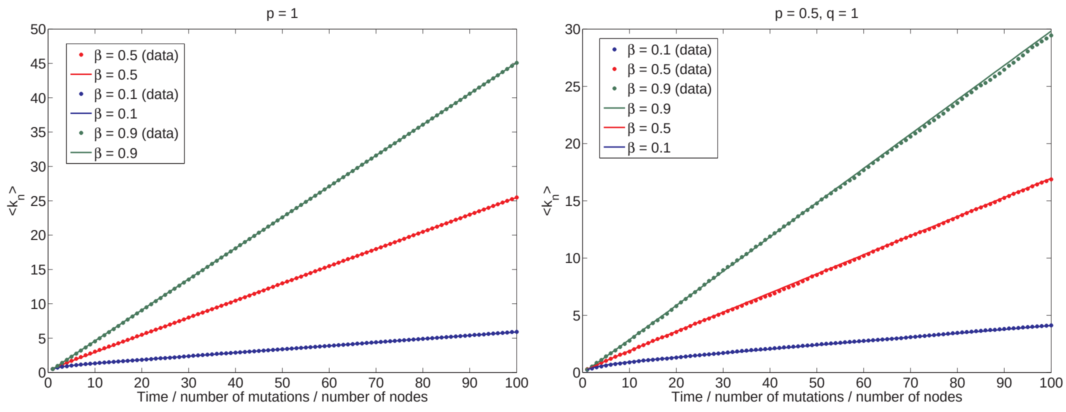


Figure 6.2: Figures showing the agreement between analytic and model data for varying values of p and q. Solid lines indicate analytic results, dots indicate model data. Model data averaged over 100 runs.

node but creates a new random subgraph. This mutation step sets up a recurrence relationship, as to model it would require complete knowledge of the distribution of nodes within the graph at the time of mutation, which is dependent on all previous mutation steps. This distribution maps to a fraction of “viable” nodes – that is, nodes which the copied node can connect to. We use λ as a proxy for this distribution, which models the proportion of nodes in the graph which are able to be connected to the mutated node (and thus have a level lower than that of the mutated node) at any one time. As such, the average outgoing degree changes as

$$\langle k_{n+1}\rangle = \frac{n\langle k_n\rangle + 1 + \beta(n\lambda - 1)}{n+1}. \tag{6.5}$$

Combining equations 6.2, 6.4, and 6.5 with the mutation probabilities results in

$$\langle k_{n+1}\rangle = \frac{p(n\langle k_n\rangle + 1 + \beta(n-1)) + (1-p)((q(n+1)\langle k_n\rangle) + (1-q)(n\langle k\rangle + 1 + \beta(\lambda n - 1)))}{n+1}. \tag{6.6}$$

The ratio between p and β determines the change in average outgoing degree, whilst λ (which is dependent on p and n) determines the long time response of the system.

6.4 Results

The analytic equations and the simulated data have a very good fit, as shown in figure 6.2. The divergence seen in some of the plots in 6.2 is due to the increasing divergence of possibilities as n increases, which requires more averaging to produce

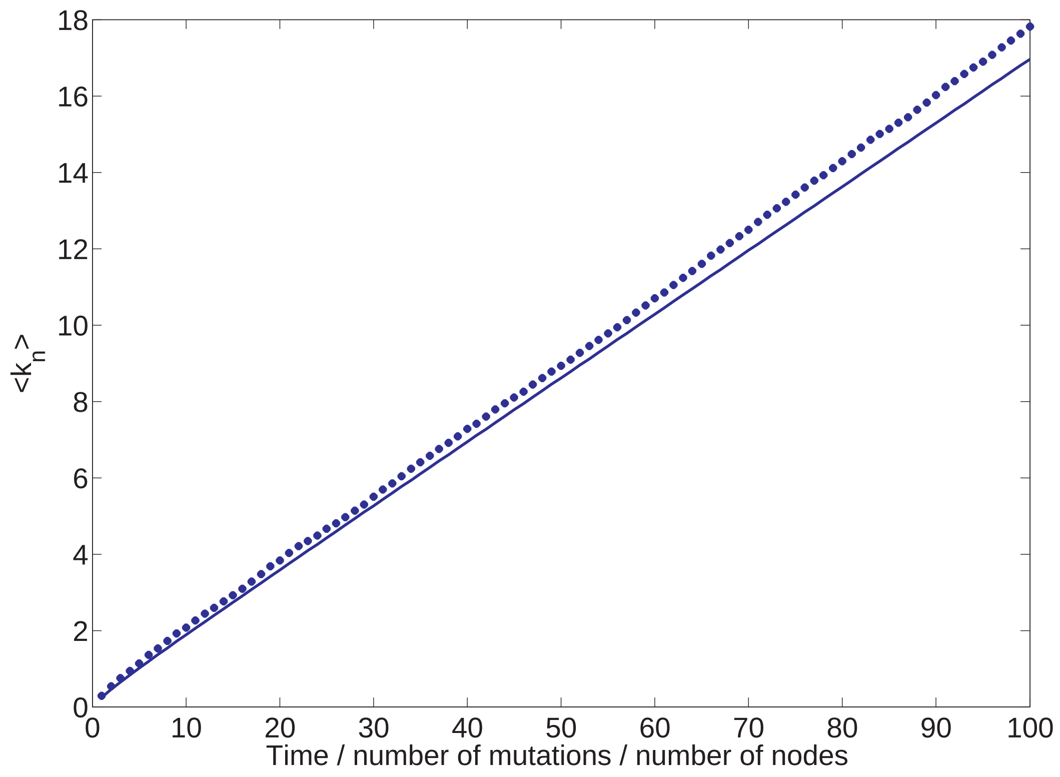


Figure 6.3: Figure showing the change in average outgoing degree with constrained mutation from the model (shown in *), and unconstrained mutation from the analytics. p = 0.5, q = 1, averaged over 100 runs.

values closer to the analytical results.

6.4.1 Maximising average outgoing degree

The first question to ask is whether there are values of p and β which maximise the change in average outgoing degree over time. An important point is to consider over what time scale the maximisation is taken – long term behaviour may be different to short or mid-term behaviour. As time scales correspond precisely to graph size, this is the same as considering optimal growth strategies for graphs of varying sizes.

Intuitively, it is obvious that for either sparse graphs or small graphs the growth rate is maximised when $\beta = p = 1$. Indeed, for all graphs with a uniform probability of choosing a node the maximum growth rate is maximised when $\beta = p = 1$. If the different combinations of mutation probabilities are considered, there are quantitative and qualitative differences in behaviour for different graph sizes. These combinations are shown in table 6.1. It can be seen that for $\beta = 1$, when

$p = 0$, $q = 0$ grows faster than $q = 0.5$ if and only if

$$n\lambda > \langle k_n \rangle. \tag{6.7}$$

As we define $\langle k_n \rangle$ to be the average outgoing degree, which is the ratio of edges to nodes in the graph, this is equivalent to saying that

$$\lambda > \frac{e}{n^2} \tag{6.8}$$

where e is the number of edges in the graph. As n increases, the inequality is satisfied since the number of edges added at each mutation step is much less than the increase in the squared function, and thus the R.H.S. will always be very small. However, $\beta = 1$ is an extreme case. If $\beta \neq 1$ then the inequality is satisfied when

$$1 + \beta(n\lambda - 1) > \langle k_n \rangle. \tag{6.9}$$

This is less trivial to satisfy. For very large n, the inequality is satisfied for sparse graphs, provided β is also small to maintain the inequality. For small n the inequality is not satisfied unless β and λ are small, and $e >> n$. However, that requires the graph to be dense, which means that $\beta \to 1$. This is as expected, since $q = 0$ will only maximise the average outgoing degree faster than $q = 0.5$ if the average outgoing degree resulting from the creation of a new subgraph is low.

It is interesting to note that when $p = 0.5$ the constrained model of mutation introduced in Chapter 4 has a higher average outgoing degree than the analytical solution predicts (see Figure 6.3). This is to be expected as the constrained model of mutation represents sampling from the tail end of the distribution of possible outcomes, whereas the analytical case takes into account all possible outcomes.

6.4.2 Steady states

A steady state is when the value of the average outgoing degree does not change, *i.e.* $\langle k_{n+1} \rangle = \langle k_n \rangle = \langle k^* \rangle$. Consider firstly the case where $q = 1$. From equation 6.6,

$$\begin{aligned} \langle k^* \rangle &= \frac{p(1 - \langle k^* \rangle + \beta(n-1)}{n+1} + \langle k^* \rangle \\ 0 &= \frac{p(1 - \langle k^* \rangle + \beta(n-1))}{n+1}, \end{aligned} \tag{6.10}$$

Assume that $p \neq 0$, then this is solved by

$$\frac{1 - \langle k^* \rangle}{n-1} = -\beta. \tag{6.11}$$

As β and $\langle k^* \rangle$ are constant, but are seen to be dependent on n (which is varying), this is a contradiction. Therefore no steady state exists for $q = 1$.

For $q = 0$, the same analysis can be performed. From 6.6

$$\begin{aligned}
\langle k^* \rangle &= \frac{p(n\langle k^* \rangle + 1 + \beta(n-1)) + (1-p)(n\langle k^* \rangle + 1 + \beta(n\lambda - 1))}{n+1} \\
\langle k^* \rangle &= \frac{n\langle k^* \rangle}{n+1} + \frac{p(1 + \beta(n-1)) + (1-p)(1 + \beta(n\lambda - 1))}{n+1} \\
\langle k^* \rangle &= \frac{p(1 + \beta(n-1)) + (1-p)(1 + \beta(n\lambda - 1))}{1-n}.
\end{aligned} \tag{6.12}$$

However, $\langle k^* \rangle \geq 0$, so for $n > 0$ there is no steady state, regardless of the values of p, β or λ.

We can see that in the case where selection is present and dependent on the structure of the graph then, on average, there is not a steady state reached. However, when individual runs are considered (*e.g.* Figure 6.4) steady states are reached. We suggest that a possible explanation for the discrepancy could be that different steady states are reached depending on the mutations experienced by each system and that, as time increases, these are more likely to be those with higher average degree. This suggests that selection pressures can induce steady states in systems with no natural steady states, which implies that although this model shows no inherent upper bound on meta-regulation, one could be induced by the selection pressures within the system.

6.4.3 Selection

The effects of selection can be measured by comparing the average outgoing degree produced to the profile produced by the analytic solution using the same parameters as the model, using the population average from the model. This is shown in Figure 6.5. As expected, selection regimes which favour more hierarchical organisms produce a higher average outgoing degree than the model. This is because in the analytical case, where there is no selection, for large n, the change in average outgoing degree tends to 0. Conversely, selection regimes which favour lower average outgoing degrees should lower than the analytical results. However, it is also possible to construct a hypothetical situation in which disruptive selection occurs, which favours either very large or very small average outgoing degrees. In such a

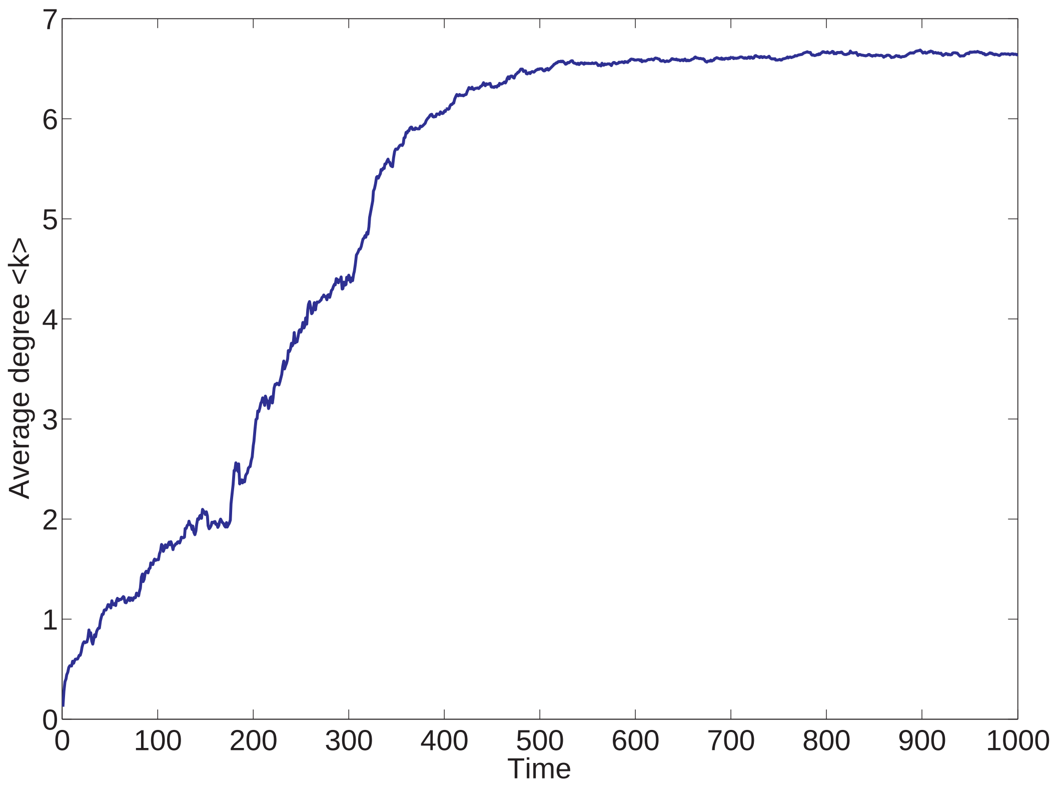


Figure 6.4: Individual run showing the change in average degree over time for algorithmic model, constrained mutation, normalisation only with multiple fitness peaks. Note the sharp jumps showing new colonisation of fitness peaks early in the run, before exploration ceases and a steady state is reached. $\beta = p = q = 0.5$.

	$p = 0$	$p = 0.5$	$p = 1$
$q = 0$	$\frac{n\langle k_n\rangle+1+\beta(n\lambda-1)}{n+1}$	$\frac{n\langle k_n\rangle+1+\beta(\frac{n}{2}(1+\lambda)-1)}{n+1}$	$\frac{n\langle k_n\rangle+1+\beta(n-1)}{n+1}$
$q = 0.5$	$\frac{(n+\frac{1}{2})\langle k_n\rangle+\frac{1}{2}+\frac{\beta}{2}(n\lambda-1)}{n+1}$	$\frac{(n+\frac{1}{4})\langle k_n\rangle+\frac{3}{4}+\frac{\beta}{2}(n(1+\frac{\lambda}{2})-\frac{3}{2})}{n+1}$	$\frac{n\langle k_n\rangle+1+\beta(n-1)}{n+1}$
$q = 1$	$\langle k_n\rangle$	$\frac{(n+\frac{1}{2})\langle k_n\rangle+\frac{1}{2}+\frac{\beta}{2}(n-1)}{n+1}$	$\frac{n\langle k_n\rangle+1+\beta(n-1)}{n+1}$

Table 6.1: Table showing how equation 6.6 changes in response to different mutation probabilities. Recall that p is the probability of creating a new node, and q the probability of copying a subgraph of a duplicated node, β is the probability of an edge being created, λ is a proxy for the fraction of nodes below a given node in the graph.

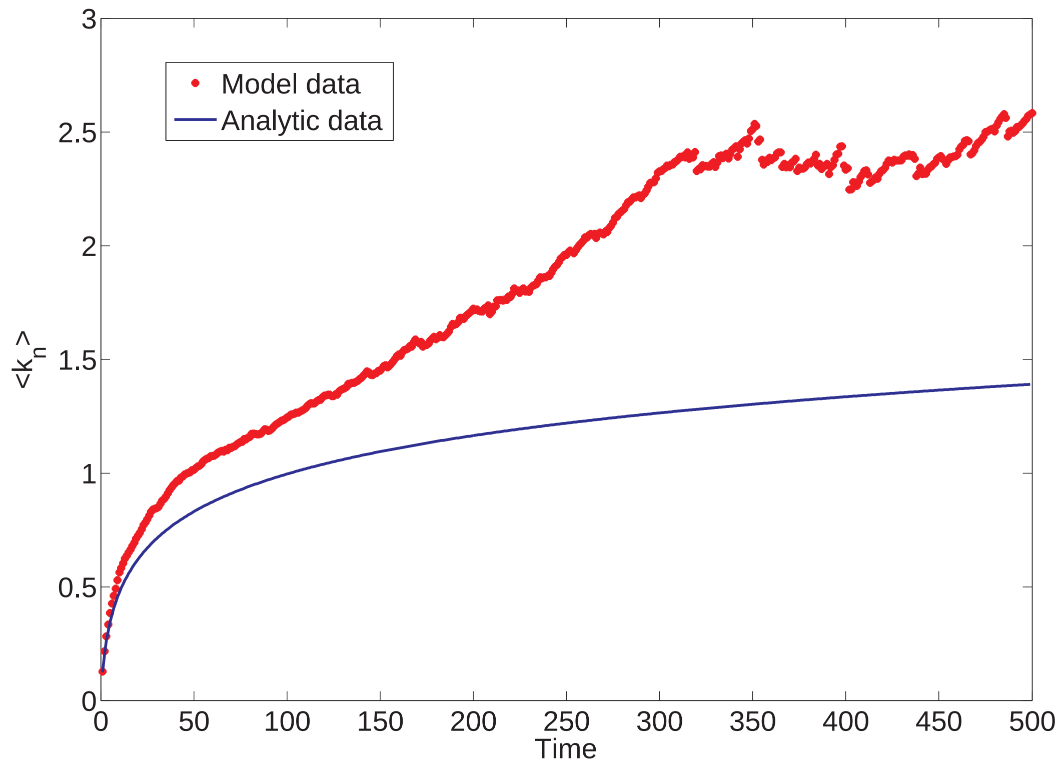


Figure 6.5: Figure showing the change in average outgoing degree from the model with normalisation and maintenance, and population average the analytics. p = 0.5, q = 1, $\beta = 0.5$, $\gamma = 0.75$, averaged over 500 runs.

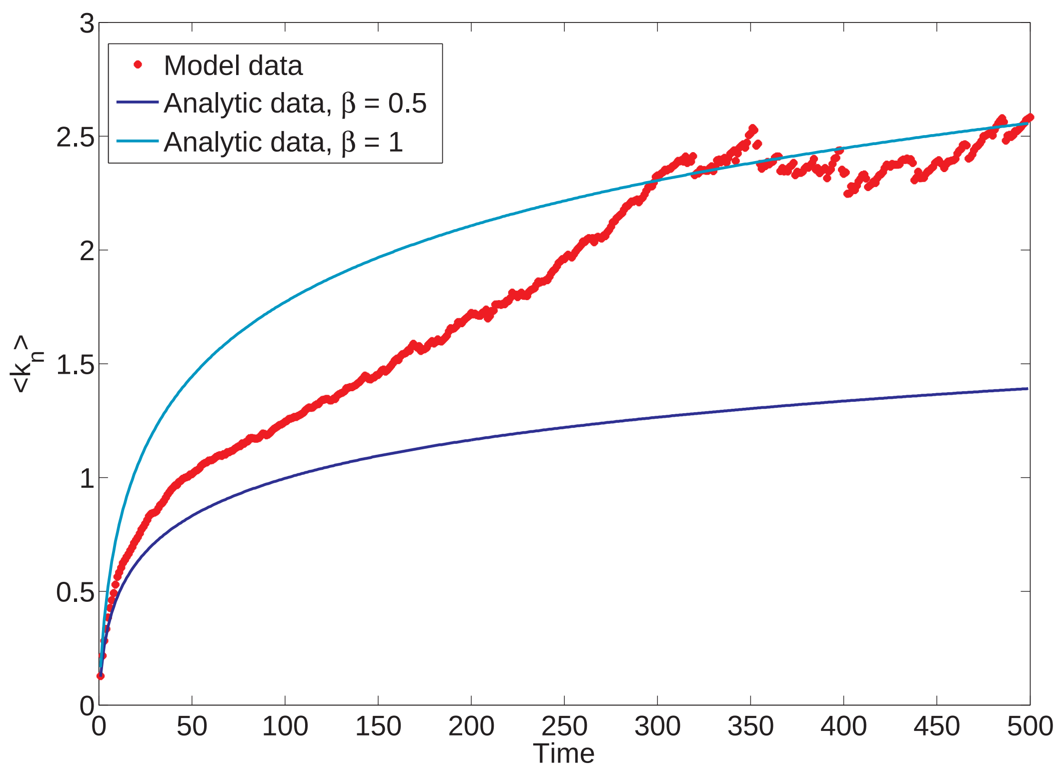


Figure 6.6: Figure showing the bounds in average outgoing degree from the model with normalisation and maintenance, and population average the analytics. The model tends to higher values of β due to its stochasticity and selection pressure pushing towards a higher average outgoing degree. Model data p = 0.5, q = 1, $\beta = 0.5$, $\gamma = 0.75$, averaged over 500 runs.

case, the average outgoing degree of that population should approach the analytical outgoing degree.

It is clear, however, that the model results are closer to the population average than the maximised case. We would expect the average outgoing degree of the model, for a given value of p, to be bounded between $0.5 \leqslant \beta \leqslant 1$. However, it is clear from Figure 6.6 that this is not the case for long time. The model does, however, tend towards the average outgoing degree values produced by $\beta = 1$ over time, suggesting that object complexity may be favoured early in the evolutionary trajectory of the system, followed by hierarchical complexity, as this ordering results in higher average degrees than the simple $\beta = p = 1$ case.

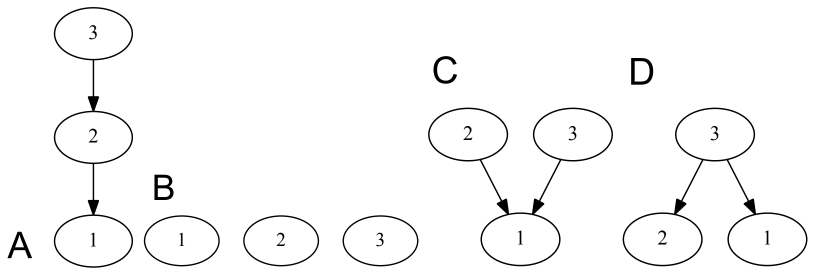


Figure 6.7: Possible graphs produced after two mutations. Referenced A, B, C, D in text from left to right.

6.4.4 Mutation Order

For the trade-off between object and hierarchical complexity to be properly considered, the analytic solution will not provide sufficient resolution, since what matters is the order in which the mutations happen. This can be shown in an example case where there are precisely 3 nodes. There are $2^3 = 8$ different mutation orderings, producing the graphs shown in Figure 6.7. The relative frequencies are such that graph A is produced by two "n" mutations (new node), which can be represented by a probability n^2, whilst graph B is produced by two "c" mutations (copy node), with probability c^2. If the process is random, by which is meant that $c = n$, graphs A and B should be seen $\frac{1}{4}$ of the time, whilst graph C should be seen $\frac{1}{8}$ of the time. Variations of D (*i.e.* 2 nodes on the base level, followed by 1 node on a level higher) make up the final $\frac{3}{8}$. This disparity arises from the many to one mapping from combinations to graph outcomes in the case of graph D.

Varying the probabilities of c and n will skew the relative frequencies of the graphs produced. For example, in the case where $c = p$ and $n = 1 - p$, then if increasing object complexity is more common than increasing hierarchical object complexity, the relative frequencies will change. However, this assumes that there are no connections between nodes. If connections exist, then the ordering of the mutations matters. This also affects the average outgoing degree produced. For example, graph D can produce an average outgoing degree of $\frac{2}{3}$ or $\frac{1}{3}$.

If connections are restricted to only one level (*i.e.* nodes on level 3 can only regulate those on level 2), then it is clear that $p = 1$ no longer produces the highest average outgoing degree. If $p = 1$ then the maximum average outgoing degree that can be reached is $\frac{n-1}{n}$. Although this is better than the average outgoing degree when $p = 0$, it is obviously not as good as a mixed strategy, since a copying step followed by a new node creation, then another copying step at the new level, produces an average outgoing degree of 1. However, this only maximises outgoing degree when

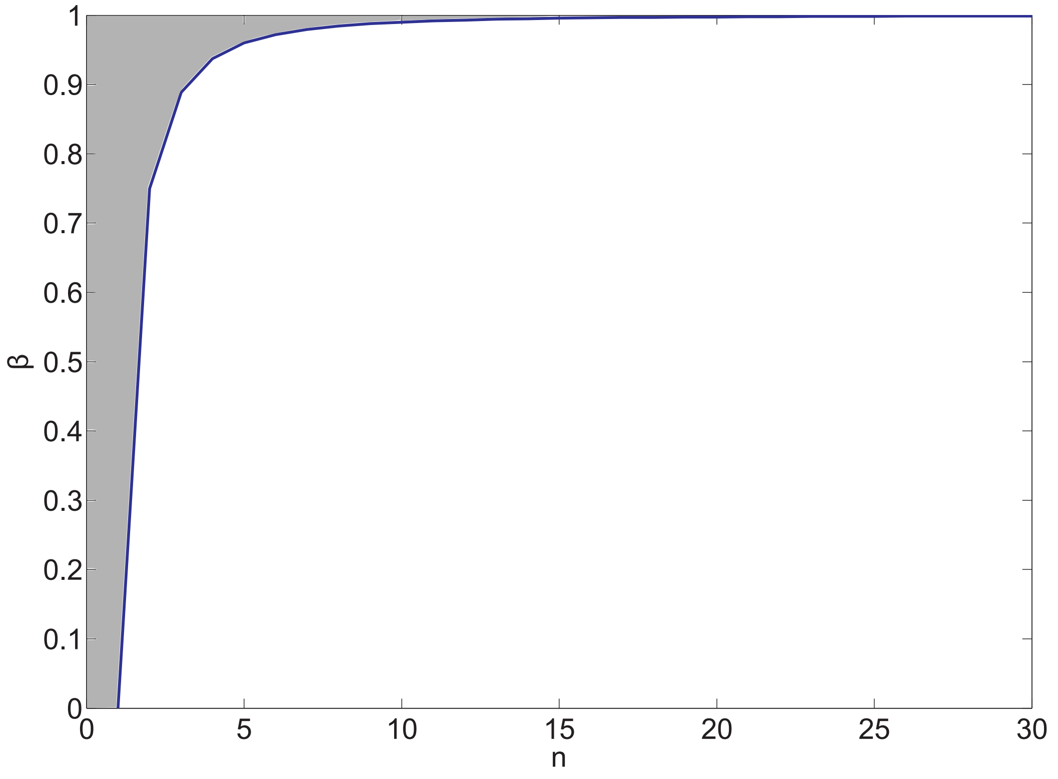


Figure 6.8: Figure showing the inequality derived in Equation 6.13. Shaded area shows combinations of β and n which satisfy the inequality $\beta > 1 - \frac{1}{n^2}$ discussed below.

the subgraph is copied along with the node. If a new subgraph is created, then this has the potential to produce a lower average outgoing degree, when $\beta \neq 1$.

If $\beta = 1$, then the average outgoing degree growth is proportional to the number of copying events. For example, following a string of $(1-p)^n$ copying steps, the average outgoing degree once the first node on a new level is created will be $\frac{\beta n}{n+1}$. This produces an average outgoing degree higher than $p = 1$ when

$$\begin{aligned}\frac{\beta n}{n+1} &> \frac{n-1}{n}\\ \beta n^2 &> n^2 - 1\\ \beta &> 1 - \frac{1}{n^2}.\end{aligned}$$

This inequality is shown in Figure 6.8. As n increases, $\beta \to 1$. Because of the restriction previously stated, this inequality applies for all levels of the hierarchy. However, once a new node has been created on a level, copying can also take place on that level. Now each node copied on a level connects to $\beta n(l-1)$ nodes. Following $(1-p)^{n-1}$ copying steps, the average outgoing degree change from copying will be either 0 or $\beta(n-2)$. This is weighted towards 0 as there are more nodes to copy, therefore the average change in outgoing degree is $\frac{\beta(n-2)}{n-1}$, for this initial copying. It can be noted that this is a similar change as the change produced by $p = 1$. However, as more nodes are copied on the upper level, the average change in outgoing degree increases. For example, in the case where the number of nodes at level 1 equals the number of nodes at level 0, $n(1) = n(0)$, then copying a node on level 1 produces a change in average outgoing degree equal to $\frac{\beta n}{2}$. This is clearly a maximum, since additional copying at level 1 will produce a lower change in average outgoing degree than this. Indeed, once the number of nodes at level 1 exceeds the number of nodes at level 0, the maximum way to increase average outgoing degree is then to create a new node on a new level. This supports a step-wise evolution of meta-regulation with periods of increasing object complexity followed by increasing hierarchical complexity.

6.5 Discussion

This chapter focussed on the evolution of meta-regulation and object vs. hierarchical complexity. These are species level traits which have the potential to direct evolutionary trends. Further, meta-regulation is seen as a key component of the major transitions in evolution [Maynard Smith and Szathmáry, 1997; Vinicius, 2010].

Using the duplication-divergence framework to analyse the model described in Chapter 4 we found that, for sparse or small graphs, the average outgoing degree is maximised when $p = \beta = 1$. However, the growth rate is also maximised when $\beta \neq 1$. Under these conditions a mixed strategy is produced - where small dense sub-graphs are sparsely connected to each other. These are analogous to the genetic kernels and modular sub-structures described in Erwin and Davidson [2009], and suggests a system with three simple mutation options and moderate selection for the average outgoing degree is sufficient to develop such structures.

The second question asked how this trade-off between increasing object and hierarchical complexity was bounded. The model does not have a steady state for average outgoing degree, suggesting that either hierarchical or object complexity (or both) is unbounded. A steady state is reached in individual runs of the model, however, suggesting that selection pressure can limit the average outgoing degree. Therefore, although theoretically there is no limit, selection pressure is sufficient to impose limits.

It is important also to note that for individual species the order of the mutation steps is important. In particular, for a given selection threshold there will be an optimum point at which meta-regulation develops. However, once this initial meta-regulation is developed, increasing object complexity maximises the average outgoing degree more than increasing hierarchical complexity. This supports our claim that meta-regulation is strongly selected for and agrees with observations regarding the nature of meta-regulation [Maynard Smith and Szathmáry, 1997] - that is, that there are periods of rapid innovation followed by periods of quiescence.

Chapter 7

Second and Third Tier Resource Interactions

7.1 Introduction

In the model as described in Chapter 4 an environment had been artificially created, using fitness functions to represent the external environment. Although this is exceedingly common in evolutionary modelling, the selection of fitness function has a large impact on the results of the model (as seen in Chapter 5). Indeed, one of the "arts" in genetic algorithm modelling is identifying a good fitness function. A simple analogy is that fitness functions act like selection functions in a genetic algorithm when they are compared to a population dynamics model. An external environment can produce non-obvious behaviour due to the increasing complexity of dynamic interactions between species and their environment. It is an additional layer of complexity compared to species interactions models such as Christensen et al. [2002]. Environments can be structured in a number of ways to explore this behaviour: spatially, temporally, and hierarchically.

Continuing the theme of uniting the first, second and third tiers of evolution by exploring the gap between first, second and third tiers we investigate how environmental variation over different time scales affects diversity. In the previous chapter we looked at intrinsic ways hierarchical complexity could be evolved, whereas in this chapter we look at fluctuations in external resources and their impact on the diversity of organisms evolved within the model described in Chaper 4. We ask the question: what role do external environments play in third tier processes?

We hypothesise that the relationship between external environments and third tier processes is dependent on the relative time scales of, on the one hand, change in the external environment and, on the other, mutation rate. Specifically, we hypothesise that

1. When the characteristic time scale of the mutation rate is smaller than the characteristic time scale of the environmental change, we will observe third tier processes tracking the environmental change.

2. When the characteristic time scale of the mutation rate is larger than the characteristic time scale of the environmental change, we will not observe third tier processes tracking the environmental change.

To answer this we introduce fluctuating resource pools to our model as an environmental change with a clearly defined characteristic time scale. We look at the number of species as a representative third tier variable. We also contrast to a simple version of the model which includes resource pools, but where these are static.

We explore the following questions within a resource model: How do fluctuating resource pools compare with static resource pools? Fluctuating resources have been used in studies to explore the effect on diversity (*e.g.* May and Mac Arthur [1972]), and on geological time scales are known to exist, as for example Milankovitch cycles. In Chesson and Huntly [1997], the assumption that harsh conditions favour coexistence by reducing the importance of species' interactions is challenged, and instead argue that fluctuations and harsh conditions can only favour coexistence if they favour niche creation. However, although such studies exist, none have tried to bridge the gap between the first, second and third tiers discussed in Gould [1985]. In this extension to the Chapter 4 model, environmental variation is explored in relation to the time scale of mutant birth rates. We would anticipate that static resource systems should exhibit boom-bust behaviour – that is, a utilising of all available resources followed by a crash, as no foresight is inherent in the system. For fluctuating resource pools, we would anticipate that some organisms should be over-fitted (that is, taking advantage of the maximum resource pool) and experience extinction due to a decreasing resource pool, whilst other generalist species (those not taking advantage of the maximum size of the resource pool) survive in the long term.

However, this question naturally extends to what time scales does environmental variation have an effect? It is easy to imagine a situation where, over sufficiently long time scales, the organisms that try to exploit the maximum number of resources will have a sufficient fitness advantage over the generalists. We predict that over very short time scales an evolutionary system cannot adapt to the changing conditions. To explore this, we predict that if the overall diversity in the system is sensitive to the environmental fluctuation, it should be shown in the power spectrum as well as the strength of the response. This quantification of the sensitivity of the system allows it to be measured, which means questions can be asked concerning critical and threshold values of both amplitude and wavelength below and above which the system responds differently. Rand and Wilson [1995] gives a method for determining critical time scales for spatially extended models, however their method focuses on internal instabilities, rather than external perturbations.

7.2 Single resource environment

The external environment is modelled solely by the use of resources, which represent an external energy source. This model naturally builds on the population dynamics model, since it models reproduction and death of individuals. However, this base model is then extended to include a maintenance step. Maintenance was explicitly modelled in Chapter 4, however, here it is a product of the dynamics. Maintenance is one of the explicit ways to model in-depth population dynamics. All biological systems require energy (thus leading to the classification of life as neg-entropic [Schrödinger, 1944]), and there are many diverse ways to "collect" the energy they require. This balance of competing demands often constrains evolution (see in particular the evolutionary adaptations of koalas and pandas as they adapt from carnivorous ancestors to highly inefficient herbivores). Maintenance represents the need for each system to produce enough energy to continue functioning. The result of not obtaining enough energy is individual death.

7.2.1 Age structuring

Although the carrying capacity is implicitly modelled, the selection function (at an individual level) is still modelled explicitly. As a result, it is possible to consider which selection functions balance the pressures of reproduction and maintenance, enabling more "complex" species to evolve and, potentially, establish.

Choosing selection functions to promote complexity firstly means defining complexity. Potential pitfalls have been briefly discussed theoretically in Chapter 3. To briefly recap, complexity in this model derives from the topology of the graph, which depends on the number of nodes and edges and their relationship to each other, and the degree of hierarchy. As such, a natural class of selection functions to explore are those which explicitly select for these features.

Alternatively, the stochasticity of the selection function can be incorporated

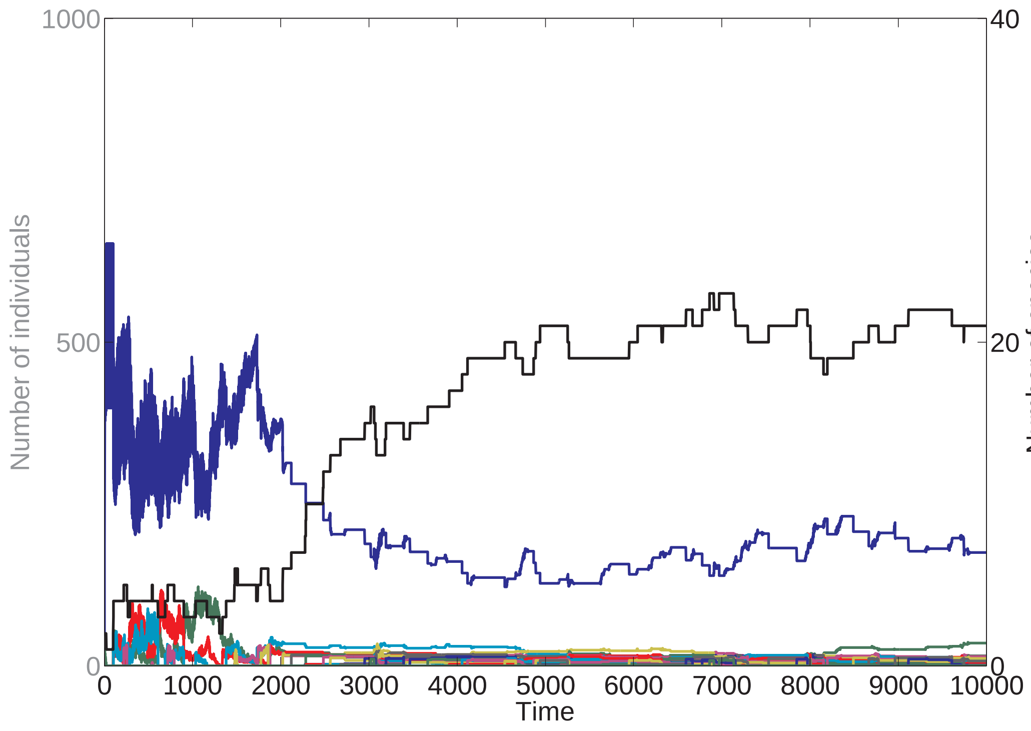


Figure 7.1: Figure showing the dynamics of a limited resource pool, where the number of individuals per species oscillates due to Lotka-Volterra style dynamics as described in the text. Different colours represent different species. Black represents number of species over time. Neutral selection bias, $p = q = \beta = 0.5$.

into the maintenance ordering, whilst a background individual death rate can be implemented. Deaths in the model are produced either by an organisms' inability to maintain itself or by age structuring the population, so that older individuals of a species have a higher rate of death. As this is a linearly increasing relationship (*i.e.* the longer the individual has been alive the higher the probability of death), we model the probability of an individuals' death per unit time as

$$P(death(S(i)_n)) = 0.5 - \frac{1}{Age(S(i)_n)}. \tag{7.1}$$

7.2.2 Limited resources

In the basic models discussed in Chapter 4 the external environment has been modelled by two distinct parameters – a fitness function and, in the case of the population dynamics implementation, a carrying capacity. The carrying capacity is now included implicitly in the model by the interaction between the absolute size of the resource pool, and the supply and demand of reproducing and dying organisms.

The result of limiting resource supply is to produce oscillatory behaviour, similar to that modelled by a Lotka-Volterra equation. The population dynamics of a limited resource supply can be seen in Figure 7.1. This results from the limited resources which causes, in the first time step, a large proportion of deaths in the population due to the inability to maintain themselves. As the system is closed and cyclic, these deaths then refill the resource pool, which allows species at the next time step to maintain and reproduce, exhausting the resource pool. Thus oscillatory behaviour is established until, after ≈ 2000 time steps, the system has one dominant species, a less dominant species in decline, and many low population species. The system is in a more stable state, with fewer oscillatory behaviours.

In the single resource system, this is the only novel behaviour compared to a population dynamics model. This is because the system at this stage is still a pure competition model – that is, any new species evolved will be competing with existing species for the same types of resource. However, there are now the existence of “fringe” species, with low number of individuals, who nevertheless persist despite the intense competition for resources. Their presence artificially inflates the diversity count of the system. This suggests that diversity may increase over time, even in highly competitive environments, if the dynamics are sufficiently slow. This was also found to be the case in Chapter 5.

In a resource driven system the order of in which species collect resources is crucial, particularly in a resource limited environment. Species that evolve are now subject to competing selection pressures induced by the existence of other species. Maintenance of a species requires collecting as many resources as there are components to the species. As the model stands, it is obvious that complexity is not at all favoured in resource limited environments – in particular, very small single node organisms are the most efficient, as additional nodes increase the number of resources needed whilst adding no benefit. This is shown in Figure 7.2. Smaller graphs are able to maintain and reproduce more easily, as they require fewer resources. However, they are also more persistent compared to larger species as, as a species, they can survive more extreme resource shortages.

Figures 7.1 and 7.2 are the result of a random environment – that is, the selection is neutral, bar the dynamics which favour small, easily maintained and reproducible organisms. This is seen by the low average number of nodes, and the continued (though decreasing) dominance of the original, single node, species in Figure 7.1.

One way of modifying this is to have resource collection be a function of the structure of the organism. This is somewhat akin to the stochastic selection

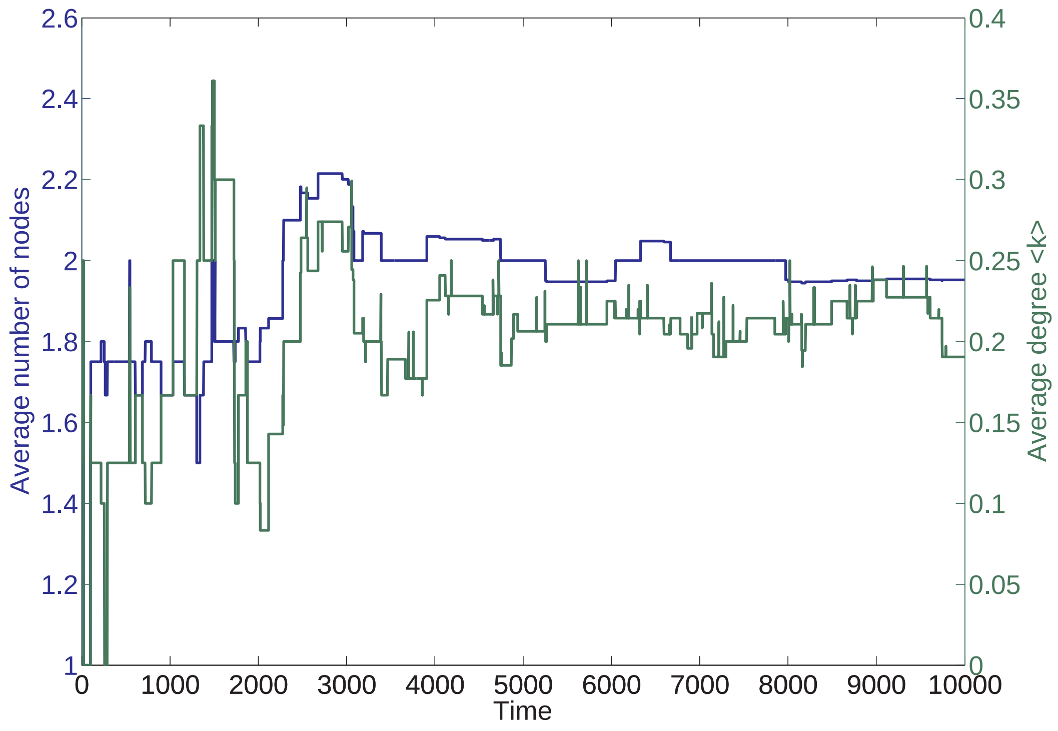


Figure 7.2: Figure showing the average number of nodes (blue) and the average degree (green) over time in a dynamic, resource-driven environment. Neutral selection bias, $p = q = \beta = 0.5$.

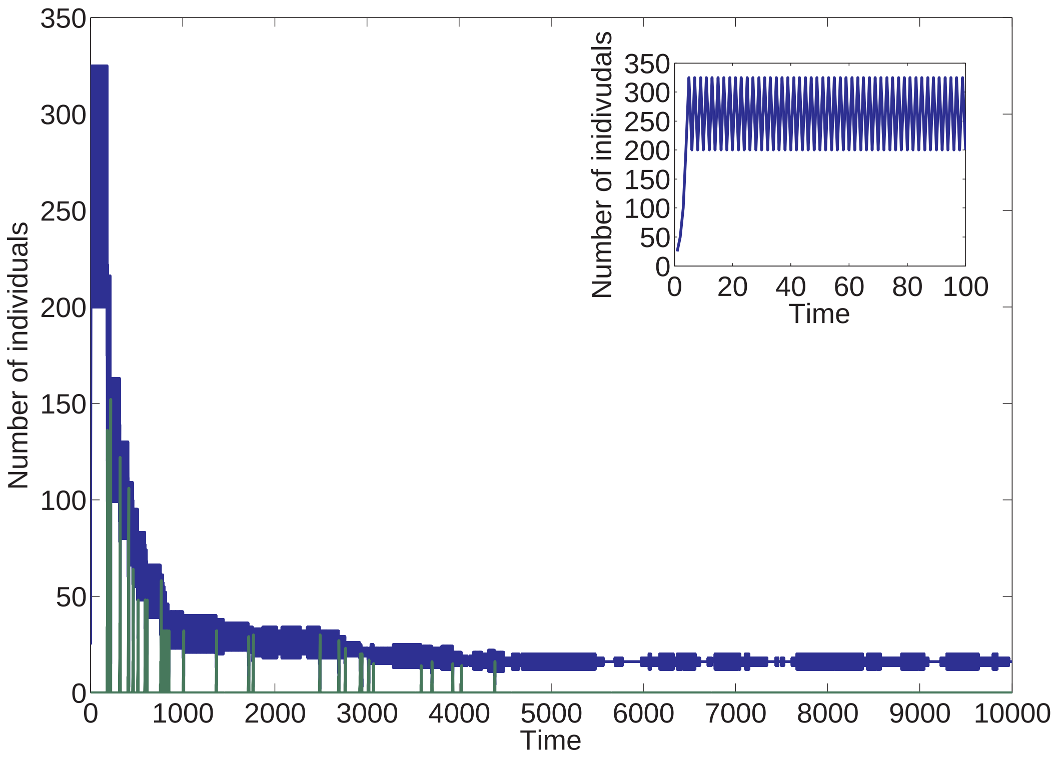


Figure 7.3: Figure showing the dynamics of a limited resource pool, where the number of individuals per species oscillates due to a “boom and bust” sequence. Green represents mutant species, which replace the resident. Inset shows close-up of oscillatory pattern for first 100 time steps. Strong edge selection bias, $p = q = \beta = 0.5$.

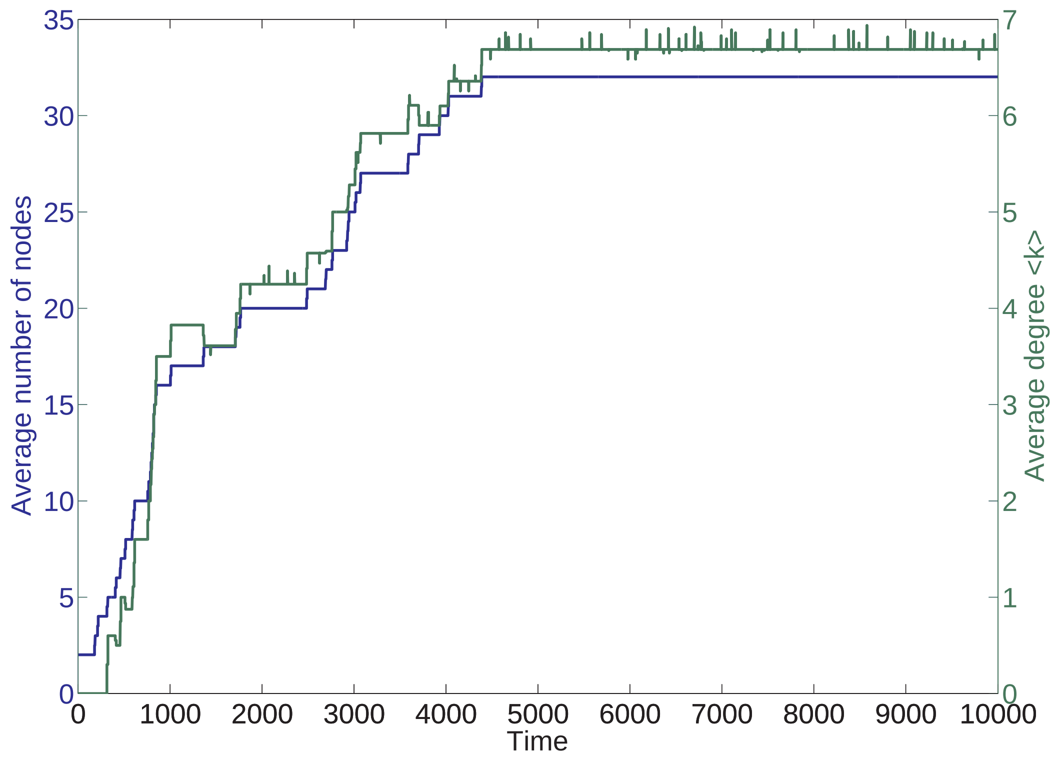


Figure 7.4: Figure showing the average number of nodes (blue) and the average degree (green) over time in a limited resource environment. Note that the strong edge selection bias means that the species evolved are of higher complexity than those in Figure 7.2. $p = q = \beta = 0.5$.

functions implemented in Chapters 4 and 5. An obvious choice for a selection function is to bias completely in favour of edges. The effects of this can be seen in Figure 7.3 and compared to the randomised model in Figure 7.1. In particular, there is a single dominating species, far less standing diversity, and a far higher average number of nodes (Figure 7.4).

A balanced selection function (*i.e.* one showing no strong bias) is achieved by modifying the random selection such that there is an edge bias – organisms with higher number of edges are more likely to acquire resources first. This likelihood is proportional to $1 - < k(S_i) >$. The outcome of this is shown in Figure 7.5. Note that strongly oscillatory behaviour only lasts for the first ≈ 1000 time steps, after which both highly populous species are in decline. The addition of new mutants does not destabilise the equilibrium and the system continues to diversify. However, after 3000 time steps, a rapid overturning of species occurs and a mutant becomes the most populous species. This is shown in close-up in Figure 7.6. This revolution

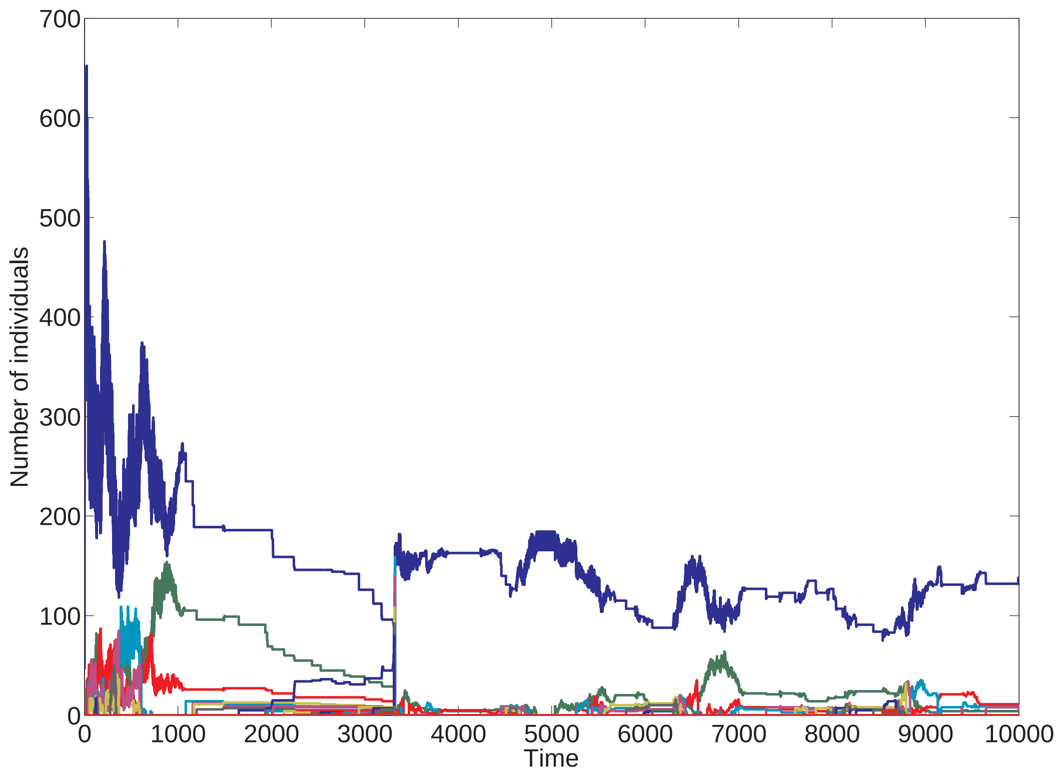


Figure 7.5: Figure showing the dynamics of a limited resource pool, where the number of individuals per species oscillates due to a “boom and bust” sequence. Different species represented by different colours. Inset shows close-up of oscillatory pattern for first 100 time steps. Moderate edge selection bias, $p = q = \beta = 0.5$.

occurs due to the multiple (relatively) concurrent deaths of lingering species, which allows the mutant to become ascendant.

7.3 Environmental Perturbation

The natural extension for a static resource pool is to explore fluctuating resources. In the earlier definition of evolvability given in Chapter 3, a characteristic time scale was defined over which organisms can react to environmental changes. An inability to react to a (relatively) sudden change would result in extinction, and an ecosystem-wide inability to react to such change would inevitably result in mass extinction. The impact of this is explored in Chapter 8.

Environmental change over time can be characterised in two dimensions – by wavelength (that is, how sudden the change is over time) and amplitude (how big a change it is). Initially we consider the effects of periodic change: we define the total

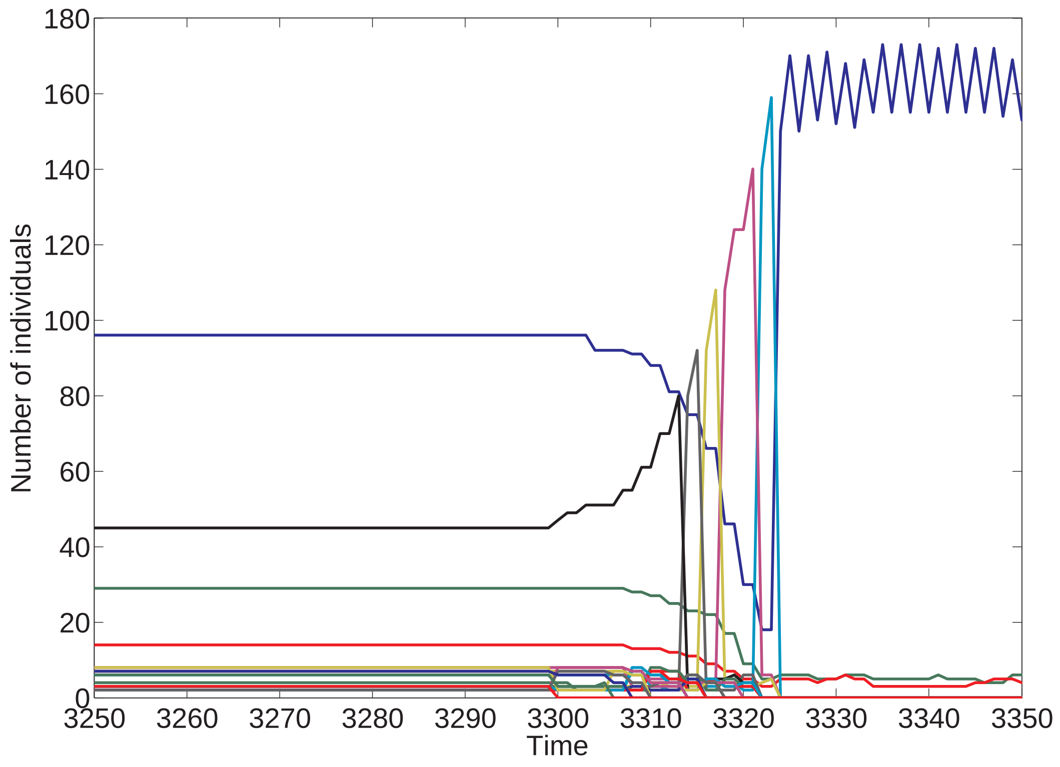


Figure 7.6: Figure showing the dynamics of a limited resource pool, where the number of individuals per species oscillates due to a "boom and bust" sequence. Different colours represent different positions – thus species 1 dies out, replaced by a species whose line turns blue. Inset shows close-up of oscillatory pattern for first 100 time steps. Moderate edge selection bias, $p = q = \beta = 0.5$.

number of resources as

$$\frac{(A_2 - A_1)}{2} sin(\frac{t}{\tau}) + \frac{(A_1 + A_2)}{2}, \tag{7.2}$$

where A_1 and A_2 are the defined maximum and minimum of the resource pool, and τ is the characteristic wavelength.

τ can be categorised into three distinct modes:

1. Long wavelength time
2. Short wavelength time
3. Random wavelength time

However, the impact of these on the system also depends on the amplitude of the fluctuations. A natural way to characterise the response of the system is by Fourier Transform analysis. If the overall diversity in the system is influenced by the environmental perturbation, this should be seen in the resultant power spectrum. The degree of responsiveness/sensitivity to the perturbation is shown by the amplitude of the peaks in the spectra.

7.3.1 Micro characteristic time scales

Micro time scales are defined to be those less than the mutation rate, *lambda*, of the system, *i.e.* $\frac{1}{\tau} > \lambda$. Because the environmental fluctuation is so fast, the system should be unable to stabilise to the conditions. This follows the ideas discussed in Chapter 3 that evolutionary units of selection are information recorders, which store information relating to the conditions which they, and their ancestors, have experienced. Thus, the expectation is that only low complexity species, that are effectively generalists, and have high evolvability, should survive, unlike in longer time scale changes, where the system can respond to the changing environment.

Figures 7.7 and 7.8 show the response at an individual level to low amplitude and high amplitude changes in the number of resources, respectively. The only qualitative change in the behaviour of the system is that of the amplitude of the change in the number of individuals, which is expected. Although there is still some stochasticity, due to age-related death, the edge bias in the maintenance order, and the Poisson process of mutation, it is the lower amplitude fluctuations which appear to show higher turnovers in species. Although this is initially counter-intuitive, it is possible that the lower amplitude environmental changes allow for more lingering species, pushing the system to an unsteady state, which then collapses simultaneously. This can be seen in Figure 7.9.

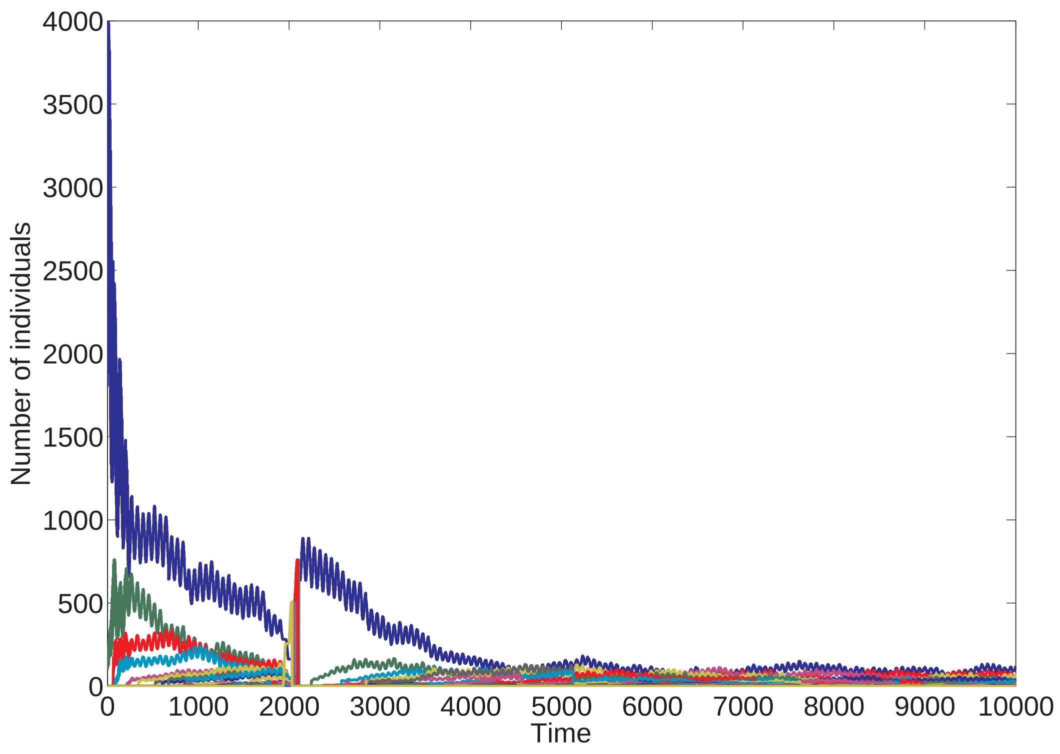


Figure 7.7: Figure showing the dynamics of a fast low amplitude fluctuating resource pool. Fluctuation given by Equation 7.2, with $\tau = 10$, $\frac{(A_2-A_1)}{2} = \frac{(A_1+A_2)}{2} = 1000$. Different species represented by different colours, dotted line gives example resource fluctuation with same wavelength, different amplitude. Moderate edge selection bias, $p = q = \beta = 0.5$. Mutation rate $\lambda = 0.01$.

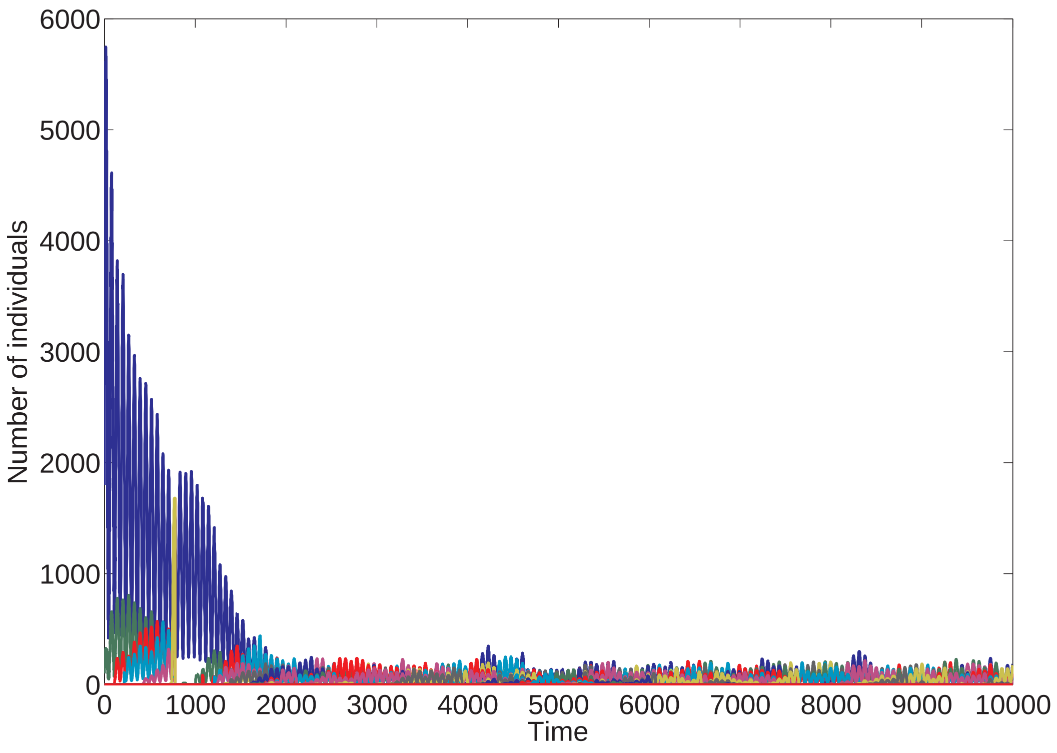


Figure 7.8: Figure showing the dynamics of a fluctuating resource pool, fluctuation given by Equation 7.2, with $\tau = 10$, $\frac{(A_2 - A_1)}{2} = 5000$, $\frac{(A_1 + A_2)}{2} = 6000$. Different species represented by different colours, dotted line gives example resource fluctuation with same wavelength, different amplitude. Moderate edge selection bias, $p = q = \beta = 0.5$. Mutation rate $\lambda = 0.01$.

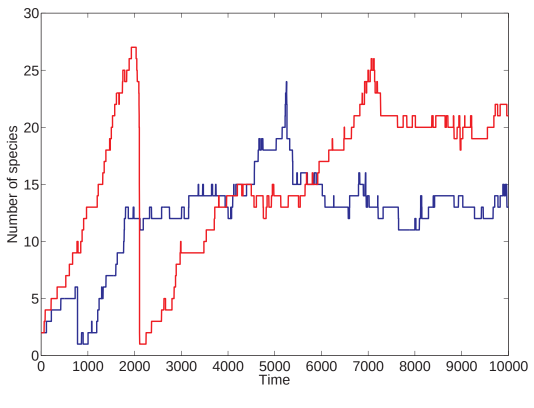


Figure 7.9: Figure showing the overall change in number of species in both high (blue) and low (red) amplitude resource pools fluctuating over fast time scales. Fluctuation given by Equation 7.2, with $\tau = 10$. $\frac{(A_2 - A_1)}{2} = \frac{(A_1 + A_2)}{2} = 1000$, represented by red line. $\frac{(A_2 - A_1)}{2} = 5000$, $\frac{(A_1 + A_2)}{2} = 6000$ represented by blue line. Moderate edge selection bias, $p = q = \beta = 0.5$. Mutation rate $\lambda = 0.01$.

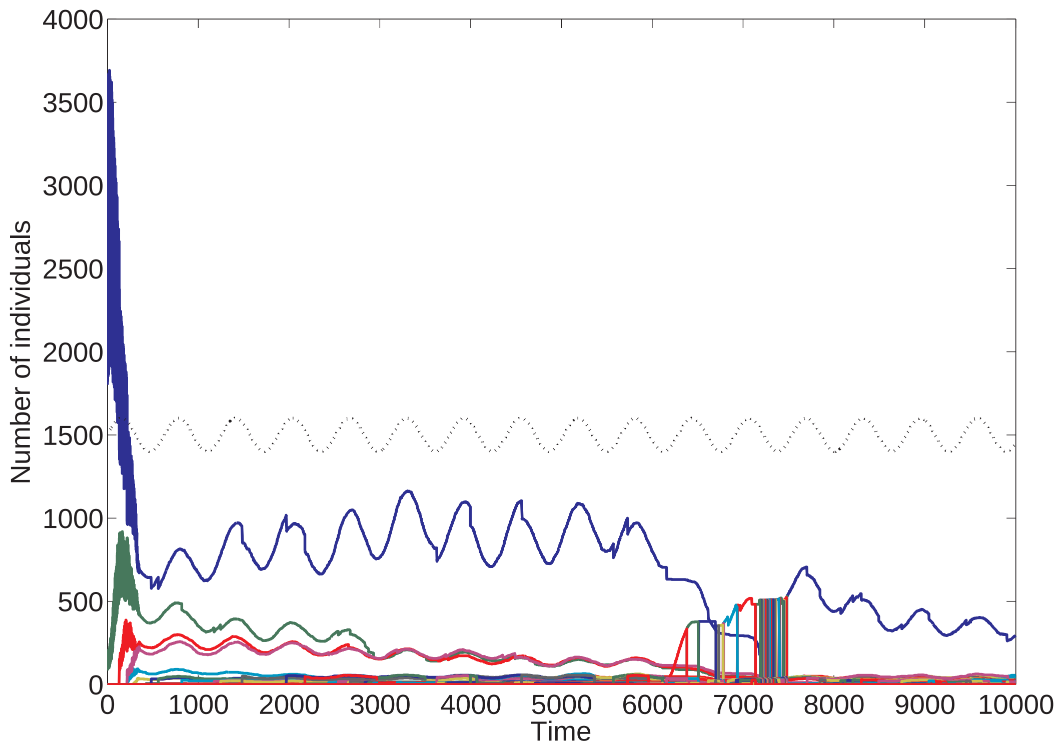


Figure 7.10: Figure showing the dynamics of a low amplitude resource pool fluctuating over meso-characteristic time scales. Fluctuation given by Equation 7.2, with $\tau = 100$, $\frac{(A_2 - A_1)}{2} = \frac{(A_1 + A_2)}{2} = 1000$. Different species represented by different colours, dotted line gives example resource fluctuation with same wavelength, different amplitude. Moderate edge selection bias, $p = q = \beta = 0.5$. Mutation rate $\lambda = 0.01$.

7.3.2 Meso-characteristic time scales

In Figure 7.10, the effect of the fluctuating resource pool is given on the individual dynamics of species. In this case, the characteristic time scale of the wavelength is the same scale as mutations occur. It can be seen that the individual dynamics are highly responsive to the fluctuations. However, the overall system diversity is not, as seen in Figure 7.11. However, the system appears to show a single large scale “boom and bust” behaviour, which was seen in the resource limited pool case described in 7.2.2. Again, we see that this is present in low amplitude fluctuations, but not in high amplitude fluctuations - similar to the response of the system under fast fluctuating resources. This would be explained if it was the rate of change of resources that was a critical driver of the diversity response system, as high amplitude changes have a high rate of change in both fast and moderate time scales.

The results in Figure 7.11 show the effect of a moderate amplitude change.

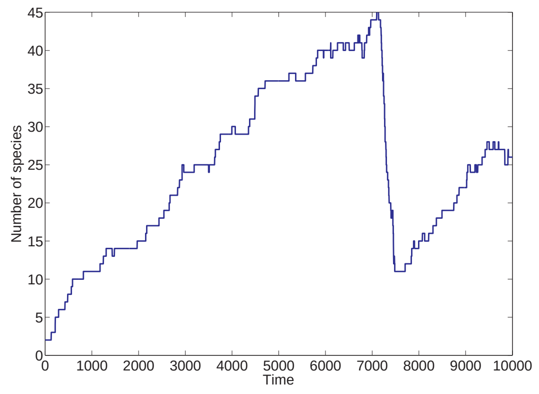


Figure 7.11: Figure showing the overall change in number of species in a low amplitude resource pool fluctuating over meso-characteristic time scales. Fluctuation given by Equation 7.2, with $\tau = 100$, $\frac{(A_2 - A_1)}{2} = \frac{(A_1 + A_2)}{2} = 1000$. Moderate edge selection bias, $p = q = \beta = 0.5$. Mutation rate $\lambda = 0.01$.

We expect that increasing the amplitude of the environmental variation should produce an increased response of the diversity scale analysis. The results of an increased amplitude are shown in Figures 7.13 and 7.14. The former shows a marked increase in the amplitude of fluctuations, but the overall population diversity again shows very little sensitivity to the fluctuations. However, if the Fourier Transform of the diversity is considered, when averaged, there is a strong response, shown in Figure 7.12. This supports the hypothesis that, over moderate time scales, the evolutionary system adapts to the fluctuating environment.

7.3.3 Macro characteristic time scales

The definition of macro characteristic time scales are those where $\frac{1}{\tau} < \lambda$. These are broad wavelength changes, and it is expected that for low amplitude fluctuations the system can adapt. For high amplitude changes, we would predict that if it is the rate of change of resource that is important then the system should show rapid extinctions (as the ratio of amplitude to τ is very close to that of the low amplitude fluctuations seen in the meso-time scale analysis). The overall diversity response is seen in Figure 7.15. This shows the response at the species level to the environmental perturbation, and an associated lag as diversification follows the rise in available resources, and then a collapse when resources decline. The species decline is rapid, but far less abrupt than that seen in the high amplitude fluctuation responses at other time scales, suggesting that these species declines are a direct response to the fluctuation. Note that the peaks produced are asymmetric, that is, that the system responds slowly to increasing abundance, but relatively rapidly to

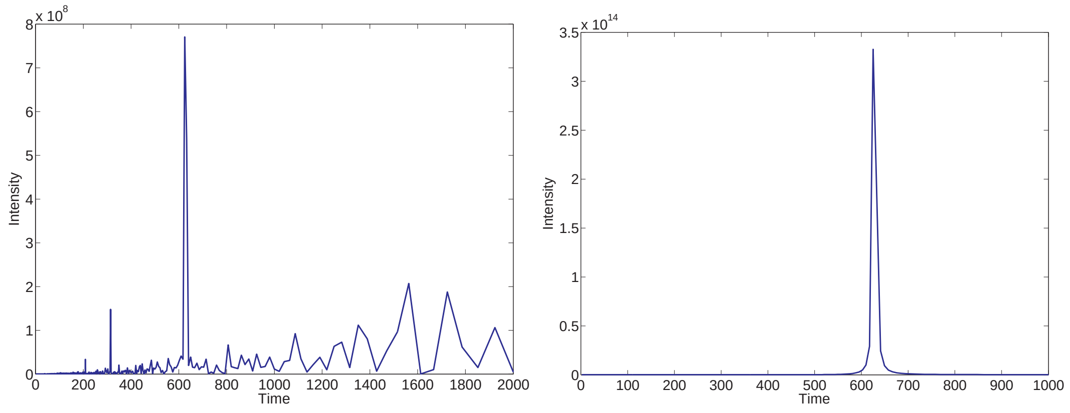


Figure 7.12: Fourier Transform analysis of the response of the average diversity of the system to a high amplitude fluctuating resource pool, fluctuating over moderate time scales ($\tau = 100$). Model data on left, averaged over 100 runs. Fourier Transform of Equation 7.2 on right.

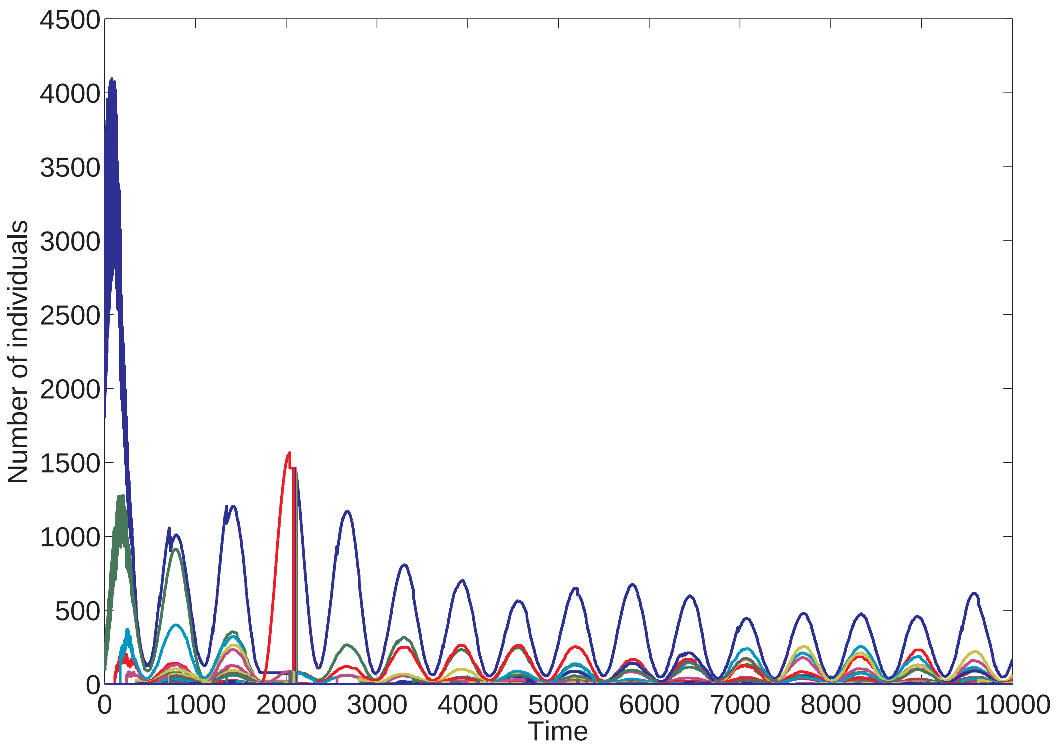


Figure 7.13: Figure showing the dynamics of a high amplitude fluctuating resource pool, fluctuating over moderate time scales. Fluctuation given by Equation 7.2, with $\tau = 100$, $\frac{(A_2 - A_1)}{2} = 5000$, $\frac{(A_1 + A_2)}{2} = 6000$. Different species represented by different colours, dotted line gives example resource fluctuation with same wavelength, different amplitude. Moderate edge selection bias, $p = q = \beta = 0.5$. Mutation rate $\lambda = 0.01$.

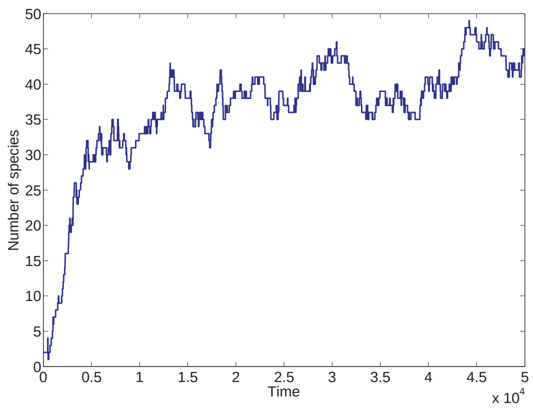


Figure 7.14: Figure showing the overall change in number of species in a high amplitude fluctuating resource pool, fluctuating over moderate time scales. Fluctuation given by Equation 7.2, with $\tau = 100$, $\frac{(A_2 - A_1)}{2} = 5000$, $\frac{(A_1 + A_2)}{2} = 6000$. Different species represented by different colours, dotted line gives example resource fluctuation with same wavelength, different amplitude. Moderate edge selection bias, $p = q = \beta = 0.5$. Mutation rate $\lambda = 0.01$.

decreasing resources. The strength of the response is shown in the Fourier Transform power spectrum plots, seen in Figure 7.17 and suggests that it is not the ratio of τ to amplitude which is important in creating multiple rapid extinctions.

The response of the system at a species level to the change in diversity for low amplitude fluctuations, as seen in Figure 7.16. As expected, this shows the characteristic rapid extinction of multiple species.

7.4 Conclusion

In this chapter, an external environment was explored, with reference to limiting and fluctuating resources. The main question considered in the first section was how an external environment would change the development and evolutionary trajectories of species. The characteristic behaviour of such resource constrained systems was to produce oscillatory behaviour, however such oscillatory behaviour was transitory, and the system did move towards a diverse, relatively stable state. The first section also considered fitness functions tied to resource gathering and structure. Results from different fitness functions found that a fitness function which solely favoured highly connected species produced a low diversity system, with persistent repeated oscillatory behaviour, possibly caused by unsuccessful mutant invasions. In the modified fitness function, which moderately favoured an average degree approaching 1, intermittent oscillatory behaviour was coupled with species turnover.

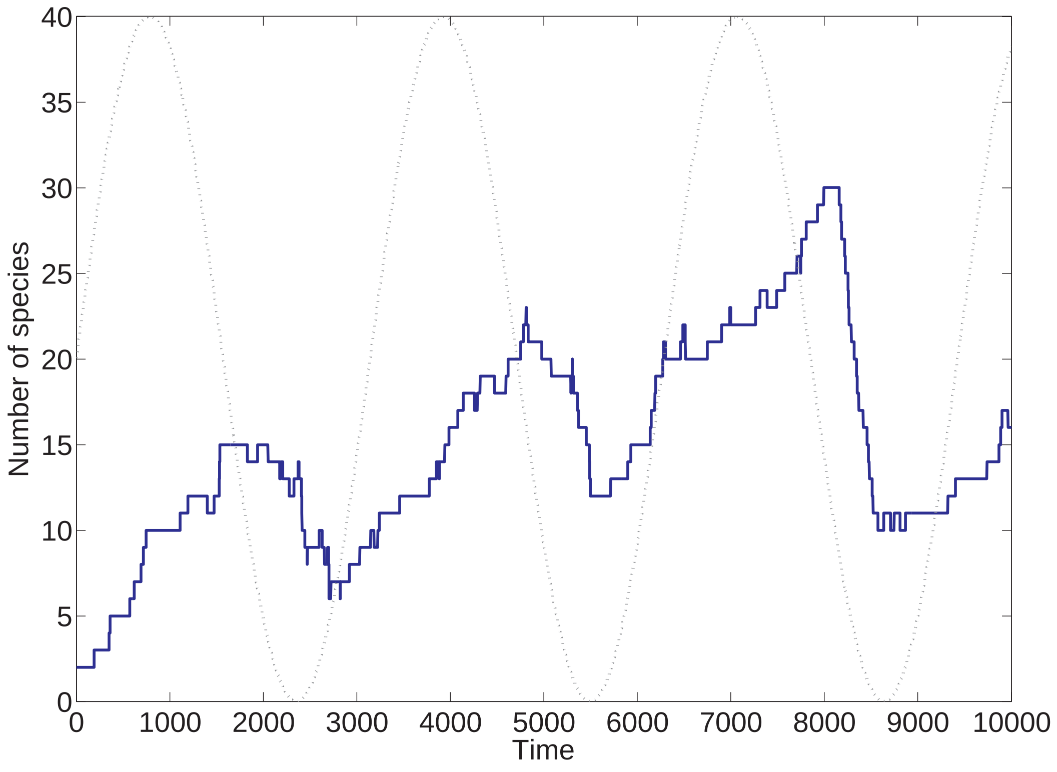


Figure 7.15: Figure showing the dynamics of a high amplitude fluctuating resource pool, fluctuating over long time scales. Fluctuation given by Equation 7.2, with $\tau = 500$, $\frac{(A_2 - A_1)}{2} = 5000$, $\frac{(A_1 + A_2)}{2} = 6000$. Different species represented by different colours, dotted line gives example resource fluctuation with same wavelength, different amplitude. Moderate edge selection bias, $p = q = \beta = 0.5$. Mutation rate $\lambda = 0.01$.

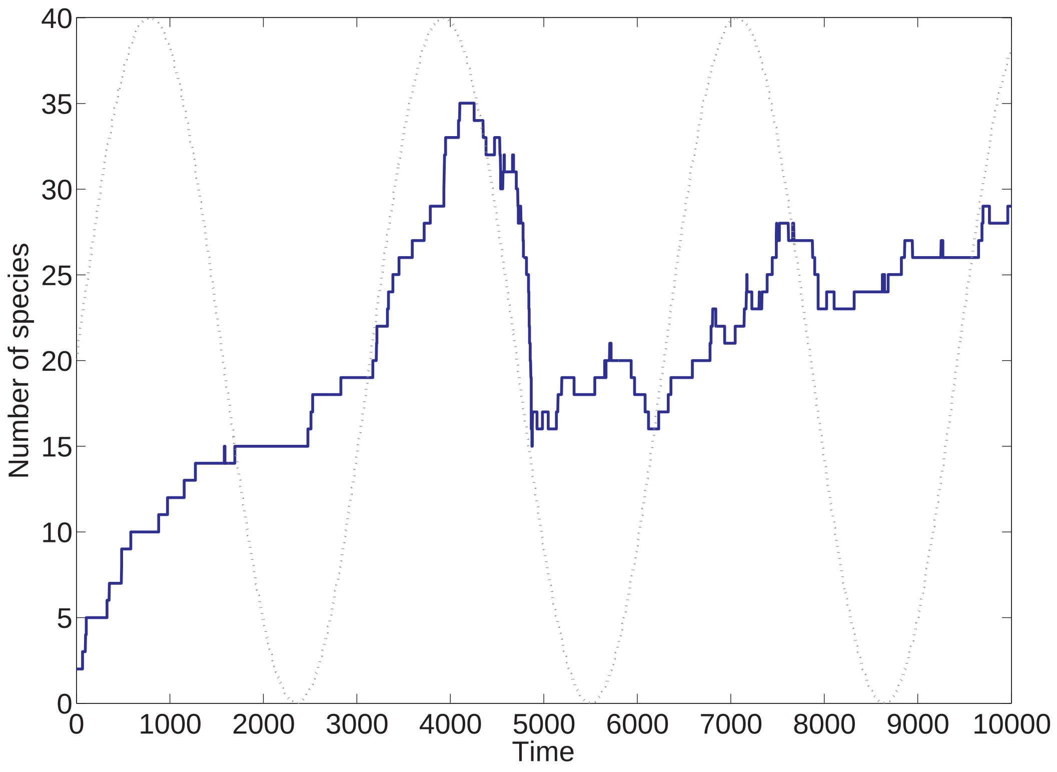


Figure 7.16: Figure showing the dynamics of a low amplitude fluctuating resource pool, fluctuating over long time scales. Fluctuation given by Equation 7.2, with $\tau = 500$, $\frac{(A_2 - A_1)}{2} = \frac{(A_1 + A_2)}{2} = 1000$. Different species represented by different colours, dotted line gives example resource fluctuation with same wavelength, different amplitude. Moderate edge selection bias, $p = q = \beta = 0.5$. Mutation rate $\lambda = 0.01$.

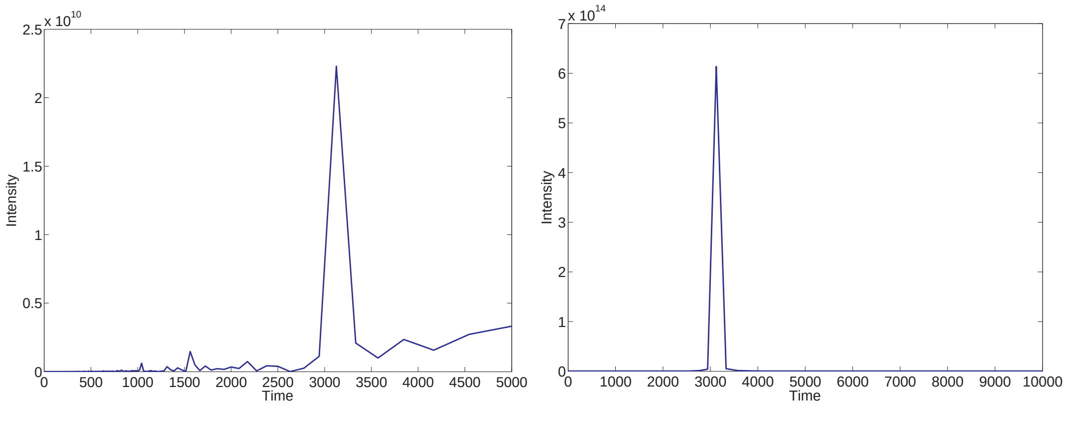


Figure 7.17: Fourier Transform power spectrum analysis of the response of the average diversity of the system to a high amplitude fluctuating input (seen in Figure 7.15), $\tau = 500$, $\lambda = 0.01$. Model data on left, averaged over 100 runs. Fourier Transform of Equation 7.2 on right.

New species became ascendant initially due to the death of lingering species, which have low number of individuals. This then triggered the deaths of multiple other species, including the previously dominant species, and a new ecosystem paradigm was developed. This apparent cascading, concurrent death of multiple species is interesting as it suggests that the dynamics of both population and resources combine to produce systems which evolve towards a precarious state. These are then susceptible to collapse following small changes in either environment. Such arguments are used in the following chapter to propose internal causes of mass extinctions throughout geological time, rather than external causes.

The second section of this chapter looked at fluctuating resources in an external environment. Building on the developments in the first section, the dependence on the characteristic wavelength of the fluctuation and the amplitude of the fluctuation was explored. As well as this, the sensitivity of the system, taken as the scale of the extinctions triggered by such fluctuations, was considered. The results from this model support our hypothesis that the impact of the environmental change is determined by the characteristic mutation rate of the system relative to the fluctuations and the amplitude of such fluctuations and that such systems can only respond to environmental changes on longer wavelengths ($\tau \geqslant$ mutation rate).

The impact of low amplitude wavelengths results in rapid extinctions of multiple species across all time scales. This is an unexpected result and we attribute this behaviour to low amplitude fluctuations enabling lingering species, which pushes the system to an unsteady state. This unsteady state eventually causes a rapid extinction of multiple species, an overturning of the species present. High amplitude changes induce overall system responses in the diversity of the system at meso- and macro-time scales.

Chapter 8

Diversity as a Driver of Mass Extinctions

8.1 Introduction

In Chapter 2, the interaction between first (population dynamics) and third tier (mass extinction) processes was described, as well as the lack of a coherent narrative explaining these interactions. In the previous chapters, this problem was approached from a first tier perspective – by studying interactions at the individual and population level to produce large scale phenomena. This chapter takes a top-down approach, examining mass extinctions and constructing a null hypothesis for their cause, based on recent developments in network theory.

In the study of mass extinctions, there is a prevailing view that such events are caused by extremely large environmental perturbations (*e.g.*White and Saunders [2005]; Courtillot et al. [1996]; Hallam and Wignall [1997]). These rare events have been encapsulated in the popular imagination by the Chixculub crater, which seeks to explain the K-Pg mass extinction (*e.g.* Schulte et al. [2010]; Hildebrand et al. [1991]; Sharpton et al. [1992]). Researchers have attempted to link various other large impact sites to mass extinctions, with varying degrees of success (*e.g.* Claeys et al. [1992]; Sandberg et al. [1988]; Hodych and Dunning [1992]; Shoemaker and Wolfe [1986]).

In this chapter we develop a null hypothesis for mass extinctions – namely that mass extinctions can be explained by a series of random, but crucially minor, events related to the background rate of extinction. We test this hypothesis using a simple Poisson process model for background extinction implemented on a scale-free network.

8.2 Causes of mass extinctions

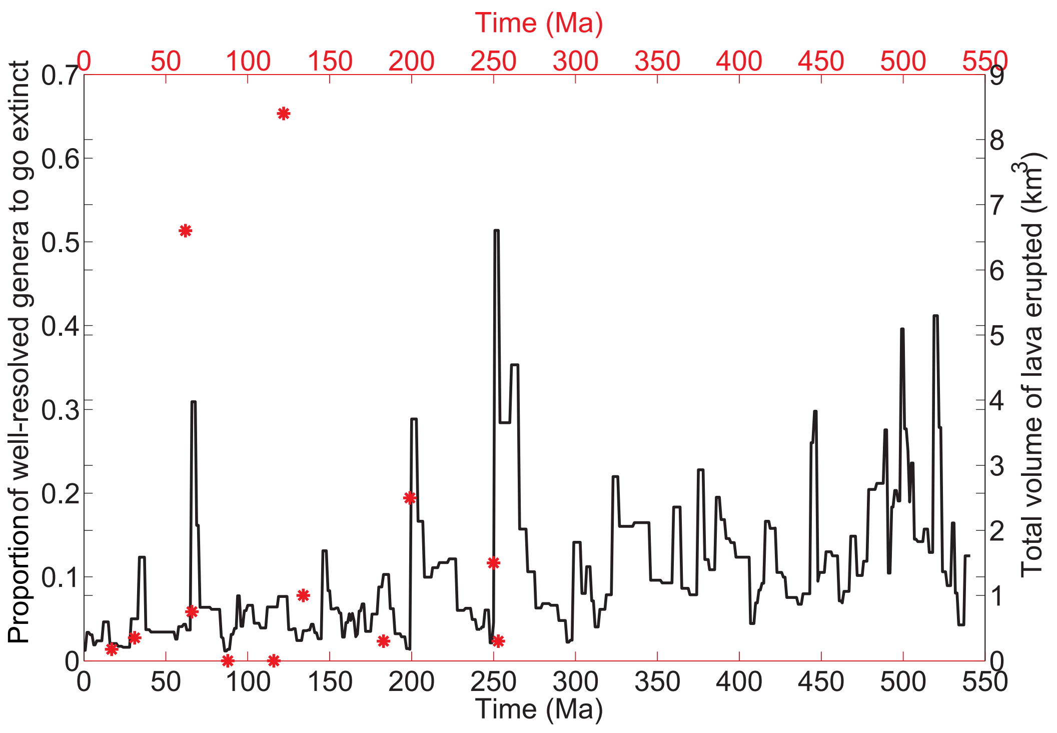
Two major proposed causes of mass extinctions are large igneous provinces (LIPs), which are episodes of mass volcanism, and large bolide impacts. Known events of both are plotted in Figure 8.1 against the extinction profile of genera from Rohde and Muller [2005].

We would expect large bolide impacts to cause large amplitude fluctuations (that is, a high rate of change of resources), whereas large igneous provinces may (depending on the volume of lava erupted over time) cause low or high amplitude fluctuations. From the results in Chapter 7 we would predict that long term eruptions would cause bigger, more rapid extinctions than meteorites.

From Figure 8.1 we can see that some large igneous provinces and some large bolide impacts are correlated with mass extinctions (for the former, the Siberian and Emeishan traps associated with the Permo-Triassic mass extinction event ($\approx$ 250Ma), and for the latter the Chixculub event and the K-Pg mass extinction event ($\approx$ 65Ma)). However, there appears to be no obvious connection between size of event and severity of extinction (nor even size of event with occurrence of extinction).

There have been vocal critics of the link between impacts and mass extinctions (see for example Keller [2005]). Similarly, there appears to be no strong statistical evidence for the correlation of mass volcanism and bolide impacts with mass extinctions [White and Saunders, 2005]. White and Saunders [2005] proposes that a combination of major catastrophes would have the appropriate frequency and magnitude to be associated with major mass extinctions. If large external environmental perturbations aren't correlated with mass extinctions, an alternative cause must be proposed. In the following sections, we develop a model to test whether it is possible for mass extinctions to be caused solely by the background rate of extinction.

If our model shows that the background rate of extinction must be implausibly high to produce mass extinctions on a time scale commensurate with that in the fossil record [Raup, 1991] then this would be evidence that large exogenous events are required for mass extinctions. If on the other hand our model shows that background extinction rates can be enough to explain the rate of mass extinctions, this would be evidence that large exogenous events are not required to produce mass extinctions. They may, of course, be contributing factors (as, for example, large exogenous events may increase the background rate of extinction).



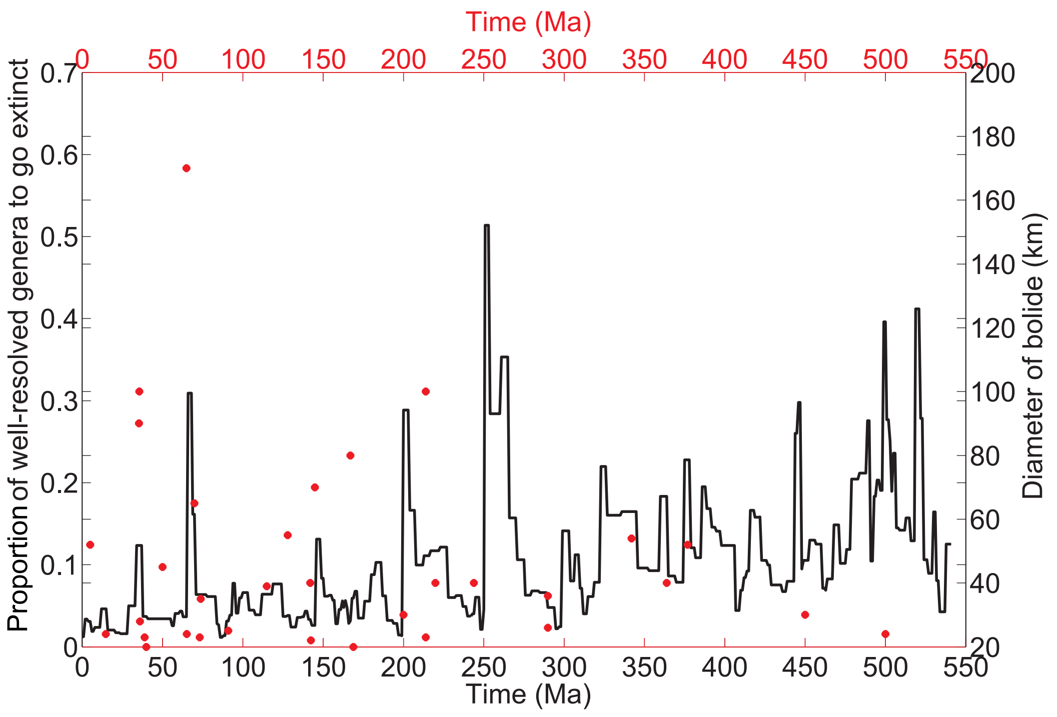


Figure 8.1: Figure showing proportion of well-resolved genera to go extinct (using supplementary data from Rohde and Muller [2005]) plotted against occurrences of large igneous provinces (upper) and meteorites (lower).

8.3 Introduction to network theory

Network vulnerabilities have been known for some time, beginning with work on Boolean networks Kauffman and Smith [1986], and recently applied to the Internet and ecological scenarios, amongst others [Solé et al., 2003]. The application to ecology is of primary relevance to this chapter; in particular, recent work by Dunne et al. [2002a] showed that large numbers of primary and secondary extinctions could occur through the removal of a small number of "keystone" nodes.

Criticism of this research direction has taken the form that ecological networks are dynamic, not stationary, and that real world networks seldom conform to strict statistical degree distributions which facilitate such cascading behaviour Dunne et al. [2002a]. These concerns are addressed by considering a network that despite being dynamic on short time scales, approaches a particular degree distribution over long (evolutionary) time scales. This separation of time scales follows both the modelling choice in Chapter 4, and the arguments presented in Metz [2011].

The degree distributions which are typically considered are of the form of a power law (*e.g.* [Dunne et al., 2002b; Montoya and Solé, 2002]). In this case, the fraction $P(k)$ of nodes in the network having k connections to other nodes goes (for large values of k) as

$$P(k) \approx ck^{-\gamma}, \tag{8.1}$$

where c is a normalisation constant and γ is a parameter which typically is in the range $2 < \gamma < 3$ for abiological networks [Montoya et al., 2006]. For biological networks, values of around $0.94 \geqslant \gamma \geqslant 1.13$ are more common [Montoya and Solé, 2002].

8.3.1 Degree distributions and expectation

The fragility of networks is tested using node removal – that is, how robust they are to fragmenting completely when nodes are removed. Scale free networks are particularly susceptible to having their highly connected nodes removed. It is clear that h, the number of highly connected nodes in a network, is a function of $P(k)$, the fraction of nodes having k connections in the network. This requires a definition of what a "highly connected node" is, in terms of the minimum number of connections needed for a node to be classed as such.

As a first approximation, assume that a node is highly connected if it has a degree that is strictly greater than one standard deviation away from the average degree distribution of the network. This is equivalent to saying that the top 16% of nodes ordered by degree in a normal degree distribution network are highly connected. This approaches the results in Dunne et al. [2002a], that found that the removal of 20% of the most connected species resulted in secondary extinctions encompassing $60 - 100\%$ of species.

8.4 Poisson process

8.4.1 Time scales

As in Chapter 4, where mutation was modelled as a Poisson process, here node removal is also modelled as a Poisson process. For such an approach, define T as the time scale over which d highly connected nodes must be removed. T can be thought of as the relaxation time of the ecosystem; that is, the time during which species eliminated have not yet been replaced. The proportion of highly connected nodes h in a system is calculated in the same way as above, which gives the probability of a highly connected node being removed at random from a network of N nodes. Given a background extinction rate of τ, the expected number of highly connected nodes being removed within a given time frame is

$$\lambda = \frac{h}{N}\tau T, \tag{8.2}$$

where λ is the Poisson process parameter. Note that this assumes that each species has the same probability of going extinct. Assuming that evolution is not an omniscient process, then species can only adapt to the environments they have experienced (and to those only poorly, given the contingent nature of evolution due to constraint). Then if the environment is perturbed in a way that the species cannot adapt to – for example, if the environmental change has a different characteristic time frame to the adaptation of the species, then it is likely that that species will go extinct. It is arguable that species that are highly connected – meaning they are the major source of energy for many species – are particularly resilient to many environmental perturbations. Thus, it may be that a more severe environmental perturbation may be required, for example a trigger such as massive volcanism. However, that is only required in the case where the null model fails.

The cumulative distribution function for the model is

$$P(X < d) = e^{-\lambda} \sum_{i=0}^{d-1} \frac{\lambda^i}{i!} . \tag{8.3}$$

$1-P$ therefore gives the probability over a given time that at least d highly connected nodes will be removed.

8.5 Extinction rates

Raup [1981] estimated the background extinction rate for species as around one species every $3\frac{1}{3}$ years. However, this estimate was based on the number of extant species being around 1.5 million. The most recent estimates of biodiversity suggest that there are closer to 8.7 million ($\pm$1.3 million) eukaryotic species [Mora et al., 2011]. Following the method in Raup [1981], this suggests that the background extinction rate for species is closer to 1.74 ± 0.26 species per year. Note that background extinction rates assume a number of conditions, and are criticised by Boucot [2006].

From the background extinction rates the expected waiting time until a cascade propagates throughout the network can be calculated, if node removal is equated with extinction. Using the data from Dunne et al. [2002a], let $d = 0.25N$, since food webs display threshold behaviour when there is the removal of $20 - 30\%$ of primary species (preferentially removing highly connected nodes). h now needs to be phrased in terms of a fraction of N. Note that for the network degree definitions in the introduction, the value of k is required, which is the objective definition of what a highly connected node is.

8.5.1 Scale free networks

In the case where the network being modelled is scale free $h \approx Nc \sum\limits_{k>C} k^{-\gamma}$, where C is the minimum degree required for a node to be deemed highly connected. Combining Equations 8.2 and 8.1

$$\lambda \approx \tau T c \sum_{k \geq C}^{N} k^{-\gamma} \tag{8.4}$$

although k is still dependent on N. The normalisation constant c is equal to $\frac{1}{\sum\limits_{k=1}^{N} k^{-\gamma}}$ resulting in

$$\lambda \approx \tau T \frac{\sum\limits_{k \geq C}^{N} k^{-\gamma}}{\sum\limits_{k=1}^{N} k^{-\gamma}}. \tag{8.5}$$

To simplify the Poisson process, let β be a parameter of the model, such that

$$\beta = \frac{h}{N} = \frac{\sum\limits_{k \geq C}^{N} k^{-\gamma}}{\sum\limits_{k=0}^{N} k^{-\gamma}} \tag{8.6}$$

so

$$\lambda = \tau T \beta, \tag{8.7}$$

where β is the fraction of highly connected nodes in the graph, $\frac{h}{N}$. Note this still leaves two unknowns in the equation: C and γ. However, γ can be extracted from the ecological network literature – here a value of 1.085 is used, after Sole and Montoya [2001]. However, there is an argument that species do not follow a power law, but instead have a truncated distribution, as there is a theoretical maximum number of links per species [Montoya et al., 2006].

8.6 Results

8.6.1 Relaxation time and extinction rate dependence

Estimating T from the literature, using relaxation times, produces in the range of 10^3 to 10^4 years for ecologically based studies [Diamond, 1972]. Although criticism has been made of empirical studies – particularly their inability to consider geologically relevant time scales [Kuussaari et al., 2009]. In contrast, Kirchner and Weil [2000] finds the strongest Fourier Transform signal for origination rates to be 10^7 years after the mass extinction event, based on the fossil record. This seems implausibly high for small extinctions, particularly as the kill curve established by Raup [1991] has them appearing at a higher frequency. We will therefore use 10^7 as an upper bound for relaxation times for large mass extinctions (those killing more than $\approx 60\%$ of species.

We initially consider the case for a small system, $N = 30000$. Using the parameter values discussed above, we find that the mean waiting time until a mass extinction is dependent on C and τ. As expected, an increase in the extinction rate, τ, means a lower threshold value of C is required to achieve a mass extinction. This relationship is seen in Figure 8.2. If C is small enough then a mass extinction is guaranteed - which supports the results in Dunne et al. [2002a] for a critical threshold of highly connected nodes. Increasing C leads to an increase in the mean waiting time until a mass extinction. We would expect this from the model due to the

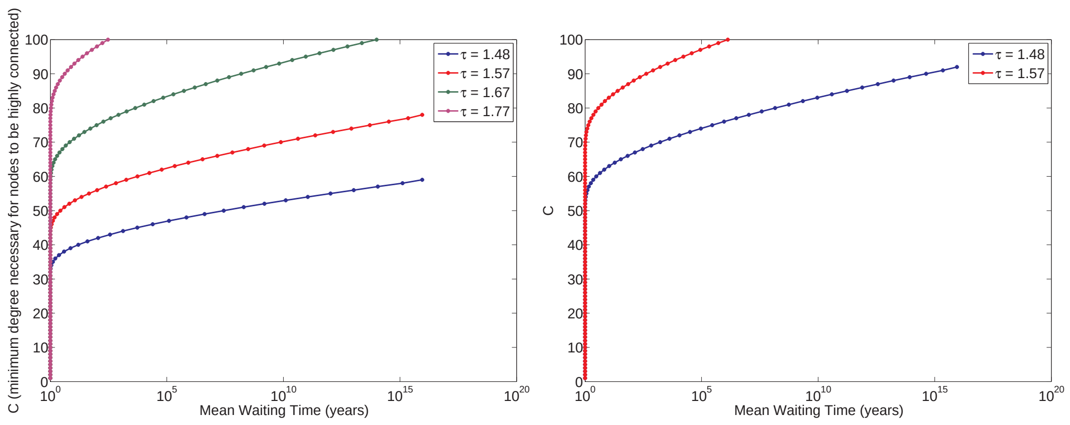


Figure 8.2: Figure showing change in mean waiting time of a mass extinction against the C – the minimum degree needed to be a highly connected node. Figure on left showing relaxation time $T = 10^4$. Figure on right showing relaxation time $T = 1.1 \times 10^4$. In both figures $N = 3 \times 10^4$.

scale-free degree distribution which means high degree nodes are scarce, resulting in a higher rate of extinction necessary to remove a particular proportion of highly connected nodes.

The overall relationship between these variables suggests that there are extinction rates which will always result in mass extinctions, regardless of the value of C. For $T = 10^4$, this occurs for $\tau \geqslant 1.96$, and for $T = 1.1 \times 10^4$, this occurs for $\tau \geqslant 1.74$.

Scaling the system size by a factor of $\frac{4}{3}$, so that $N = 10^5$, requires a doubling of the relaxation time of the system in order to see mass extinctions over different waiting times. It is also the case that larger system sizes require a lower value of C for a given mean waiting time, as seen in Figure 8.3. However, it's also the case that, as the system size increases, the probability of mass extinction becomes more and more sensitive and approaches a binary system where there is either definite mass extinction independent of C or no mass extinction. This is most clear in Figure 8.4 which shows a system size of $N = 10^6$ (for reference there are currently thought to be 8.7×10^6 extant species). This is an unexpected result, and may explain why some major environmental perturbations are associated with mass extinctions and some are not.

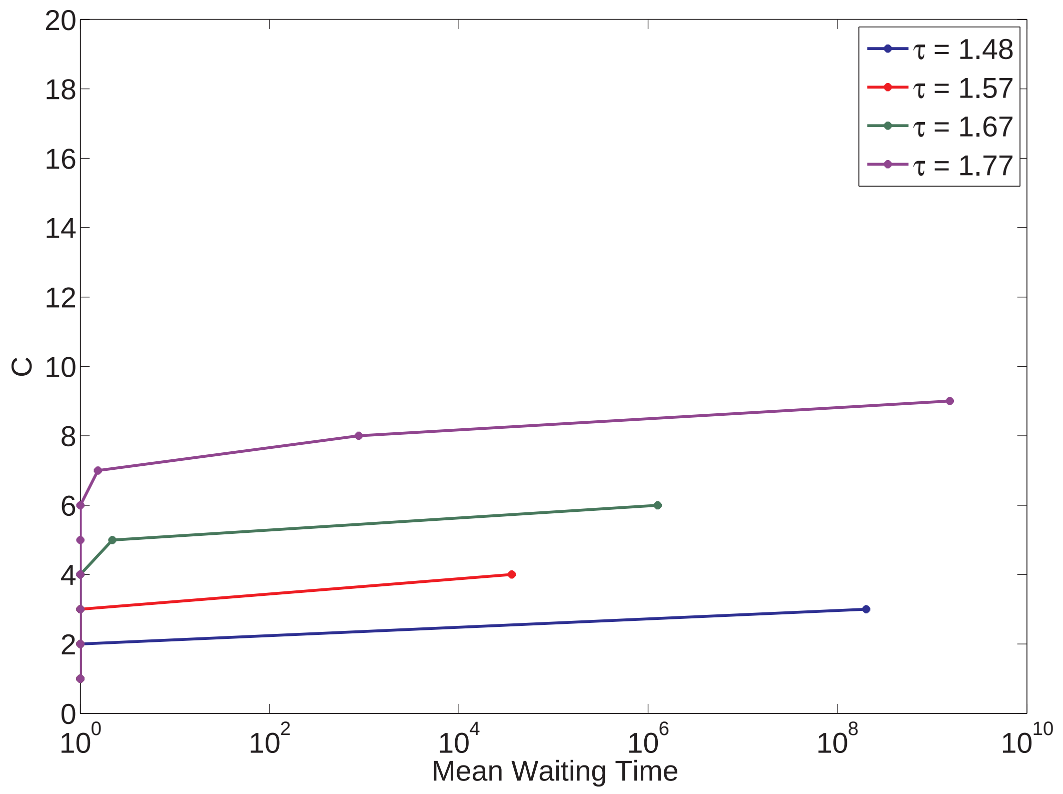


Figure 8.3: Figure showing change in mean waiting time of a mass extinction against the C – the minimum degree needed to be a highly connected node. Relaxation time $T = 2 \times 10^4$, system size $N = 10^5$.

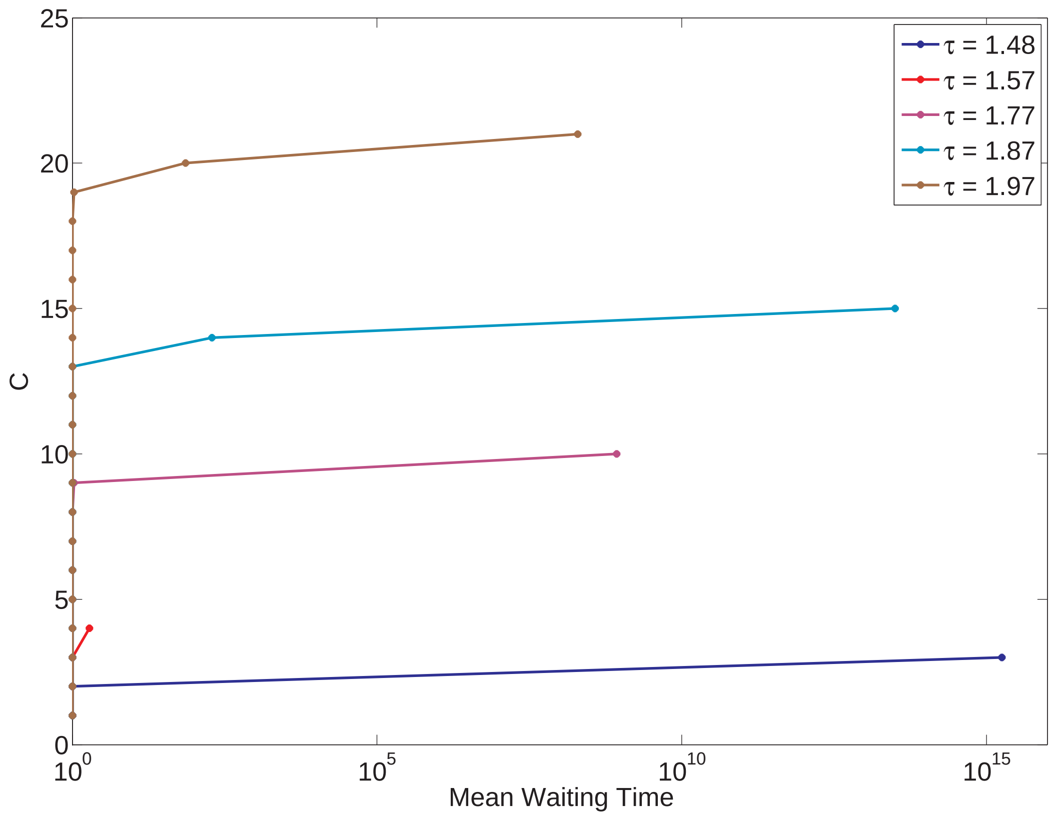


Figure 8.4: Figure showing change in mean waiting time of a mass extinction against the C – the minimum degree needed to be a highly connected node. Relaxation time $T = 2 \times 10^5$, system size $N = 10^6$.

8.7 Conclusion

In Gould [1985], the dissociation between first tier evolutionary processes and third tier macro-evolutionary processes was discussed. Mass extinctions were deemed geologically to be major calamities, caused by extreme environmental upheaval, and punctuating the fossil record. But as processes in their own right there is relatively little understanding of the causal processes behind such catastrophes. Similarly, there have been no universal hypotheses for why some major environmental perturbations cause mass extinctions, and why some do not.

In this chapter, a null hypothesis model of mass extinctions is proposed, derived from second tier processes. Assuming a uniform extinction rate a Poisson process model is sufficient to cause global mass extinctions using biologically realistic parameters. Our model shows that the background rate of extinction can be enough to explain the rate of mass extinctions, and this is some evidence that large exogenous events are not required to produce mass extinctions (although they may be contributing factors).

We found a crucial dependence on the exact value of the background extinction rate, τ. Estimated empirical values of τ could put us into the regime where background extinction is sufficient to cause mass extinction or could put us into the regime where exogenous events are required. Although we were not able to show conclusively that the standard view requiring exogenous events is incorrect, we have at least demonstrated that the alternative deserves serious consideration as a possibility. This is reinforced by the lack of correlation between historical mass extinctions and known large exogenous environmental perturbations.

The model suggests that, as system size increases, the likelihood of a mass extinction increases. This agrees with the models discussed in Chapter 7 where systems with high diversity and low amplitude environmental fluctuations were prone to rapid turnovers of species.

This raises questions about why some large environmental perturbations are associated with mass extinctions and others are not. One possibility, according to our model, is that there may be ecosystems which are not large enough to be affected (that is, the environmental perturbation does not cause enough additional extinctions to definitely cause a mass extinction, but might instead lower the mean waiting time until another extinction). This would support the hypothesis proposed in Butterfield [2007], whereby the mode and tempo of macro-evolutionary processes is different in the Pre-Cambrian, and would suggest a causal link between diversity, structure of ecosystems, and mass extinctions.

Chapter 9

Conclusions

There is grandeur in this view of life, with its several powers, having been originally breathed into a few forms or into one; and that, whilst this planet has gone cycling on according to the fixed law of gravity, from so simple a beginning endless forms most beautiful and most wonderful have been, and are being, evolved.
– Charles Darwin, On the Origin of Species,

In this thesis we set out to show that the third tier of evolution could be derived from processes acting on the first and second tiers. We approached this paradox both theoretically, in Chapter 3, analytically in Chapters 6 and 8, and by mathematical modelling in Chapters 4, 5, and 7. We proposed that first tier interactions control third tier processes, and that the trends seen in the third tier are, in fact, an expected consequence of this control.

9.1 Chapter 3

Chapter 3 examined the degree of constraint in evolutionary systems, and its effect on the outcomes, following Leimar [2002]. We proposed that constraint could be a first tier process that is a driver of evolutionary trends.

We also considered the paradox of evolvability and proposed that if we consider biological evolutionary processes as a non-equilibrium physical system then, after Wissner-Gross and Freer [2013], this seems to negate the so-called paradox of evolvability, *i.e.* that evolvability should not be a persistent feature of organisms as selection is short-sighted in its maximisation of fitness, since non-equilibrium physical systems can delay immediate maximisation to instead maximise entropy production for between now and a future time.

The chapter concluded with a discussion on a possible framework for evolution, combining the observations made from the model comparison analysis with the consideration of constraint and evolvability.

9.2 Chapter 5

In this chapter we approach the gap between first and second tiers from the first tier, by asking what the effect of different fitness landscapes is on the overall diversity of the system. Using the model described in Chapter 4, we find that it exhibits both rapid evolutionary innovation (characteristic of punctuated equilibrium) but also gradualistic change. This is enhanced by the ruggedness of the fitness landscape, and by the presence of a cost to species' structure in the single peak fitness landscape. However, we do not find that the ruggedness or lack of ruggedness induces a particular mode of speciation.

We also find that, in the single peak fitness landscape, increasing the number of degrees of freedom in the species under selection increased also increased the diversity of the system. This agrees with our initial prediction, and we propose that this is because with a higher combination of traits, more species can coexist at arbitrarily close points in the landscape. This slows the rate of competition between species, resulting in an artificial inflation of the diversity of the system.

However, in the case of a rugged fitness landscape the additional degree of freedom did not have a discernible effect on the diversity of the system. This suggests that if the fitness landscape exhibits high degrees of ruggedness, the resultant diversity is high, due to interactions between traits. However, the addition of a cost to the species' structure made no discernible difference to the diversity of the system. This disagrees with our hypothesis and suggests that either there either there is a maximum to the amount of diversity supported by additional degrees of freedom (with diminishing returns for each additional degree of freedom added) or that the mutations result in jumps to other local peaks, rather than coexistence around a local peak.

Further work could explore whether a particular degree of ruggedness that correlates with second tier trends, or if some other parameter could induce particular second tier processes. Additionally, we could examine the precise relationship between degrees of freedom and induced diversity.

9.3 Chapter 6

In Chapter 5 we approached the paradox of the first tier (as described in Chapter 1) by examining the connection between the first and second tiers (population dynamics and evolutionary trends respectively). In this chapter we considered the second tier and, specifically, the role of meta-regulation in constraining such evolutionary trends. We proposed that meta-regulation is one of the more important regulators in evolutionary trends, being responsible for constraint as well as aiding evolvability and altering the tempo of evolutionary processes (as argued in Chapter 3). In Chapter 6 we used the results from Chapter 5 to quantify the variation seen in the average outgoing degree to model meta-regulation rules.

Using the duplication-divergence framework to analyse the model described in Chapter 4 we found that, for sparse or small graphs, the average outgoing degree is maximised when $p = \beta = 1$. However, the growth rate is also maximised when $\beta \neq 1$. Under these conditions a mixed strategy is produced - where small dense sub-graphs are sparsely connected to each other. These are analogous to the genetic kernels and modular sub-structures described in Erwin and Davidson [2009], and suggests a system with three simple mutation options and moderate selection for the average outgoing degree is sufficient to develop such structures.

We also asked how this trade-off between increasing object and hierarchical complexity was bounded. The model does not have a steady state for average outgoing degree, suggesting that either hierarchical or object complexity (or both) is unbounded. A steady state is reached in individual runs of the model, however, suggesting that selection pressure can limit the average outgoing degree. Therefore, although theoretically there is no limit, selection pressure is sufficient to impose limits.

It is important also to note that for individual species the order of the mutation steps is important. In particular, for a given selection threshold there will be an optimum point at which meta-regulation develops. However, once this initial meta-regulation is developed, increasing object complexity maximises the average outgoing degree more than increasing hierarchical complexity. This supports observations regarding the nature of meta-regulation [Maynard Smith and Szathmáry, 1997] - that is, that there are periods of rapid innovation followed by periods of quiescence. This would support a hypothesis that linked the first and second tiers via meta-regulation.

Further work could derive the precise recurrence relation for the meta-regulation model, and explore whether models with (for example) edge point mutations show a steady state.

9.4 Chapter 7

Chapter 7 continued the theme of uniting the first, second and third tiers of evolution by exploring the gap between first, second and third tiers. We asked how environmental variation over different time scales affects diversity. In the previous chapter we looked at intrinsic ways hierarchical complexity could be evolved, whereas in this chapter we looked at fluctuations in external resources and their impact on the diversity of organisms evolved within the model described in Chaper 4. We asked the question: what role do external environments play in third tier processes?

The results from model suggest that, as expected, the impact of the environmental change is determined by the characteristic mutation rate of the system relative to the fluctuations and the amplitude of such fluctuations. Thus, systems can only respond to environmental changes on longer wavelengths ($\tau \geqslant$ mutation rate). Furthermore, we find that the impact of low amplitude wavelengths results in rapid extinctions of multiple species across all time scales. High amplitude changes induce overall system responses in the diversity of the system at meso- and macro-time scales. We attribute this behaviour to low amplitude fluctuations enabling lingering species, which pushes the system to an unsteady state. This unsteady state eventually causes a rapid extinction of multiple species, an overturning of the species present. This apparent cascading, concurrent death of multiple species is interesting as it suggests that the dynamics of both population and resources combine to produce systems which evolve towards a precarious state.

9.5 Chapter 8

In Chapter 8 a null hypothesis for mass extinctions was developed – namely that mass extinctions can be explained by a series of random, unfortunate, but crucially minor, events related to the background rate of extinction. The background rate of extinction is directly driven by competition between species - a first tier process. Mass extinctions are a clear third tier process.

We showed that this model is sufficient to create mass extinctions, using biologically realistic parameters, and thus large exogenous events are not required. The model depends strongly on the number of connections needed to be a highly connected node, and assumes a scale-free network, however we propose that this links processes on the second tier (diversification and species interaction) to those on the third tier (mass extinctions).

Further work could look at different degree distributions in networks to check whether the biologically realistic parameters still held.

9.6 Closing Remarks

In this thesis we developed a model to explore how third tier evolutionary processes could derive from first and second tier processes.

Our model showed firstly that by increasing the number of degrees of freedom, or by increasing the ruggedness of the fitness landscape, higher system diversity was evolved. We would like to explore further how closely ruggedness is tied to particular macro-ecological trends, and whether modularity is causal by increasing the degrees of freedom in the system.

We showed that the analytical version of the model had no steady state behaviour, suggesting no limit to hierarchical complexity, and our model supported a mixed strategy for evolving hierarchical networks whereby object complexity was increased followed by hierarchical complexity. It would be interesting to know the precise tipping points between increasing object complexity and increasing hierarchical complexity.

When examining characteristic time scales, our model suggests that systems can only adapt to environmental changes that are less frequent than the mutation rate. We also showed that low amplitude environmental changes caused rapid extinctions of multiple species on all time scales. This ties third tier behaviour to second tier process.

Finally, we proposed a model for explaining mass extinctions using a background rate of extinction. This also suggests that third tier processes can be explained by species' interactions.

Appendix A

Model Code

```
#include <boost/property_map/property_map.hpp>
#include <boost/graph/adjacency_list.hpp>
#include <boost/random/normal_distribution.hpp>
#include <boost/random/mersenne_twister.hpp>
#include <boost/random/variate_generator.hpp>
#include <boost/random/uniform_01.hpp>
#include <boost/graph/graph_traits.hpp>
#include <boost/graph/properties.hpp>
#include <boost/graph/adjacency_iterator.hpp>
#include <boost/graph/graphviz.hpp>
#include <boost/graph/copy.hpp>
#include <boost/graph/clustering_coefficient.hpp>
#include <boost/ptr_container/ptr_list.hpp>
#include <boost/ptr_container/ptr_vector.hpp>
#include <boost/algorithm/string.hpp>
#include <boost/tokenizer.hpp>
#include <numeric>
#include <iostream>
#include <math.h>
#include <algorithm>
#include <boost/graph/exterior_property.hpp>
#include <boost/math/constants/constants.hpp>

using namespace std;
using namespace boost;

/* Program definitions */
#define multipeak 0 // Selection on multiple peaks
#define peakselect 0 // Selection to particular fitness value
#define levelselect 0 // Selection based on levels
#define breakup 1 // No dissociation of graphs on death.
#define fluctuations 0 // Fluctuating environment according to a sine wave
#define ages 0 // Age structured populations
#define totrescol 1 // Number of different resources. Requires genotype == 0 && resources == 1.
#define resources 0 // Environment present = 1; no environment = 0
#define norm 0 // Normalisation on = 1; off = 0;
#define mult 0 // Multiplication = 1; summation = 0; NO NORMALISATION if == 1
#define levelcost 0 // External environment of levels on = 1; off = 0; Requires maintenance == 1;
#define dynamics 1 // Population dynamics = 1; genetic algorithm = 0;
#define maintenance 1 // Maintenance on = 1; off = 0;
//#define rmain 1 // Restricted == 1 or unrestricted == 0 maintenance
#define sysexp 0 // System size expanding = 1; fixed size = 0;
#define pnew 0.5 // Probability of creating a new node;
#define qcopy 0.5 // Probability of copying the subgraph;
#define beta 0.5 // Probability of creating an edge between two vertices
#define tsteps 10000 // Number of time steps
#define runs 1 // Number of runs
#define sysmax 10000 // Absolute maximum system size
#define metscale 0.66 // Metabolic scaling coefficient. Should vary between 2/3 and 3/4
#define genotype 0 // Genotype on == 1; off == 0
```

```
#define deathrate 0.15 // 1 - probability of individual dying
#define analytics 0 // 1 sampling for analytics (only looks at progressive mutation)
#define comparison 0 // Used for obtaining a mean rate of mutation to compare GA with PD
#define constraint 1 // Constrain so that mutation probability is node dependent. Doesn't work with p=1 (obviously)

/* Defining the graph structures to be used in the species definition. */

struct Gnodes {
int level; // Node level
double ability;
double maintain;
int colour;
};
// Defines the vertex property Nodes, which have a level assigned.

struct Level {
int maxlevel;
int ancestor;
int identity;
int copyno;
double abtotal;
double maintotal;
ptr_vector<int> age;
int tstart;
double fittotal;
double normfit;
};
// Defines graph level properties

typedef boost::adjacency_list<boost::vecS, boost::vecS, boost::bidirectionalS,
Gnodes, boost::no_property, Level, boost::listS> Bodyplan; // Adjacency list of the hierarchical structure that is the species'
//blueprint, using Nodes and Speciesfit structure.

typedef boost::graph_traits<Bodyplan>::out_edge_iterator outedgeiterator;

typedef boost::graph_traits<Bodyplan>::vertex_descriptor vertex_t;

typedef boost::graph_traits<Bodyplan>::edge_descriptor edge_t;

typedef boost::adjacency_iterator_generator<Bodyplan, vertex_t, outedgeiterator>::type adjacency_iterator;

typedef boost::adjacency_list<boost::vecS, boost::vecS, boost::directedS,
boost::no_property, boost::no_property, boost::no_property, boost::listS> Branchgraph;

typedef ptr_list<Bodyplan> Population;
typedef ptr_list<string> Genopop;

typedef ptr_list<ptr_vector<double> > doublelists;

typedef find_iterator<string::iterator> string_find_iterator;

typedef tokenizer<boost::char_separator<char> > tokeniser;

typedef vector<string> split_vector_type;

typedef boost::mt19937 Engine;
typedef boost::normal_distribution<> Normal;

typedef exterior_vertex_property<Bodyplan, double> ClusteringProperty;
typedef ClusteringProperty::container_type ClusteringContainer;
typedef ClusteringProperty::map_type ClusteringMap;

double normal_gen(Engine &eng, Normal dist) {
return dist(eng);
}

double rn(void) {
boost::mt19937 rng(time(0));
static boost::uniform_01<boost::mt19937> zeroone(rng);
return zeroone();
}
```

```
struct my_node_writer {
my_node_writer(Bodyplan& species_) :
species(species_) {
}
;
template<class Vertex>
void operator()(ostream& out, Vertex v) {
out << " [label=\"" << species[v].level << "\"]" << std::endl;
}
;
Bodyplan species;
};

void degcalc(Population &pop, ptr_vector<double> &deg, ptr_vector<double> &degdis,
int y, ptr_vector<double> &counter) {

Population::iterator itpop;
double avdegspec = 0;
double avdeg = 0;

for (itpop = pop.begin(); itpop != pop.end(); itpop++) {
avdegspec = 0;
avdegspec = ((double)num_edges(*itpop) / (double)num_vertices(*itpop));

avdeg += avdegspec;
}

avdeg /= (double)pop.size();
deg[y] += avdeg;
counter[y] += 1;
}

int levelcount(Population &pop) {
Population::iterator i;
for (i = pop.begin(); i != pop.end(); i++) {
for (unsigned int b = 1; b < num_vertices(*i); b++) {
if ((*i)[b].level > (*i)[graph_bundle].maxlevel) {
(*i)[graph_bundle].maxlevel = (*i)[b].level;
}
}

}

int maxmaxlevel = 0;

for (i = pop.begin(); i != pop.end(); i++) {
if ((*i)[graph_bundle].maxlevel > maxmaxlevel) {
maxmaxlevel = (*i)[graph_bundle].maxlevel;
}
}

return maxmaxlevel;
}
;

int levelcount(Genopop &pop) {
Genopop::iterator i;

int maxmaxlevel = 0;

for (i = pop.begin(); i != pop.end(); i++) {
int speclevel = 0;
for (string_find_iterator It = make_find_iterator(*i,
first_finder("[", is_iequal())); It != string_find_iterator();
++It) {
speclevel += 1;
}
speclevel -= 1; // Because level counting starts at 0

if (speclevel > maxmaxlevel) {
maxmaxlevel = speclevel;
}
```

```
}

return maxmaxlevel;
}
;

int nodelevelcount(Genopop &pop, int parentnode) {
char_separator<char> levsep("[]");
tokeniser levtok(pop.back(), levsep);

ptr_vector<int> levelnums;
int levelcounter = 0;
int nodecounter = 0;
string level;

for (tokeniser::iterator levit = levtok.begin(); levit != levtok.end();) {
levelnums.push_back(new int (1));
levelnums.back() = nodecounter;
level = *levit;
for (string_find_iterator It = make_find_iterator(level,
first_finder("{", is_iequal())); It != string_find_iterator();
++It) {
levelnums.back() += 1;
nodecounter++;
}

if ((parentnode) < levelnums.back()) {
levit = levtok.end();
} else {
levelcounter++;
levit++;
}

}

return levelcounter + 1;
}

void popgrowth(Population &pop, int &systemsize, ptr_vector<int> &survivorlength, int y) {
if (peakselect == 0) {
Population::iterator its;
double popsum = 0;

//cout << pop.size() << endl;

if (sysexp == 1) {
systemsize = (((double) pop.size()) / ((double)1 + (double) pop.size()))
* (int) sysmax;
}

double totsumcheck = 0;

if (maintenance == 1) {

for (its = pop.begin(); its != pop.end(); ++its) {
popsum += floor(
(((*its)[graph_bundle].abtotal
/ (*its)[graph_bundle].maintotal)
* (double)((*its)[graph_bundle].copyno)));
//cout << "Number of individuals " << (*its)[graph_bundle].copyno << endl;
//cout << "Ability score " << (*its)[graph_bundle].abtotal << endl;
}

for (its = pop.begin(); its != pop.end(); ++its) {
(*its)[graph_bundle].copyno = floor(
(((((*its)[graph_bundle].abtotal)
/ (*its)[graph_bundle].maintotal)
* (double)((*its)[graph_bundle].copyno)) / popsum)
* systemsize);
//cout << "New no. of individuals " << (*its)[graph_bundle].copyno
//<< endl;
totsumcheck += (*its)[graph_bundle].copyno;
```

```
}

} else {
for (its = pop.begin(); its != pop.end(); ++its) {
popsum += floor(
(((*its)[graph_bundle].abtotal)
* ((*its)[graph_bundle].copyno)));
//cout << "Number of individuals " << (*its)[graph_bundle].copyno << endl;
//cout << "Ability score " << (*its)[graph_bundle].abtotal << endl;
}

for (its = pop.begin(); its != pop.end(); ++its) {
(*its)[graph_bundle].copyno = floor(
((((*its)[graph_bundle].abtotal)
* (double)((*its)[graph_bundle].copyno)) / popsum)
* systemsize);
//cout << "New no. of individuals " << (*its)[graph_bundle].copyno
//<< endl;
totsumcheck += (*its)[graph_bundle].copyno;
}
}

unsigned int z = pop.size();

for (unsigned int yy = 0; yy < z; yy++) {
its = pop.begin();
advance(its, yy);

if ((*its)[graph_bundle].copyno < 2) {
survivorlength.push_back(new int (1));
survivorlength.back() = y - (*its)[graph_bundle].tstart;
pop.release(its); // erase returns an iterator pointing to the new location of the element that followed the last element erased
//by the function call, which is the list end if the operation erased the last element
//in the sequence.
z = pop.size();
}
}

//cout << "system size is " << systemsize << endl;

//cout << "Population sum is " << popsum << endl;

//cout << "Copy no sum is " << totsumcheck << endl;

//cout << "Population size is " << pop.size() << endl;
} else {
Population::iterator its;
double popsum = 0;

if (sysexp == 1) {
systemsize = (((double) pop.size()) / ((double)1 + (double) pop.size()))
* (int) sysmax;
}

double totsumcheck = 0;

double x = 0;
double fitav = 0;

/* Calculate fitness */
for (its = pop.begin(); its != pop.end(); its++) {
if (maintenance == 0) {
x = (*its)[graph_bundle].abtotal - 5;
} else {
x = ((*its)[graph_bundle].abtotal / (*its)[graph_bundle].maintotal) - 5;
}

x *= x;

(*its)[graph_bundle].fittotal = (25 - x);
if ((*its)[graph_bundle].fittotal < 0) {
(*its)[graph_bundle].fittotal = 0;
```

```
}

fitav += (*its)[graph_bundle].fittotal;

(*its)[graph_bundle].normfit = (*its)[graph_bundle].fittotal;
}

/* Normalise */
if (norm == 1) {
fitav /= (double)pop.size();

if (fitav != 0) {
for (its = pop.begin(); its != pop.end(); its++) {
(*its)[graph_bundle].normfit /= fitav;
}
}
}

/* Calculate population change */
for (its = pop.begin(); its != pop.end(); ++its) {
popsum += floor(
(((*its)[graph_bundle].normfit)
* ((*its)[graph_bundle].copyno)));
}

for (its = pop.begin(); its != pop.end(); ++its) {
(*its)[graph_bundle].copyno = floor(
((((*its)[graph_bundle].normfit)
* (double)((*its)[graph_bundle].copyno)) / popsum)
* systemsize);
totsumcheck += (*its)[graph_bundle].copyno;
}

unsigned int z = pop.size();

for (unsigned int yy = 0; yy < z; yy++) {
its = pop.begin();
advance(its, yy);

if ((*its)[graph_bundle].copyno < 2) {
survivorlength.push_back(new int (1));
survivorlength.back() = y - (*its)[graph_bundle].tstart;
pop.release(its); // erase returns an iterator pointing to the new location of the element that followed the last element erased by
//the function call, which is the list end if the operation erased the last element in the sequence.
z = pop.size();
}
}

}
}
;

void popgrowth(Genopop &pop, doublelists &ablist, doublelists &mainlist,
ptr_vector<double> &copynums, int &systemsize, int &specext, doublelists &age) {
Genopop::iterator its;

doublelists::iterator itm;
doublelists::iterator ita;
ptr_vector<double>::iterator itc;

double popsum = 0;

//cout << pop.size() << endl;

if (sysexp == 1) {
systemsize = (((double) pop.size()) / ((double)1 + (double) pop.size()))
* (int) sysmax;
}

double totsumcheck = 0;

if (maintenance == 1) {
```

```
itm = mainlist.begin();
itc = copynums.begin();

for (ita = ablist.begin(); ita != ablist.end(); ++ita) {
popsum += floor(
((std::accumulate((*ita).begin(), (*ita).end(), (double) 0)
/ std::accumulate((*itm).begin(), (*itm).end(),
(double) 0)) * (*itc)));
++itm;
++itc;
//cout << popsum << endl;
}

itm = mainlist.begin();
itc = copynums.begin();

for (ita = ablist.begin(); ita != ablist.end(); ++ita) {
(*itc) = floor(
((((std::accumulate((*ita).begin(), (*ita).end(), (double) 0)
/ std::accumulate((*itm).begin(), (*itm).end(),
(double) 0)) * (*itc))) / popsum)
* systemsize);
//cout << "New no. of individuals " << (*its)[graph_bundle].copyno
//<< endl;
totsumcheck += (*itc);
//cout << totsumcheck << endl;
++itm;
++itc;
}

} else {

itc = copynums.begin();

for (ita = ablist.begin(); ita != ablist.end(); ++ita) {
popsum += floor(
(std::accumulate((*ita).begin(), (*ita).end(), (double) 0)
* (*itc)));
++itc;
//cout << "Number of individuals " << (*its)[graph_bundle].copyno << endl;
//cout << "Ability score " << (*its)[graph_bundle].abtotal << endl;
}

itc = copynums.begin();
ita = ablist.begin();

for (its = pop.begin(); its != pop.end(); ++its) {
(*itc) = floor(
((std::accumulate((*ita).begin(), (*ita).end(), (double) 0)
* (*itc)) / popsum) * systemsize);
//cout << "New no. of individuals " << (*its)[graph_bundle].copyno
//<< endl;
totsumcheck += (*itc);
++itc;
++ita;
}
}

unsigned int z = pop.size();

for (unsigned int y = 0; y < z; y++) {
its = pop.begin();
itc = copynums.begin();
ita = ablist.begin();
if (maintenance == 1) {
itm = mainlist.begin();
advance(itm, y);
}

advance(its, y);
advance(itc, y);
advance(ita, y);

if ((*itc) < 1) {
pop.release(its); // erase returns an iterator pointing to the new location of the element that followed the last element erased
//by the function call, which is the list end if the operation erased the last element in the sequence.
copynums.release(itc);
ablist.release(ita);
specext += 1;

if (maintenance == 1) {
mainlist.release(itm);
}
z = pop.size();
}
}

//cout << "system size is " << systemsize << endl;

//cout << "Population sum is " << popsum << endl;

//cout << "Copy no sum is " << totsumcheck << endl;

//cout << "Population size is " << pop.size() << endl;

}
;

void selection(Population &pop, int &realsize, int y, ptr_vector<int> &survivorlength) {
Population::iterator its;

if (levelselect == 1) {

ptr_vector<double> selectar;

int maxlevel;
maxlevel = levelcount(pop);

for (int a = 0; a < (maxlevel + 1); a++) {
selectar.push_back(new double (1) );
//selectar[a] = (3 * (a + 1)) / (1 + (3 * (a + 1))); // Creates a monotonically increasing selection cut-off.
//selectar[a] = 0.9; // Constant function
//selectar[a] = a; // Linear function
//selectar[a] = 1/(a+2); // Monotonically decreasing selection function.
selectar[a] = 0.01;
}

ptr_vector<double> levasum, levmsum, levsum;

//cout << "Population size " << pop.size() << endl;

int tmax = pop.size();
int count = 1;

for (int t = 0; t < tmax;) {
its = pop.begin();
advance(its, t);
int maxlev = (*its)[graph_bundle].maxlevel;
//cout << "Number of vertices " << num_vertices(*its) << endl;

ptr_vector<double> levasum, levmsum, levsum;

for (int b = 0; b < (maxlev + 1); b++) {
levasum.push_back(new double (1) );
levasum[b] = 0;
levmsum.push_back(new double (1) );
levmsum[b] = 0;
levsum.push_back(new double (1) );
levsum[b] = 0;
}

for (unsigned int z = 0; z < num_vertices(*its); z++) {
if (maintenance == 0) {
```

```
levsum[(*its)[z].level] += (*its)[z].ability;
} else {
levasum[(*its)[z].level] += (*its)[z].ability;
levmsum[(*its)[z].level] += (*its)[z].maintain;
}
}

if (maintenance == 1) {
for (int b = 0; b < (maxlev + 1); b++) {
levsum[b] = levasum[b] / levmsum[b];
}
}

double randdeath = 0;

for (int yy = 0; yy < (maxlev + 1);) {
//cout << "Level sum " << y << " is " << levsum[y] << endl;
//cout << "Selection is " << selectar[y] << endl;
randdeath = rn();

if (randdeath < (selectar[yy]/levsum[yy])) {
//if (randdeath < (0.25/(*its)[graph_bundle].abtotal)){
survivorlength.push_back(new int (1));
survivorlength.back() = y - (*its)[graph_bundle].tstart;
//cout << "survivorback = " << survivorlength.back() << endl;
pop.release(its);
yy = maxlev + 1;
//cout << "population is " << pop.size() << endl;
} else {
yy++;
if (yy == maxlev + 1) {
t++;
}
}
}

levsum.release();
if (maintenance == 1) {
levmsum.release();
levasum.release();
}

//cout << "Count is" << count << endl;
if (count == tmax) {
t = tmax;
}
++count;
}

//cout << "Population size " << pop.size() << endl;

selectar.release();
} else {
if (peakselect == 0) {
int tmax = pop.size();
int count = 1;
double randdeath = 0;

for (int t = 0; t < tmax;) {
its = pop.begin();
advance(its, t);
randdeath = rn();
if (randdeath < ((double)0.1/(*its)[graph_bundle].abtotal)){
survivorlength.push_back(new int (1));
survivorlength.back() = y - (*its)[graph_bundle].tstart;
pop.release(its);
} else {
t++;
}

if (count == tmax) {
t = tmax;
}
++count;
}
} else {
const double pi = boost::math::constants::pi<double>();
int tmax = pop.size();
int count = 1;
double peak = pi / 3;
double dist = 0;
double randdeath = 0;

for (int t = 0; t< tmax;) {
its = pop.begin();
advance(its, t);
if (maintenance == 0) {
dist = peak - (*its)[graph_bundle].abtotal;
dist *= dist;
dist = sqrt(dist);
randdeath = rn();
if (randdeath < dist) {
survivorlength.push_back(new int (1));
survivorlength.back() = y - (*its)[graph_bundle].tstart;
pop.release(its);
} else {
t++;
}

if (count == tmax) {
t = tmax;
}
++count;

} else {
dist = peak - ((*its)[graph_bundle].abtotal / (*its)[graph_bundle].maintotal);
dist *= dist;
dist = sqrt(dist);
randdeath = rn();
if (randdeath < dist) {
survivorlength.push_back(new int (1));
survivorlength.back() = y - (*its)[graph_bundle].tstart;
pop.release(its);
} else {
t++;
}

if (count == tmax) {
t = tmax;
}
++count;

}
}
}
}


// Top n% survive
/* ptr_vector<double> totfit;

for (its = pop.begin(); its != pop.end(); its++) {
totfit.push_back(new double (1));
if (maintenance == 1) {
totfit.back() = ((*its)[graph_bundle].abtotal)
/ ((*its)[graph_bundle].maintotal);
} else {
totfit.back() = (*its)[graph_bundle].abtotal;
}
}

sort(totfit.begin(), totfit.end());

ptr_vector<double>::iterator itf;
```

```
itf = unique(totfit.begin(), totfit.end());

totfit.resize(distance(totfit.begin(), itf));
realsize = totfit.size();

reverse(totfit.begin(), totfit.end());

int r = totfit.size();

if (r > 3) {
totfit.resize(3);
}
double kprob = 0;
totfit.pop_back();

int tmax = pop.size();
int count = 1;

for (int t = 0; t < tmax;) {
its = pop.begin();
advance(its, t);
if (totfit.size() > 0) {
if (maintenance == 1) {
if ((((*its)[graph_bundle].abtotal)
/ ((*its)[graph_bundle].maintotal)) < totfit.back()) {
kprob = rn();
if (kprob < 0.9) {
survivorlength.push_back(new int (1));
survivorlength.back() = y - (*its)[graph_bundle].tstart;
pop.release(its);
} else {
t++;
}
} else {
t++;
}
} else {
if ((*its)[graph_bundle].abtotal < totfit.back()) {
kprob = rn();
if (kprob < 0.9) {
survivorlength.push_back(new int (1));
survivorlength.back() = y - (*its)[graph_bundle].tstart;
pop.release(its);
} else {
t++;
}
} else {
t++;
}
}
}
if (count == tmax) {
t = tmax;
}

++count;
}

totfit.release();*/
}
;

void selection(Genopop &pop, doublelists &ab, doublelists &main) {
Genopop::iterator its;

doublelists::iterator ita;
doublelists::iterator itm;

ptr_vector<double> selectar;

int maxlevel;
```

```
maxlevel = levelcount(pop);

for (int a = 0; a < (maxlevel + 1); a++) {
selectar.push_back(new double (1) );
//selectar[a] = (3 * (a + 1)) / (1 + (3 * (a + 1))); // Creates a monotonically increasing selection cut-off.
selectar[a] = 0.0; // Constant function
//selectar[a] = a; // Linear function
//selectar[a] = 1/(a+2); // Monotonically decreasing selection function.
}

//cout << "Population size " << pop.size() << endl;

int tmax = pop.size();
int count = 1;

/* Population loop */
for (int t = 0; t < tmax;) {
its = pop.begin();
ita = ab.begin();
itm = main.begin();
advance(its, t);
advance(ita, t);
advance(itm, t);

int maxlev = 0;

//for (string_find_iterator It = make_find_iterator((*its),
//first_finder("[", is_iequal())); It != string_find_iterator();
//++It) {
//maxlev += 1;
//}

double levasum, levmsum, levsum;

char_separator<char> sep("[]");
tokeniser tokens(*its, sep);

int countnodes = 0;
int pastnodes = 0;

tokeniser::iterator beg = tokens.begin();

/* Level loop */
for (int y = 0; y < maxlev;) {

string levelstring = *beg;

for (string_find_iterator it = make_find_iterator((levelstring),
first_finder("{", is_iequal()));
it != string_find_iterator(); ++it) {
countnodes += 1;
}

if (maintenance == 0) {
levsum = std::accumulate((*ita).begin() + pastnodes,
(*ita).begin() + pastnodes + countnodes, (double) 0);
} else {
levasum = std::accumulate((*ita).begin() + pastnodes,
(*ita).begin() + pastnodes + countnodes, (double) 0);
levmsum = std::accumulate((*itm).begin() + pastnodes,
(*itm).begin() + pastnodes + countnodes, (double) 0);
}

if (maintenance == 1) {
levsum = levasum / levmsum;
}

if (levsum < (selectar[y])) {
pop.erase(its);
if (maintenance == 1) {
main.erase(itm);
}
ab.erase(ita);
y = maxlev;
} else {
y++;
pastnodes = countnodes;
countnodes = 0;
if (y == maxlev) {
t++;
} else {
beg++;
}
}

}

if (count == tmax) {
t = tmax;
}

++count;

}

//cout << "Population size " << pop.size() << endl;

selectar.release();

}


/* Mutation process */

void mutation(Population &pop, Engine &eng,
Normal &normdist, ptr_vector< ptr_vector<int> > &balls, int y) {

double h = 0;
Population::iterator itspec;
itspec = pop.begin(); // Sets iterator to start of species list.
int specmaxlevel = 0;
int maxmaxlevel = 0;
double numbervert = 0;

if (dynamics == 1) {
// Selects the species to be mutated. Species with high copy numbers more likely to mutate.
double specsum = 0;
for (itspec = pop.begin(); itspec != pop.end(); itspec++) {
specsum += ((*itspec)[graph_bundle].copyno) * num_vertices(*itspec);
}

h = rn();

double speccop = 0;

itspec = pop.begin();

for (unsigned int i = 0; i < pop.size(); i++) {
speccop += ((double)((*itspec)[graph_bundle].copyno * (double)num_vertices(*itspec))
/ specsum);
if ((h < speccop)
&& (h
>= speccop
- ((double)((*itspec)[graph_bundle].copyno
* (double)num_vertices(*itspec)) / specsum))) {
h = i;
i = pop.size();
} else {
++itspec;
}
}

} else {
double specsum = 0;
```

```
for (itspec = pop.begin(); itspec != pop.end(); itspec++) {
specsum += num_vertices(*itspec);
}

h = rn();

double speccop = 0;

itspec = pop.begin();

for (unsigned int i = 0; i < pop.size(); i++) {
speccop += ((double)num_vertices(*itspec) / specsum);
if ((h < speccop)
&& (h >= speccop - ((double)num_vertices(*itspec) / specsum))) {
h = i;
i = pop.size();
} else {
++itspec;
}
}

if (analytics == 1) {
//h = pop.size() - 1;
h = 0;
}

}

itspec = pop.begin();
advance(itspec, h); // Advances iterator to species selected for mutation.

doublelists::iterator ita;

numbervert = num_vertices(*itspec);
int mutnode = 0;
if (constraint == 0) {
mutnode = floor(rn() * numbervert); // Choose a node in the species to mutate.
}
else {
int node1 = 0;
double probnode = rn();
double nodesum = 0;
double nodeop = 0;

for (int k = 0; k < num_vertices(*itspec); k++) {
nodesum += (((double)in_degree(k,*itspec)+(double)out_degree(k,*itspec))/(2*(double)numbervert));
}

for (int j = 0; j < num_vertices(*itspec);j++) {
nodeop += (((double)in_degree(j,*itspec)+(double)out_degree(j,*itspec))/(2*(double)numbervert))/nodesum;
if ((probnode < nodeop)
&& (probnode >= nodeop - ((((double)in_degree(j,*itspec)+(double)out_degree(j,*itspec))/(2*(double)numbervert))/nodesum))) {
mutnode = j;
j = num_vertices(*itspec);
}
}

}

double p1 = rn(); // Generates a uniform random number between 0 and 1 for probabilities.

if (p1 >= pnew) {
double p2 = rn(); // Generates a uniform random number between 0 and 1 for probabilities.

/* Copying a node of the species */
if (p2 < qcopy) {

pop.push_back(new Bodyplan);
copy_graph(*itspec, pop.back());

pop.back()[graph_bundle].ancestor = (*itspec)[graph_bundle].identity;
```

```
for (unsigned int b = 0; b < num_vertices(pop.back()); b++) {
pop.back()[b].level = (*itspec)[b].level;
pop.back()[b].ability = (*itspec)[b].ability;
if (maintenance == 1) {
pop.back()[b].maintain = (*itspec)[b].maintain;
}
}

(pop.back())[graph_bundle].abtotal =
(*itspec)[graph_bundle].abtotal;
if (maintenance == 1) {
(pop.back())[graph_bundle].maintotal =
(*itspec)[graph_bundle].maintotal;
}

// Create a copy of the chosen node in the new species.
vertex_t copy = add_vertex(pop.back()); // Adds a vertex to the graph.

pop.back()[copy].level = (*itspec)[mutnode].level; // Copies node level number.
pop.back()[copy].colour = (*itspec)[mutnode].colour;

if (pop.back()[copy].level != 0) {
pop.back()[copy].ability = (*itspec)[mutnode].ability; // Copies ability score
} else {
pop.back()[copy].ability = normal_gen(eng, normdist);
if (pop.back()[copy].ability < 0) {
pop.back()[copy].ability = 0 - pop.back()[copy].ability;
}
}

if (maintenance == 1) {
if (pop.back()[copy].level == 0) {
pop.back()[copy].maintain = normal_gen(eng, normdist);
//if (rmain == 1) {
//while (pop.back()[copy].maintain
// > pop.back()[copy].ability) {
//pop.back()[copy].maintain = normal_gen(eng,
// normdist);
//}
//}
if (pop.back()[copy].maintain < 0) {
pop.back()[copy].maintain = 0
- pop.back()[copy].maintain;
}
} else {
pop.back()[copy].maintain = (*itspec)[mutnode].maintain;
}

}

//cout << "Copy all ability score is " << pop.back()[copy].ability
//<< endl;
//cout << "Copy all maintenance score is "
//<< pop.back()[copy].maintain << endl;

if (maintenance == 1 && levelcost == 1) {
pop.back()[copy].maintain = pow(pop.back()[copy].maintain,(double)metscale);
}

//cout << "Copied ability score is " << pop.back()[copy].ability
//<< endl;

if (mult == 0) {
if (maintenance == 1) {
(pop.back())[graph_bundle].maintotal +=
pop.back()[copy].maintain; // Updates ability score on the level the node was added.
}
(pop.back())[graph_bundle].abtotal += pop.back()[copy].ability; // Updates ability score on the level the node was added.
} else {
if (maintenance == 1) {
(pop.back())[graph_bundle].maintotal *=
pop.back()[copy].maintain;
```

```
}
(pop.back())[graph_bundle].abtotal *= pop.back()[copy].ability; // Updates ability score on the level the node was added.

}
//cout << "Updates ability score is "
//<< (ab.back())[pop.back()[copy].level] << endl;

// The copied node's edges are transferred to the newly created node. This maintains downstream connections.
outedgeiterator edgeoutiterBegin, edgeoutiterEnd,
edgeoutiterCurrent;
boost::tie(edgeoutiterBegin, edgeoutiterEnd) = out_edges(mutnode, *itspec);
for (edgeoutiterCurrent = edgeoutiterBegin;
edgeoutiterCurrent != edgeoutiterEnd;
++edgeoutiterCurrent) {
add_edge(copy, target(*edgeoutiterCurrent, *itspec),
pop.back()); // Copies out edges over.
}

int oldcop = (*itspec)[graph_bundle].copyno;

if (dynamics == 1 || resources == 1) {
pop.back()[graph_bundle].maxlevel =
(*itspec)[graph_bundle].maxlevel;
(pop.back())[graph_bundle].copyno = ceil(
0.05 * (*itspec)[graph_bundle].copyno);
(*itspec)[graph_bundle].copyno = floor(
0.95 * (*itspec)[graph_bundle].copyno);
}

(pop.back())[graph_bundle].tstart = y;

int wholedeath;
double addresources,adddeath;

if (resources == 1) {
if (totrescol == 1 && breakup == 1) {
addresources = ceil((0.05 * oldcop));
adddeath = addresources / (double)num_vertices(*itspec);
wholedeath = ceil(adddeath);
(*itspec)[graph_bundle].copyno -= wholedeath;

balls.front()[0] += ceil((((double)wholedeath - adddeath)
* (double)num_vertices(*itspec)) - 0.00000000001);
} else {
// Ball tallying for colours
ptr_vector< ptr_vector<int> > colournums;
for (unsigned int bc = 0; bc < balls.size(); bc++) {
colournums.push_back(new ptr_vector<int> );
if (breakup == 0){
for (unsigned int bl = 0; bl < balls[bc].size(); bl++) {
colournums.back().push_back(new int (1));
colournums.back().back() = 0;
}
} else {
colournums.back().push_back(new int (1));
colournums.back().back() = 0;
}
}

// Counting colours in parent
if (breakup == 1) {
for (unsigned int l = 0; l < num_vertices(*itspec); l++) {
colournums[(*itspec)[l].colour][0] += 1; // Number of balls of different colours in parent
}
} else {
for (unsigned int l = 0; l < num_vertices(*itspec); l++) {
colournums[(*itspec)[l].colour][(*itspec)[l].level] += 1; // Number of balls of different colours and levels in parent
}
}

// Ball pool tallying for mutated colour
if (breakup == 1) {
```

```
addresources = (ceil((0.05 * oldcop))); // of a particular ball colour (because only add one new ball)
adddeath = addresources
/ (double) colournums[pop.back()[copy].colour][0]; // Works out additional death due to increase in particular colour
wholedeath = ceil(adddeath); // Number of individuals of parent that die due to resource reallocation
(*itspec)[graph_bundle].copyno -= wholedeath;
} else {
addresources = (ceil((0.05 * oldcop))); // of a particular ball colour and level (because only add one new ball)
adddeath = addresources
/ (double) colournums[pop.back()[copy].colour][pop.back()[copy].level]; // Works out additional death due to increase in particular ball type
wholedeath = ceil(adddeath); // Number of individuals of parent that die due to resource reallocation
(*itspec)[graph_bundle].copyno -= wholedeath;
}

// Other colour ball pool tallying
if (breakup == 1) {
ptr_vector<int>::iterator ballit;
int balltrack = 0;
for (ballit = balls.begin(); ballit != balls.end();
ballit++) {
if (balltrack != pop.back()[copy].colour) {
(*ballit) += wholedeath * colournums[balltrack][0];
} else {
(*ballit) += ceil((wholedeath - adddeath)
* colournums[balltrack][0]);
}
balltrack++;
}
} else {
for (unsigned int bc = 0; bc < balls.size(); bc++) {
for (unsigned int bl = 0; bl < balls[bc].size(); bl++) {
if (bc != pop.back()[copy].colour) {
balls[bc][bl] += wholedeath * colournums[bc][bl];
} else {
balls[bc][bl] += ceil((wholedeath - adddeath)
* colournums[bc][bl]);
}
}
}

}
}
}

if (resources == 1) {
int totalballs = 0;

for (Population::iterator its = pop.begin(); its != pop.end(); its++) {
totalballs += (*its)[graph_bundle].copyno * num_vertices(*its);
}

if (breakup == 1) {
for (unsigned int bc = 0; bc < balls.size(); bc++) {
totalballs += balls[bc][0];
}
} else {
for (unsigned int bc = 0; bc < balls.size(); bc++) {
totalballs += accumulate(balls[bc].begin(), balls[bc].end(), (double) 0);
}
}

if (totalballs != 1050) {
if (fluctuations == 0) {
cout << "COPY NODE AND SUBGRAPH " << totalballs << endl;
}
}
}
if (ages == 1) {
cout << (*itspec)[graph_bundle].copyno << endl;
for (signed int tulip = 0; tulip < (oldcop - (*itspec)[graph_bundle].copyno); tulip++) {
(*itspec)[graph_bundle].age.release((*itspec)[graph_bundle].age.begin());
}
for (int tulip2 = 0; tulip2 < (pop.back())[graph_bundle].copyno; tulip2++) {
```

```
(pop.back())[graph_bundle].age.push_back(new int (0) );
}
}


}

/* Copy the node but not subgraph */
else {
pop.push_back(new Bodyplan);
copy_graph(*itspec, pop.back());
(pop.back())[graph_bundle].abtotal =
(*itspec)[graph_bundle].abtotal;
if (maintenance == 1) {
(pop.back())[graph_bundle].maintotal =
(*itspec)[graph_bundle].maintotal;
}

pop.back()[graph_bundle].ancestor = (*itspec)[graph_bundle].identity;

for (unsigned int b = 0; b < num_vertices(pop.back()); b++) {
pop.back()[b].level = (*itspec)[b].level;
pop.back()[b].ability = (*itspec)[b].ability;
if (maintenance == 1) {
pop.back()[b].maintain = (*itspec)[b].maintain;
}
}

// Create a copy of the chosen node in the new species.
vertex_t copy = add_vertex(pop.back()); // Adds a vertex to the graph.

pop.back()[copy].level = (*itspec)[mutnode].level; // Copies node level number.
pop.back()[copy].colour = (*itspec)[mutnode].colour;

if (mult == 0) {
pop.back()[copy].ability = 0; // Assign ability score of 0.
if (maintenance == 1) {
pop.back()[copy].maintain = 0;
}
} else {
pop.back()[copy].ability = 1;
if (maintenance == 1) {
pop.back()[copy].maintain = 1;
}
}

if (pop.back()[copy].level != 0) {

Bodyplan holdinggraph;
copy_graph(pop.back(), holdinggraph);

int highsublev = 0;
/*int outdeg = out_degree(copy, holdinggraph);

while (outdeg == 0
|| highsublev != (holdinggraph[copy].level - 1)) {
holdinggraph = pop.back();
highsublev = 0;

for (unsigned int y = 0; y < num_vertices(holdinggraph);
y++) {
double p3 = rn();
if (holdinggraph[y].level < holdinggraph[copy].level) {
if (p3 < beta) {
add_edge(copy, y, holdinggraph);
if (holdinggraph[y].level > highsublev) {
highsublev = holdinggraph[y].level;
}
if (mult == 0) {
holdinggraph[copy].ability +=
holdinggraph[y].ability;
if (maintenance == 1) {
```

```
holdinggraph[copy].maintain +=
holdinggraph[y].maintain;
}
} else {
holdinggraph[copy].ability *=
holdinggraph[y].ability;
if (maintenance == 1) {
holdinggraph[copy].maintain *=
holdinggraph[y].maintain;
}
}
}
}
}*/

// Create new random subgraph
ptr_vector<int> samelev;
for (unsigned int evert = 0; evert < num_vertices(holdinggraph); evert++) {
double p3 = rn();
if (holdinggraph[evert].level < holdinggraph[copy].level) {
if (p3 < beta) {
add_edge(copy, evert, holdinggraph);
if (mult == 0) {
holdinggraph[copy].ability +=
holdinggraph[evert].ability;
if (maintenance == 1) {
holdinggraph[copy].maintain +=
holdinggraph[evert].maintain;
}
} else {
holdinggraph[copy].ability *=
holdinggraph[evert].ability;
if (maintenance == 1) {
holdinggraph[copy].maintain *=
holdinggraph[evert].maintain;
}
}

}
if (holdinggraph[evert].level == (holdinggraph[copy].level - 1)) {
samelev.push_back(new int (1));
samelev.back() = evert;
}
}
}

/* Choose node on level l-1 and ensure that connection is present */
double p5 = rn();
int cnode = floor(p5 * samelev.size());

if (edge(copy, samelev[cnode], holdinggraph).second == false){
if (mult == 0) {
holdinggraph[copy].ability +=
holdinggraph[samelev[cnode]].ability;
if (maintenance == 1) {
holdinggraph[copy].maintain +=
holdinggraph[samelev[cnode]].maintain;
}
} else {
holdinggraph[copy].ability *=
holdinggraph[samelev[cnode]].ability;
if (maintenance == 1) {
holdinggraph[copy].maintain *=
holdinggraph[samelev[cnode]].maintain;
}
}
}

int outdeg = out_degree(copy, holdinggraph);

pop.back().clear();
copy_graph(holdinggraph, pop.back());
```

```
holdinggraph.clear();

} else {
pop.back()[copy].ability = normal_gen(eng, normdist);
if (pop.back()[copy].ability < 0) {
pop.back()[copy].ability = 0 - pop.back()[copy].ability;
}

if (maintenance == 1) {
pop.back()[copy].maintain = normal_gen(eng, normdist);
//if (rmain == 1) {
//while (pop.back()[copy].maintain
// > pop.back()[copy].ability) {
//pop.back()[copy].maintain = normal_gen(eng,
// normdist);
//}
//}
if (pop.back()[copy].maintain < 0) {
pop.back()[copy].maintain = 0
- pop.back()[copy].maintain;
}
}
}

if (maintenance == 1 && levelcost == 1) {
pop.back()[copy].maintain = pow(pop.back()[copy].level,(double)metscale);
}

if (mult == 0) {
if (maintenance == 1) {
pop.back()[graph_bundle].maintotal +=
pop.back()[copy].maintain;
}
(pop.back())[graph_bundle].abtotal += pop.back()[copy].ability; // Updates ability score on the level the node was added.
} else {
if (maintenance == 1) {
pop.back()[graph_bundle].maintotal *=
pop.back()[copy].maintain;
}
(pop.back())[graph_bundle].abtotal *= pop.back()[copy].ability; // Updates ability score on the level the node was added.

}

//cout << "Copy node ability score is " << pop.back()[copy].ability
//<< endl;
//cout << "Copy node maintenance score is "
//<< pop.back()[copy].maintain << endl;

(pop.back())[graph_bundle].tstart = y;

int oldcop = (*itspec)[graph_bundle].copyno;

if (dynamics == 1 || resources == 1) {
pop.back()[graph_bundle].maxlevel =
(*itspec)[graph_bundle].maxlevel;
(pop.back())[graph_bundle].copyno = ceil(
0.05 * (*itspec)[graph_bundle].copyno);
//cout << "Copy new " << (pop.back())[graph_bundle].copyno << endl;
(*itspec)[graph_bundle].copyno = floor(
0.95 * (*itspec)[graph_bundle].copyno);
//cout << "Copy new " << (*itspec)[graph_bundle].copyno << endl;

}
if (resources == 1) {
if (totrescol == 1 && breakup == 1) {
double addresources, adddeath;
addresources = ceil((0.05 * oldcop));
adddeath = addresources / num_vertices(*itspec);
int wholedeath = ceil(adddeath);
(*itspec)[graph_bundle].copyno -= wholedeath;
balls.front()[0] += ceil((((double)wholedeath - adddeath)
* (double)num_vertices(*itspec)) - 0.00000000001);
```

```
} else {
// Ball tallying for colours
ptr_vector< ptr_vector<int> > colournums;
for (unsigned int bc = 0; bc < balls.size(); bc++) {
colournums.push_back(new ptr_vector<int> );
if (breakup == 0){
for (unsigned int bl = 0; bl < balls[bc].size(); bl++) {
colournums.back().push_back(new int (1));
colournums.back().back() = 0;
}
} else {
colournums.back().push_back(new int (1));
colournums.back().back() = 0;
}
}

// Counting colours in parent
if (breakup == 1) {
for (unsigned int l = 0; l < num_vertices(*itspec); l++) {
colournums[(*itspec)[l].colour][0] += 1; // Number of balls of different colours in parent
}
} else {
for (unsigned int l = 0; l < num_vertices(*itspec); l++) {
colournums[(*itspec)[l].colour][(*itspec)[l].level] += 1; // Number of balls of different colours and levels in parent
}
}

double addresources, adddeath;
int wholedeath;
// Ball pool tallying for mutated colour
if (breakup == 1) {
addresources = (ceil((0.05 * oldcop))); // of a particular ball colour (because only add one new ball)
adddeath = addresources
/ (double) colournums[pop.back()[copy].colour][0]; // Works out additional death due to increase in particular colour
wholedeath = ceil(adddeath); // Number of individuals of parent that die due to resource reallocation
(*itspec)[graph_bundle].copyno -= wholedeath;
} else {
addresources = (ceil((0.05 * oldcop))); // of a particular ball colour and level (because only add one new ball)
adddeath = addresources
/ (double) colournums[pop.back()[copy].colour][pop.back()[copy].level]; // Works out additional death due to increase in particular ball type
wholedeath = ceil(adddeath); // Number of individuals of parent that die due to resource reallocation
(*itspec)[graph_bundle].copyno -= wholedeath;
}

// Other colour ball pool tallying
if (breakup == 1) {
ptr_vector<int>::iterator ballit;
int balltrack = 0;
for (ballit = balls.begin(); ballit != balls.end();
ballit++) {
if (balltrack != pop.back()[copy].colour) {
(*ballit) += wholedeath * colournums[balltrack][0];
} else {
(*ballit) += ceil((wholedeath - adddeath)
* colournums[balltrack][0]);
}
balltrack++;
}
} else {
for (unsigned int bc = 0; bc < balls.size(); bc++) {
for (unsigned int bl = 0; bl < balls[bc].size(); bl++) {
if (bc != pop.back()[copy].colour) {
balls[bc][bl] += wholedeath * colournums[bc][bl];
} else {
balls[bc][bl] += ceil((wholedeath - adddeath)
* colournums[bc][bl]);
}
}
}

}
```

```
}
}

if (resources == 1) {
int totalballs = 0;

for (Population::iterator its = pop.begin(); its != pop.end(); its++) {
totalballs += (*its)[graph_bundle].copyno * num_vertices(*its);
}

if (breakup == 1) {
for (unsigned int bc = 0; bc < balls.size(); bc++) {
totalballs += balls[bc][0];
}
} else {
for (unsigned int bc = 0; bc < balls.size(); bc++) {
totalballs += accumulate(balls[bc].begin(), balls[bc].end(), (double) 0);
}
}

if (totalballs != 1050) {
if (fluctuations == 0) {
cout << "COPY NODE NEW SUBGRAPH " << totalballs << endl;
}
}
}
if (ages == 1) {
for (int tulip = 0; tulip < (oldcop - (*itspec)[graph_bundle].copyno); tulip++) {
(*itspec)[graph_bundle].age.release((*itspec)[graph_bundle].age.begin());
}
for (int tulip2 = 0; tulip2 < (pop.back())[graph_bundle].copyno; tulip2++) {
(pop.back())[graph_bundle].age.push_back(new int (0) );
}
}

}

}

/* New node created */
else {
maxmaxlevel = levelcount(pop);

specmaxlevel = (*itspec)[graph_bundle].maxlevel;

pop.push_back(new Bodyplan);
copy_graph(*itspec, pop.back());
pop.back()[graph_bundle].ancestor = (*itspec)[graph_bundle].identity;

(pop.back())[graph_bundle].abtotal = (*itspec)[graph_bundle].abtotal;
if (maintenance == 1) {
(pop.back())[graph_bundle].maintotal =
(*itspec)[graph_bundle].maintotal;
}

for (unsigned int b = 0; b < num_vertices(pop.back()); b++) {
pop.back()[b].level = (*itspec)[b].level;
pop.back()[b].ability = (*itspec)[b].ability;
//cout << "Node " << b << " ability is " << pop.back()[b].ability
//<< endl;
if (maintenance == 1) {
pop.back()[b].maintain = (*itspec)[b].maintain;
}

}

vertex_t newn2 = add_vertex(pop.back()); // Adds a vertex to the graph.

(pop.back())[newn2].level = specmaxlevel + 1; // New node level number.
double newcol = floor(rn() * balls.size());
pop.back()[newn2].colour = newcol; // Randomly generate new ball colour
```

```
if (mult == 0) {
pop.back()[newn2].ability = 0;
if (maintenance == 1) {
pop.back()[newn2].maintain = 0;
}
} else {
pop.back()[newn2].ability = 1;
if (maintenance == 1) {
pop.back()[newn2].maintain = 1;
}
}

// Create new subgraph //
Bodyplan holdinggraph;
copy_graph(pop.back(), holdinggraph);

// Create new random subgraph
ptr_vector<int> samelev;
for (unsigned int evert = 0; evert < num_vertices(holdinggraph); evert++) {
double p3 = rn();
if (holdinggraph[evert].level < holdinggraph[newn2].level) {
if (p3 < beta) {
add_edge(newn2, evert, holdinggraph);
if (mult == 0) {
holdinggraph[newn2].ability +=
holdinggraph[evert].ability;
if (maintenance == 1) {
holdinggraph[newn2].maintain +=
holdinggraph[evert].maintain;
}
} else {
holdinggraph[newn2].ability *=
holdinggraph[evert].ability;
if (maintenance == 1) {
holdinggraph[newn2].maintain *=
holdinggraph[evert].maintain;
}
}
}
if (holdinggraph[evert].level == (holdinggraph[newn2].level - 1)) {
samelev.push_back(new int (1));
samelev.back() = evert;
}
}
}

/* Choose node on level l-1 and ensure that connection is present */
double p5 = rn();
int cnode = floor(p5 * samelev.size());

if (edge(newn2, samelev[cnode], holdinggraph).second == false){
add_edge(newn2, samelev[cnode], holdinggraph);
if (mult == 0) {
holdinggraph[newn2].ability +=
holdinggraph[samelev[cnode]].ability;
if (maintenance == 1) {
holdinggraph[newn2].maintain +=
holdinggraph[samelev[cnode]].maintain;
}
} else {
holdinggraph[newn2].ability *=
holdinggraph[samelev[cnode]].ability;
if (maintenance == 1) {
holdinggraph[newn2].maintain *=
holdinggraph[samelev[cnode]].maintain;
}
}


if (num_edges(holdinggraph) == 0) {
cout << out_degree(newn2,holdinggraph) << endl;
```

```
}

samelev.erase(samelev.begin(), samelev.end());

}

pop.back().clear();
copy_graph(holdinggraph, pop.back());

holdinggraph.clear();

double prevlev = 0;

pop.back()[graph_bundle].maxlevel = (specmaxlevel + 1);


if (maintenance == 1 && levelcost ==1) {
pop.back()[newn2].maintain = pow(pop.back()[newn2].maintain,(double)metscale);
}


if (mult == 0) {
(pop.back())[graph_bundle].abtotal += pop.back()[newn2].ability;
if (maintenance == 1) {
pop.back()[graph_bundle].maintotal +=
pop.back()[newn2].maintain;
}
} else {
if (maintenance == 1) {
pop.back()[graph_bundle].maintotal *=
pop.back()[newn2].maintain;
}
(pop.back())[graph_bundle].abtotal *= pop.back()[newn2].ability;
}

int oldcop = (*itspec)[graph_bundle].copyno;

if (dynamics == 1 || resources == 1) {
(pop.back())[graph_bundle].copyno = ceil(
0.05 * (*itspec)[graph_bundle].copyno);
//cout << "New " << (pop.back())[graph_bundle].copyno << endl;
(*itspec)[graph_bundle].copyno = floor(
0.95 * (*itspec)[graph_bundle].copyno);
//cout << "New " << (*itspec)[graph_bundle].copyno << endl;
}

double addresources, adddeath;
int wholedeath;

if (breakup == 0) {
// Populating balls vector of appropriate levels and colour
if ((signed int) ((pop.back()[newn2].level + 1) - balls[pop.back()[newn2].colour].size()) > 0) {
for (int difflev = 0; difflev < ((pop.back()[newn2].level + 1) - balls[pop.back()[newn2].colour].size()); difflev++) {
balls[pop.back()[newn2].colour].push_back(new int (1));
balls[pop.back()[newn2].colour].back() = 0;
}
}
}

(pop.back())[graph_bundle].tstart = y;

//cout << "Resources" << endl;

if (resources == 1) {
if (totrescol == 1 && breakup == 1) {
addresources = ceil((0.05 * oldcop));
adddeath = addresources / (double)num_vertices(*itspec);
wholedeath = ceil(adddeath);
(*itspec)[graph_bundle].copyno -= wholedeath;
balls.front()[0] += ceil((((double)wholedeath - adddeath)
* (double)num_vertices(*itspec)) - 0.00000000001);
} else {
```

```
//cout << "Tally" << endl;
// Ball tallying for colours
ptr_vector< ptr_vector<int> > colournums;

for (unsigned int bc = 0; bc < balls.size(); bc++) {
colournums.push_back(new ptr_vector<int> );
if (breakup == 0){
for (unsigned int bl = 0; bl < balls[bc].size(); bl++) {
colournums.back().push_back(new int (1));
colournums.back().back() = 0;
}
} else {
colournums.back().push_back(new int (1));
colournums.back().back() = 0;
}
}
//cout << "Counting" << endl;
// Counting colours in parent
if (breakup == 1) {
for (unsigned int l = 0; l < num_vertices(*itspec); l++) {
colournums[(*itspec)[l].colour][0] += 1; // Number of balls of different colours in parent
}
} else {
for (unsigned int l = 0; l < num_vertices(*itspec); l++) {
colournums[(*itspec)[l].colour][(*itspec)[l].level] += 1; // Number of balls of different colours and levels in parent
}
}

//cout << "Reroll" << endl;
// Reroll node colour if no balls in parent or ball pool
double newcol = 0;
if (breakup == 1) {
while (colournums[(pop.back()[newn2].colour)][0] == 0
&& balls[pop.back()[newn2].colour][0] == 0) {
newcol = floor(rn() * (double) balls.size());
pop.back()[newn2].colour = newcol;
}
} else {
if (colournums[(pop.back()[newn2].colour)][pop.back()[newn2].level] == 0
&& balls[pop.back()[newn2].colour][pop.back()[newn2].level] == 0) {
while ((colournums[(pop.back()[newn2].colour)][pop.back()[newn2].level - 1] == 0
&& balls[pop.back()[newn2].colour][pop.back()[newn2].level - 1] == 0) || (colournums[(pop.back()[newn2].colour)][pop.back()[newn2].level] == 0
&& balls[pop.back()[newn2].colour][pop.back()[newn2].level] == 0)) {
newcol = floor(rn() * (double) balls.size());
pop.back()[newn2].colour = newcol;
}
}
}


addresources = ceil((0.05 * oldcop)); // Number of additional resources required due to mutation

if (breakup == 1) {
if (colournums[pop.back()[newn2].colour][0] != 0) {
adddeath = ceil(addresources / (double)colournums[pop.back()[newn2].colour][0]); // Number of parent deaths required to produce
//additional resources
if (adddeath > (*itspec)[graph_bundle].copyno) {
// Should never happen if 0.05 is the fraction used
balls[pop.back()[newn2].colour][0] -= (adddeath - (*itspec)[graph_bundle].copyno); // If the additional deaths are more than the
//parent can provide then take from ball pool
(*itspec)[graph_bundle].copyno = 0; // Balls used up from parents, no individuals remaining
} else {
(*itspec)[graph_bundle].copyno -= adddeath; // Subtract additional deaths from parent

// Other colour ball pool tallying
int balltrack = 0;
for (unsigned int bc = 0; bc < balls.size(); bc++) {
if (balltrack != pop.back()[newn2].colour) {
balls[bc][0] += adddeath * colournums[balltrack][0];
} else {
balls[bc][0] += ceil((adddeath - (addresources / (double)colournums[pop.back()[newn2].colour][0]))
* colournums[balltrack][0]);
}
balltrack++;
}
}
} else {

balls[pop.back()[newn2].colour][0] -= addresources; // Number of balls of a colour not present in parent to be removed from ball pool
adddeath = 0; // Additional death therefore 0
if (balls[pop.back()[newn2].colour][0] < 0) {
int ballcorrect = 0 - balls[pop.back()[newn2].colour][0]; // Number of balls wrongly removed from ball pool
pop.back()[graph_bundle].copyno -= ballcorrect; // Number of new mutant individuals, correcting for ball pool
(*itspec)[graph_bundle].copyno = oldcop - pop.back()[graph_bundle].copyno; // Correcting parent individuals
adddeath = 0;
}
}

} else {
//cout << "Ind" << endl;
if (colournums[pop.back()[newn2].colour][pop.back()[newn2].level] != 0) {
adddeath = ceil(addresources / (double)colournums[pop.back()[newn2].colour][pop.back()[newn2].level]); // Number of
//parent deaths required to produce additional resources
if (adddeath > (*itspec)[graph_bundle].copyno) {
// Should never happen if 0.05 is the fraction used
balls[pop.back()[newn2].colour][pop.back()[newn2].level] -= (adddeath - (*itspec)[graph_bundle].copyno); // If the
//additional deaths are more than the parent can provide then take from ball pool
(*itspec)[graph_bundle].copyno = 0; // Balls used up from parents, no individuals remaining
} else {
(*itspec)[graph_bundle].copyno -= adddeath; // Subtract additional deaths from parent

// Other colour ball pool tallying
for (unsigned int bc = 0; bc < balls.size(); bc++) {
for (unsigned int bl = 0; bl < balls[bc].size(); bl++) {
if (bc != pop.back()[newn2].colour) {
balls[bc][bl] += adddeath * colournums[bc][bl];
} else {
balls[bc][bl] += ceil((adddeath - (addresources / (double)colournums[pop.back()[newn2].colour][bl]))
* colournums[bc][bl]);
}
}
}
}
} else {
//cout << "Else381" << endl;
balls[pop.back()[newn2].colour][pop.back()[newn2].level] -= addresources; // Number of balls of a colour not
//present in parent to be removed from ball pool
adddeath = 0; // Additional death therefore 0
if (balls[pop.back()[newn2].colour][pop.back()[newn2].level] < 0) {
int ballcorrect = 0 - balls[pop.back()[newn2].colour][pop.back()[newn2].level]; // Number of balls wrongly removed from ball pool
pop.back()[graph_bundle].copyno -= ballcorrect; // Number of new mutant individuals, correcting for ball pool
(*itspec)[graph_bundle].copyno = oldcop - pop.back()[graph_bundle].copyno; // Correcting parent individuals
adddeath = 0;
}
}


}
}

}

//cout << "Ages" << endl;
if (ages == 1) {
for (int tulip = 0; tulip < (oldcop - (*itspec)[graph_bundle].copyno); tulip++) {
(*itspec)[graph_bundle].age.release((*itspec)[graph_bundle].age.begin());
}
for (int tulip2 = 0; tulip2 < (pop.back())[graph_bundle].copyno; tulip2++) {
(pop.back())[graph_bundle].age.push_back(new int (0) );
}
}


//cout << "Totalballs" << endl;
```

```
if (resources == 1) {
int totalballs = 0;

for (Population::iterator its = pop.begin(); its != pop.end(); its++) {
totalballs += (*its)[graph_bundle].copyno * num_vertices(*its);
}

if (breakup == 1) {
for (unsigned int bc = 0; bc < balls.size(); bc++) {
totalballs += balls[bc][0];
}
} else {
for (unsigned int bc = 0; bc < balls.size(); bc++) {
totalballs += accumulate(balls[bc].begin(), balls[bc].end(), (double) 0);
}
}

if (totalballs != 1050) {
if (fluctuations == 0) {
cout << "NEW NODE AND SUBGRAPH " << totalballs << endl;
}
}
}

}

}

void mutation(Genopop &pop, doublelists &ab, doublelists &main,
ptr_vector<double> &copy, Engine &eng,
Normal &normdist, int &balls, doublelists &age) {

double h = 0;
Genopop::iterator itspec;
itspec = pop.begin(); // Sets iterator to start of species list.
int numbervert = 0;
ptr_vector< double >::iterator itc;
doublelists::iterator itage;

if (dynamics == 1 || resources == 1) {
// Selects the species to be mutated. Species with high copy numbers more likely to mutate.
double specsum = 0;
int numvert = 0;
ptr_vector<double> speccophold;
itc = copy.begin();
for (itspec = pop.begin(); itspec != pop.end(); itspec++) {

for (string_find_iterator It = make_find_iterator((*itspec),
first_finder("{", is_iequal()));
It != string_find_iterator(); ++It) {
numvert += 1;
}
speccophold.push_back(new double (1) );
speccophold.back() = (*itc) * numvert;
specsum += ((*itc) * numvert);
itc++;
numvert = 0;

}

h = rn();

double speccop = 0;

for (unsigned int i = 0; i < pop.size(); i++) {
speccop += (speccophold[i] / specsum);
if ((h < speccop) && (h >= speccop - (speccophold[i] / specsum))) {
h = i;
i = pop.size();
}
}
```

```
if (analytics == 1) {
//h = pop.size() - 1;
h = 0;
}

numbervert = speccophold[h];

} else {
double specsum = 0;
ptr_vector<double> speccophold;
int numvert = 0;

for (itspec = pop.begin(); itspec != pop.end(); itspec++) {
for (string_find_iterator It = make_find_iterator((*itspec),
first_finder("{", is_iequal()));
It != string_find_iterator(); ++It) {
numvert += 1;
}
speccophold.push_back(new double (1) );
speccophold.back() = numvert;
specsum += numvert;
numvert = 0;
}

h = rn();

double speccop = 0;

for (unsigned int i = 0; i < pop.size(); i++) {
speccop += (speccophold[i] / specsum);
if ((h < speccop) && (h >= speccop - (speccophold[i] / specsum))) {
h = i;
i = pop.size();
}
}

if (analytics == 1) {
//h = pop.size() - 1;
h = 0;
}

numbervert = speccophold[h];

}

if (analytics == 1 && h != 0) {
cout << "H IS WRONG" << endl;
}
/* New mutant pushed to back of species/ability/maintenance lists */
doublelists::iterator ita;
doublelists::iterator itm;

pop.push_back(new string);
itspec = pop.begin();
advance(itspec, h); // Advances iterator to species selected for mutation.
pop.back() = (*itspec);

if (resources == 0) {
ab.push_back(new ptr_vector<double>);
ita = ab.begin();
advance(ita, h);
for (ptr_vector<double>::iterator itab = (*ita).begin();
itab != (*ita).end(); itab++) {
(ab.back()).push_back(new double (1) );
(ab.back()).back() = *itab;
}

if (maintenance == 1) {
itm = main.begin();
advance(itm, h);
for (ptr_vector<double>::iterator itma = (*itm).begin();
itma != (*itm).end(); itma++) {
```

```
(main.back()).push_back(new double (1) );
(main.back()).back() = *itma;
}
}
}
double p1 = rn(); // Generates a uniform random number between 0 and 1 for probabilities.

if (p1 >= pnew) {
double p2 = rn(); // Generates a uniform random number between 0 and 1 for probabilities.
/* Generate node to be copied */

double newnumvert = 0;

for (string_find_iterator It = make_find_iterator((*itspec),
first_finder("{", is_iequal())); It != string_find_iterator();
++It) {
newnumvert += 1;
}

unsigned int m1 = floor(rn() * newnumvert);

/* Copy node and subgraph */
if (p2 < qcopy) {
//cout << "COPY NODE AND SUBGRAPH" << endl;

// Create a copy of the chosen node in the new species.
//cout << "check 1" << endl;
char_separator<char> sep("[{}]");
tokeniser tokens(pop.back(), sep);

tokeniser::iterator node = tokens.begin();
node = tokens.begin();
advance(node, m1);

string copiednode = *node;
copiednode.insert(copiednode.begin(), '{');
copiednode.insert(copiednode.begin(), '}');
copiednode.push_back('}');

replace_nth(pop.back(), "}", m1, copiednode);
//cout << "check 2" << endl;
// Update ability and maintenance
//cout << "m1 +1 is " << m1+1 << endl;
//cout << "ab.back() size is " << (ab.back()).size() << endl;
if (resources == 0) {
if (m1 + 1 < (ab.back()).size()) {
ptr_vector<double>::iterator abit;
abit = (ab.back()).begin();
advance(abit, m1 + 1);
(ab.back()).insert(abit, new double (1) );
if (copiednode != "}{0}") {
(ab.back())[m1 + 1] = (ab.back())[m1];
} else {
(ab.back())[m1 + 1] = normal_gen(eng, normdist);
if ((ab.back())[m1 + 1] < 0) {
(ab.back())[m1 + 1] = 0 - (ab.back())[m1 + 1];
}
}
} else {
(ab.back()).push_back(new double (1) );
(ab.back()).back() = (ab.back())[m1];
}
//cout << "check 3" << endl;
if (maintenance == 1) {
if (m1 < (main.back()).size()) {
ptr_vector<double>::iterator mainit;
mainit = (main.back()).begin();
advance(mainit, m1 + 1);
(main.back()).insert(mainit, new double (1) );
if (copiednode != "}{0}") {
(main.back())[m1 + 1] = (main.back())[m1];
} else {
```

```
(main.back())[m1 + 1] = normal_gen(eng, normdist);
if ((main.back())[m1 + 1] < 0) {
(main.back())[m1 + 1] = 0
- (main.back())[m1 + 1];
}
}
} else {
(main.back()).push_back(new double (1) );
(main.back()).back() = (main.back())[m1];
}

}
}

//cout << "Updated maintenance and ability" << endl;
// Update upstream nodes
char_separator<char> levsep("[]");
char_separator<char> genesep("{}");
tokeniser levtok(pop.back(), levsep);
tokeniser::iterator levit = levtok.begin();

int minchangelev;
minchangelev = nodelevelcount(pop, m1);
advance(levit, minchangelev);
string level;
string gene;
ptr_list<string> mutant;
ptr_list<string> mutgene;

for (tokeniser::iterator lev2it = levtok.begin(); lev2it != levit;
lev2it++) {
level = *lev2it;
mutant.push_back(new string);
mutant.back() = level;
}

for (; levit != levtok.end(); levit++) {
tokeniser genetok(*levit, genesep);
for (tokeniser::iterator genit = genetok.begin();
genit != genetok.end(); genit++) {
gene = *genit;
gene.insert(m1 + 1, "0");
mutgene.push_back(new string);
mutgene.back() = gene;
}
level = join(mutgene, "}{");
level.insert(level.begin(), '{');
level.insert(level.end(), '}');
mutant.push_back(new string);
mutant.back() = level;
mutgene.release();
}

//cout << "Updated upstream nodes" << endl;

if (levelcost == 1 && resources == 0 && maintenance == 1) {
if (minchangelev > 0) {
(main.back())[m1 + 1] = pow(main.back()[m1+1],(double)metscale);
}
}

pop.back() = join(mutant, "][");
(pop.back()).insert((pop.back()).begin(), '[');
(pop.back()).insert((pop.back()).end(), ']');

copy.push_back(new double (1) );

if (ages == 1) {
age.push_back(new ptr_vector<double>);
itage = age.begin();
advance(itage,h);
}
```

```
itc = copy.begin();
advance(itc, (int)h);

double speccop;
speccop = *itc;

copy.back() = (int) ceil((0.05 * speccop));
(*itc) = (int) floor((0.95 * speccop));

if (resources == 1) {
double addresources, adddeath;
addresources = ceil((0.05 * speccop));
adddeath = addresources / (double) newnumvert;
int wholedeath = ceil(adddeath);
(*itc) -= wholedeath;
balls += ceil((wholedeath - adddeath) * newnumvert);
}

if (ages == 1) {
for (int cit = 0; cit < (speccop - (*itc)); cit++) {
(*itage).release((*itage).begin());
}

for (int cit2 = 0; cit2 < copy.back(); cit2++) {
(age.back()).push_back(new double (0));
}
}

mutant.release();
mutgene.release();

//cout << "COPY NODE AND SUBGRAPH" << endl;
}

/* Copy the node but not subgraph */
else {

/* Create a copy of the chosen node in the new species and generate new subgraph.*/
char_separator<char> sep("[{}]");
tokeniser tokens(pop.back(), sep);

tokeniser::iterator node = tokens.begin();
node = tokens.begin();
advance(node, m1);

string copiednode = *node;
double flip;

ptr_list<string> newsub;
string holding;
ptr_vector<int> npl;

if (copiednode != "0") {
/* Number of nodes per level */
char_separator<char> levsep("[]");
tokeniser levsepar(pop.back(), levsep);

tokeniser::iterator nodelev = levsepar.begin();
int viablelevs = nodelevelcount(pop, m1);
viablelevs -= 2;
advance(nodelev, (viablelevs + 1));

for (tokeniser::iterator levcount = levsepar.begin();
levcount != nodelev; levcount++) {
holding = (*levcount);
npl.push_back(new int (1));
npl.back() = 0;
for (string_find_iterator It = make_find_iterator(holding,
first_finder("{", is_iequal()));
It != string_find_iterator(); ++It) {
npl.back() += 1;
}
}
for (ptr_vector<int>::iterator itz = npl.begin();
itz != npl.end(); itz++) {
newsub.push_back(new string);
for (int zz = 0; zz < (*itz); zz++) {
flip = rn();
if (flip > beta) {
(newsub.back()).push_back('0');
} else {
(newsub.back()).push_back('1');
}
}
}

/* Choose node on level l-1 and ensure that connection is present */

/* Choose node on level l-1 and ensure that connection is present */
double p5 = rn();
int chosennode = floor(p5 * npl.back());

ptr_vector<int> nplcopy;

nplcopy = npl;
nplcopy.pop_back();

int advancstr = accumulate(nplcopy.begin(), nplcopy.end(), (double) 0);

chosennode += advancstr;

if (npl.size() != newsub.size()) {
cout << "ERROR IN MUTATION" << endl;
}

string wholenewsub;

for (ptr_list<string>::iterator iterat = newsub.begin();
iterat != newsub.end(); iterat++) {
wholenewsub.append(*iterat);
}

wholenewsub[chosennode] = '1';

copiednode = wholenewsub;

}

copiednode.insert(copiednode.begin(), '{');
copiednode.insert(copiednode.begin(), '}');
copiednode.push_back('}');

replace_nth(pop.back(), "}", m1, copiednode);

/* Update ability and maintenance */
if (resources == 0) {
if (m1 + 1 < (ab.back()).size()) {
ptr_vector<double>::iterator abit;
abit = (ab.back()).begin();
advance(abit, m1 + 1);
(ab.back()).insert(abit, new double (1) );
if (copiednode != "}{0}") {
for (int a = 2; a < (copiednode.length() - 1); a++) {
if (copiednode[a] == '1') {
(ab.back())[m1 + 1] += (ab.back())[a - 2];
}
}
} else {
(ab.back())[m1 + 1] = normal_gen(eng, normdist);
if ((ab.back())[m1 + 1] < 0) {
(ab.back())[m1 + 1] = 0 - (ab.back())[m1 + 1];
}
}
```

```
} else {
(ab.back()).push_back(new double (1) );
for (int a = 2; a < (copiednode.length() - 1); a++) {
if (copiednode[a] == '1') {
(ab.back()).back() += (ab.back())[a - 2];
}
}
}

if (maintenance == 1) {
if (m1 < (main.back()).size()) {
ptr_vector<double>::iterator mainit;
mainit = (main.back()).begin();
advance(mainit, m1 + 1);
(main.back()).insert(mainit, new double (1) );
if (copiednode != "}{0}") {
for (int a = 2; a < (copiednode.length() - 1);
a++) {
if (copiednode[a] == '1') {
(main.back())[m1 + 1] +=
(main.back())[a - 2];
}
}
} else {
(main.back())[m1 + 1] = normal_gen(eng, normdist);
if ((main.back())[m1 + 1] < 0) {
(main.back())[m1 + 1] = 0
- (main.back())[m1 + 1];
}
}
} else {
(main.back()).push_back(new double (1) );
for (int a = 2; a < (copiednode.length() - 1); a++) {
if (copiednode[a] == '1') {
(main.back()).back() += (ab.back())[a - 2];
}
}
}

}
}

/* Update upstream nodes */

char_separator<char> levsep("[]");
char_separator<char> genesep("{}");
tokeniser levtok(pop.back(), levsep);
tokeniser::iterator levit = levtok.begin();

int minchangelev;

minchangelev = nodelevelcount(pop, m1);

advance(levit, minchangelev);
string level;
string gene;
ptr_list<string> mutant;
ptr_list<string> mutgene;

for (tokeniser::iterator lev2it = levtok.begin(); lev2it != levit;
lev2it++) {
level = *lev2it;
mutant.push_back(new string);
mutant.back() = level;
}

for (; levit != levtok.end(); levit++) {
tokeniser genetok(*levit, genesep);
for (tokeniser::iterator genit = genetok.begin();
genit != genetok.end(); genit++) {
gene = *genit;
gene.insert(m1 + 1, "0");
```

```
mutgene.push_back(new string);
mutgene.back() = gene;
}
level = join(mutgene, "}{");
level.insert(level.begin(), '{');
level.insert(level.end(), '}');
mutant.push_back(new string);
mutant.back() = level;
mutgene.release();
}

if (levelcost == 1 && resources == 0 && maintenance == 1) {
if (minchangelev > 0) {
(main.back())[m1 + 1] = pow(main.back()[m1+1],(double)metscale);
}
}

pop.back() = join(mutant, "][");
(pop.back()).insert((pop.back()).begin(), '[');
(pop.back()).insert((pop.back()).end(), ']');

mutant.release();
mutgene.release();
newsub.release();
npl.release();

copy.push_back(new double (1) );
itc = copy.begin();
advance(itc, (int)h);
if (ages == 1) {
age.push_back(new ptr_vector<double>);
itage = age.begin();
advance(itage,h);
}
double speccop;
speccop = *itc;
copy.back() = ceil((0.05 * speccop));
(*itc) = floor((0.95 * speccop));

if (resources == 1) {
double addresources, adddeath;
addresources = ceil((0.05 * speccop));
adddeath = addresources / newnumvert;
int wholedeath = ceil(adddeath);
(*itc) -= wholedeath;
balls += ceil((wholedeath - adddeath) * newnumvert);
}

if (ages == 1) {
for (int cit = 0; cit < (speccop - (*itc)); cit++) {
(*itage).release((*itage).begin());
}
for (int cit2 = 0; cit2 < copy.back(); cit2++) {
(age.back()).push_back(new double (0));
}
}
//cout << "COPY NODE, NEW SUBGRAPH" << endl;
}

}

/* New node created */
else {
//cout << "NEW NODE " << endl;

/* Number of nodes per level */
char_separator<char> levsep("[]");
tokeniser levsepar(pop.back(), levsep); // Separate "parent" into 1 level per token

tokeniser::iterator nodelev = levsepar.begin();
string holding;
ptr_vector<int> npl;
```

```
ptr_list<string> newsub;
double flip = 0;

for (tokeniser::iterator levcount = levsepar.begin();
levcount != levsepar.end(); levcount++) {
holding = (*levcount);
npl.push_back(new int (1));
newsub.push_back(new string);
npl.back() = 0;
for (string_find_iterator It = make_find_iterator(holding,
first_finder("{", is_iequal()));
It != string_find_iterator(); ++It) {
npl.back() += 1; // Find the nodes per level
flip = rn();
if (flip > beta) {
(newsub.back()).push_back('0');
} else {
(newsub.back()).push_back('1');
}
}
}

/* Choose node on level l-1 and ensure that connection is present */
double p5 = rn();
int chosennode = floor(p5 * npl.back());

ptr_vector<int> nplcopy;

nplcopy = npl;
nplcopy.pop_back();

int advancstr = accumulate(nplcopy.begin(), nplcopy.end(), (double) 0);

chosennode += advancstr;

if (npl.size() != newsub.size()) {
cout << "ERROR IN MUTATION" << endl;
}

string mutantgene;

for (ptr_list<string>::iterator iterat = newsub.begin();
iterat != newsub.end(); iterat++) {
mutantgene.append(*iterat);
}

mutantgene[chosennode] = '1';

/* Update ability and maintenance */
if (resources == 0) {
(ab.back()).push_back(new double (1) );
(ab.back()).back() = 0;
for (unsigned int a = 0; a < mutantgene.length(); a++) {
if (mutantgene[a] == '1') {
(ab.back()).back() += (ab.back())[a];
}
}

if (maintenance == 1) {
(main.back()).push_back(new double (1) );
(main.back()).back() = 0;
for (unsigned int a = 0; a < mutantgene.length(); a++) {
if (mutantgene[a] == '1') {
(main.back()).back() += (main.back())[a];
}
}
}
}

/* Insert node into genotype */
mutantgene.insert(mutantgene.begin(), '{');
mutantgene.insert(mutantgene.begin(), '[');
mutantgene.append("}]");
(pop.back()).append(mutantgene);

newsub.release();
npl.release();

copy.push_back(new double (1) );
itc = copy.begin();
advance(itc, (int)h);
double speccop;
speccop = *itc;
copy.back() = ceil((0.05 * speccop));
(*itc) = floor((0.95 * speccop));

if (ages == 1) {
age.push_back(new ptr_vector<double>);
itage = age.begin();
advance(itage,h);
}

if (resources == 1) {
int newnumvert = 0;

for (string_find_iterator It = make_find_iterator((*itspec),
first_finder("{", is_iequal()));
It != string_find_iterator(); ++It) {
newnumvert += 1;
}

double addresources, adddeath;
addresources = ceil((0.05 * speccop));
adddeath = addresources / (double)newnumvert;
int wholedeath = ceil(adddeath);
(*itc) -= wholedeath;
balls += ceil((wholedeath - adddeath) * newnumvert);
}

if (ages == 1) {
for (int cit = 0; cit < (speccop - (*itc)); cit++) {
(*itage).release((*itage).begin());
}
for (int cit2 = 0; cit2 < copy.back(); cit2++) {
(age.back()).push_back(new double (0));
}
}

}
}

/* Population normalisation */

void popnorm(Population &pop) {
Population::iterator itpop;

if (peakselect == 0) {
double abpopsum = 0;
double mainpopsum = 0;

/* Compute average ability score of species */
for (itpop = pop.begin(); itpop != pop.end(); itpop++) {
abpopsum += (*itpop)[graph_bundle].abtotal;

if (maintenance == 1) {
mainpopsum += (*itpop)[graph_bundle].maintotal;
}
}

abpopsum = abpopsum / (double)pop.size();
if (maintenance == 1) {
mainpopsum = mainpopsum / (double)pop.size();
}
```

```
double acheck, mcheck;

for (itpop = pop.begin(); itpop != pop.end(); itpop++) {
(*itpop)[graph_bundle].abtotal = (*itpop)[graph_bundle].abtotal
/ abpopsum;

if (maintenance == 1) {
(*itpop)[graph_bundle].maintotal = (*itpop)[graph_bundle].maintotal
/ mainpopsum;
}
acheck = 0;
mcheck = 0;
for (unsigned int z = 0; z < num_vertices(*itpop); z++) {
(*itpop)[z].ability = (*itpop)[z].ability / abpopsum;
acheck += (*itpop)[z].ability;

if (maintenance == 1) {
(*itpop)[z].maintain = (*itpop)[z].maintain / mainpopsum;
mcheck += (*itpop)[z].maintain;
}
}

if ((*itpop)[graph_bundle].abtotal - acheck > 1e-005
|| (*itpop)[graph_bundle].maintotal - mcheck > 1e-005) {
cout << "ERROR!" << endl;
cout << (*itpop)[graph_bundle].abtotal - acheck << endl;
cout << (*itpop)[graph_bundle].maintotal - mcheck << endl;
}
}
}
}

void fitnessfunct(Population &pop, int y, ptr_vector<int> &survivorlength) {
Population::iterator itpop;
double x = 0;
double fitav = 0;

if (multipeak == 0) {
/* Calculate fitness */
for (itpop = pop.begin(); itpop != pop.end(); itpop++) {
if (maintenance == 0) {
x = (*itpop)[graph_bundle].abtotal - 5;
} else {
x = ((*itpop)[graph_bundle].abtotal / (*itpop)[graph_bundle].maintotal) - 5;
}

x *= x;

(*itpop)[graph_bundle].fittotal = (25 - x);
if ((*itpop)[graph_bundle].fittotal < 0) {
(*itpop)[graph_bundle].fittotal = 0;
}

fitav += (*itpop)[graph_bundle].fittotal;

(*itpop)[graph_bundle].normfit = (*itpop)[graph_bundle].fittotal;
}

/* Normalise */
if (norm == 1) {
fitav /= pop.size();

if (fitav != 0) {
for (itpop = pop.begin(); itpop != pop.end(); itpop++) {
(*itpop)[graph_bundle].normfit /= fitav;
}
}
}

/* Selection */
double randdeath = 0;
int tmax = pop.size();
int count = 1;
double deathlimit = 0;

for (int t = 0; t< tmax;) {
itpop = pop.begin();
advance(itpop, t);

randdeath = rn();
if (norm == 0) {
deathlimit = ((*itpop)[graph_bundle].normfit/(double)25) - 1;
} else {
deathlimit = (1 - (*itpop)[graph_bundle].normfit)/2;
}

if (deathlimit < 0) {
deathlimit = 0 - deathlimit;
}

if (randdeath < deathlimit) {
survivorlength.push_back(new int (1));
survivorlength.back() = y - (*itpop)[graph_bundle].tstart;
pop.release(itpop);
} else {
t++;
}

if (count == tmax) {
t = tmax;
}
++count;

}
} else {

double averagedegree = 0;
double nodesperlevel = 0;

/* Calculate fitness */
for (itpop = pop.begin(); itpop != pop.end(); itpop++) {
averagedegree = (double)num_edges(*itpop) / (double)num_vertices(*itpop);
nodesperlevel = (double)num_vertices(*itpop) / ((double)(*itpop)[graph_bundle].maxlevel + 1);

(*itpop)[graph_bundle].fittotal = sin(averagedegree + sqrt((double)2)) + sin(nodesperlevel + sqrt((double) 3)) + 2;

fitav += (*itpop)[graph_bundle].fittotal;

(*itpop)[graph_bundle].normfit = (*itpop)[graph_bundle].fittotal;
}

/* Normalise */
if (norm == 1) {
fitav /= pop.size();

if (fitav != 0) {
for (itpop = pop.begin(); itpop != pop.end(); itpop++) {
(*itpop)[graph_bundle].normfit /= fitav;
}
}
}

/* Selection */
double randdeath = 0;
int tmax = pop.size();
int count = 1;
double deathlimit = 0;

for (int t = 0; t< tmax;) {
itpop = pop.begin();
advance(itpop, t);

randdeath = rn();
if (norm == 0) {
```

```
deathlimit = ((*itpop)[graph_bundle].normfit/(double)25) - 1;
} else {
deathlimit = (1 - (*itpop)[graph_bundle].normfit)/2;
}

if (deathlimit < 0) {
deathlimit = 0 - deathlimit;
}

if (randdeath < deathlimit) {
survivorlength.push_back(new int (1));
survivorlength.back() = y - (*itpop)[graph_bundle].tstart;
pop.release(itpop);
} else {
t++;
}

if (count == tmax) {
t = tmax;
}
++count;

}
}

}

void popnorm(Genopop &pop, doublelists &ab, doublelists &main) {

doublelists::iterator ita;
doublelists::iterator itm;

ptr_vector<double> avspecsab, avspecmain;

double abpopsum = 0;
double mainpopsum = 0;

if (maintenance == 1) {
itm = main.begin();
}

/* Compute ability score of species */
for (ita = ab.begin(); ita != ab.end(); ita++) {
abpopsum = std::accumulate((*ita).begin(), (*ita).end(),
(double) abpopsum);

if (maintenance == 1) {
mainpopsum = std::accumulate((*itm).begin(), (*itm).end(),
mainpopsum);
itm++;
}
}

abpopsum = abpopsum / (double)pop.size();
if (maintenance == 1) {
mainpopsum = mainpopsum / (double)pop.size();
}

double acheck, mcheck;
double abtotal = 0;
double maintotal = 0;

if (maintenance == 1) {
itm = main.begin();
}

/* Normalising scores of species */
for (ita = ab.begin(); ita != ab.end(); ita++) {

abtotal = std::accumulate((*ita).begin(), (*ita).end(), (double) 0)
/ abpopsum;
```

```
if (maintenance == 1) {
maintotal = std::accumulate((*itm).begin(), (*itm).end(),
(double) 0) / mainpopsum;
}

acheck = 0;
mcheck = 0;

for (unsigned int z = 0; z < (*ita).size(); z++) {
(*ita)[z] = (*ita)[z] / abpopsum;
acheck += (*ita)[z];

if (maintenance == 1) {
(*itm)[z] = (*itm)[z] / mainpopsum;
mcheck += (*itm)[z];
}
}

if (abtotal - acheck > 1e-005 || maintotal - mcheck > 1e-005) {
cout << "ERROR!" << endl;
cout << abtotal - acheck << endl;
cout << maintotal - mcheck << endl;
}

if (maintenance == 1) {
itm++;
}

}

}

void degcalc(Genopop &pop, ptr_vector<double> &deg, ptr_vector<double> &degdis,
int y, ptr_vector<double> &counter, ptr_vector<double> &copynums) {

Genopop::iterator itpop;
ptr_vector<double>::iterator itcop;
double avdegspec = 0;
double avdeg = 0;
double numedges = 0;
double numvertices = 0;
itcop = copynums.begin();

for (itpop = pop.begin(); itpop != pop.end(); itpop++) {
avdegspec = 0;
numedges = 0;
numvertices = 0;
for (string_find_iterator It = make_find_iterator((*itpop),
first_finder("1", is_iequal())); It != string_find_iterator();
++It) {
numedges += 1;
}
for (string_find_iterator It = make_find_iterator((*itpop),
first_finder("{", is_iequal())); It != string_find_iterator();
++It) {
numvertices += 1;
}

if (resources == 1 && dynamics == 1) {
avdegspec = (numedges / numvertices) * (double)(*itcop);
} else {
avdegspec = (numedges / numvertices);
}

avdeg += avdegspec;

//degdis.push_back(new double (1) );
//degdis.back() = avdegspec;
}

if (resources == 1 && dynamics == 1) {
avdeg /= accumulate(copynums.begin(), copynums.end(), (double) 0);
```

```
} else {
avdeg /= (double)pop.size();
}

deg[y] += avdeg;
counter[y] += 1;
}

int resourcealloc(Genopop &pop, ptr_vector<double> &copynums, int &balls, doublelists &age) {
Genopop::iterator its;
ptr_vector<double>::iterator itc;
doublelists::iterator itage;

double deathprob;
int tmax = pop.size();
int counter;
int count = 1;
ptr_vector<int> numvert;

if (ages == 1) {
for (itage = age.begin(); itage != age.end(); itage++) {
sort((*itage).begin(), (*itage).end());
reverse((*itage).begin(), (*itage).end());
}
}

/* Death step with probability r */

for (int t = 0; t < tmax;) {
itc = copynums.begin();
its = pop.begin();
itage = age.begin();
advance(its, t);
advance(itc, t);
advance(itage, t);
//cout << (*itage).size() << endl;

counter = *itc;

numvert.push_back(new int (1));
numvert.back() = 0;

for (string_find_iterator It = make_find_iterator((*its),
first_finder("{", is_iequal())); It != string_find_iterator();
++It) {
numvert.back() += 1;
}


if (ages == 0) {
int numedge = 0;
for (string_find_iterator Itz = make_find_iterator((*its),
first_finder("1", is_iequal())); Itz != string_find_iterator();
++Itz) {
numedge += 1;
}

int numlevels = 0;
for (string_find_iterator It = make_find_iterator((*its),
first_finder("[", is_iequal())); It != string_find_iterator();
++It) {
numlevels += 1;
}
numlevels -= 1; // Make lowest level a level 0.

/* Exponetial death rate */
double param = 0;
param = (double) numlevels / ((double)numedge + (double)numvert.back());
double deathrate5 = 1
- (1 / exp(param));
for (int z = 0; z < counter; z++) {
deathprob = rn();
```

```
if (deathprob < deathrate5) {
*itc -= 1;
}
}

/* Fixed death rate
for (int z = 0; z < counter; z++) {
deathprob = rn();
if (deathprob < deathrate) {
*itc -= 1;
}
}
*/

/* Death rate dependent on number of nodes
double deathrate2 = 1 / numvert.back();
for (int z = 0; z < counter; z++) {
deathprob = rn();
if (deathprob < deathrate2) {
*itc -= 1;
}
}
*/

/* Death rate dependent on number of levels
double deathrate3 = 1 / (numlevels + numvert.back());
for (int z = 0; z < counter; z++) {
deathprob = rn();
if (deathprob < deathrate3) {
*itc -= 1;
}
}
*/

/* Death rate dependent on both levels and nodes
int numlevels = 0;
for (string_find_iterator It = make_find_iterator((*its),
first_finder("[", is_iequal())); It != string_find_iterator();
++It) {
numlevels += 1;
}

double deathrate4 = 1/sqrt((numlevels*numlevels)+(numvert.back()*numvert.back()));
for (int z = 0; z < counter; z++) {
deathprob = rn();
if (deathprob < deathrate4) {
*itc -= 1;
}
}
*/
} else {
int countmax = counter;
int agecount = 1;
if ((*itage).size() != countmax) {
cout << "ERROR" << endl;
cout << (*itage).size() << endl;
}

for (int z = 0; z < countmax; ) {
deathprob = rn();
if (deathprob < ((*itage)[z]*rn())) {
*itc -= 1;
(*itage).erase((*itage).begin() + z);
} else {
z++;
}
if (agecount == countmax) {
z = countmax;
}

++agecount;
}
```

```
}

if (*itc < 1) {
balls += (counter * numvert.back());
pop.release(its);
copynums.release(itc);
numvert.pop_back();
age.release(itage);
} else {
balls += ((counter - (*itc)) * numvert.back());
++t;
}

if (count == tmax) {
t = tmax;
}

++count;
}

/* Maintenance step */
ptr_vector<int> permvec;
itc = copynums.begin();
int speccounter = 0;

for (its = pop.begin(); its != pop.end(); its++) {
for (int z = 1; z < (*itc) + 1; z++) {
permvec.push_back(new int (1));
permvec.back() = speccounter;
}
speccounter++;
itc++;
}

random_shuffle(permvec.begin(), permvec.end());

int mainballs = 0;
mainballs = balls;
int tracker = permvec.size();

for (int z = 0; z < tracker; z++) {
if (numvert[permvec[z]] > balls) {
if (ages == 1) {
itage = age.begin();
advance(itage, permvec[z]);
(*itage).release((*itage).begin());
}
copynums[permvec[z]] -= 1;
mainballs += numvert[permvec[z]];
permvec.erase(permvec.begin() + z);
tracker -= 1;
z -= 1;
} else {
balls -= numvert[permvec[z]];
}
}

mainballs -= balls;

/* Removing any species with 0 copynumber */
int countmax = pop.size();
int bcount = 0;
for (int c = 0; c < countmax; c++) {
itc = copynums.begin();
its = pop.begin();
itage = age.begin();
advance(itc, c);
advance(its, c);
advance(itage, c);

if ((*itc) == 0) {
copynums.release(itc);
```

```
pop.release(its);
age.release(itage);
for (unsigned int u = 0; u < permvec.size(); u++) {
if (permvec[u] > c) {
permvec[u] -= 1;
}
}
}

if (bcount == countmax) {
c = countmax;
}

++bcount;
}

/* Update ages of surviving individuals */
if (ages == 1) {
for (itage = age.begin(); itage != age.end(); itage++) {
for (int u = 0; u < (*itage).size(); u++) {
(*itage)[u] += 1;
}
}

}

/* Reproduction step */
if (balls > 0) {
tracker = permvec.size();

for (int z = 0; z < tracker; z++) {
itage = age.begin();
if (numvert[permvec[z]] <= balls) {
balls -= numvert[permvec[z]];
copynums[permvec[z]] += 1;
if (ages == 1) {
advance(itage, permvec[z]);
(*itage).push_back(new double (0));
}
}
}
}

balls += mainballs;

/* Check for ball conservation */
int totalballs = 0;

for (unsigned int t = 0; t < pop.size(); t++) {
totalballs += copynums[t] * numvert[t];
}

totalballs += balls;

if (totalballs != 1001) {
cout << "BALLS NOT CONSERVED" << endl;
cout << "total balls is " << totalballs << endl;
}

return balls;
}

void resourcealloc(Population &pop, ptr_vector< ptr_vector<int> > &balls, int y, ptr_vector<int> &survivorlength) {
Population::iterator its;

double deathprob;
int tmax = pop.size();
int counter;
int dcount = 1;
int ballsmax;
int balldiff;
int maxlevel;
maxlevel = levelcount(pop);

ptr_vector< ptr_vector<ptr_vector<int> > > listspeccolournums;
ptr_vector< ptr_vector<ptr_vector<int> > >::iterator itlist;
ptr_vector<ptr_vector<int> > speccolournums;


if (fluctuations == 1) {
const double pi = boost::math::constants::pi<double>();
ballsmax = floor((5000*sin((double)y/500)) + 6000);
balldiff = ballsmax - floor((5000*sin((double)(y-1)/500)) + 6000);
}

if (ages == 1) {
for (its = pop.begin(); its != pop.end(); its++) {
sort((*its)[graph_bundle].age.begin(), (*its)[graph_bundle].age.end());
reverse((*its)[graph_bundle].age.begin(), (*its)[graph_bundle].age.end());
}
}

/* Death step with probability r */
for (int t = 0; t < tmax;) {

its = pop.begin();
advance(its, t);

counter = (*its)[graph_bundle].copyno;

// Counting colours per species
if (breakup == 0) {
listspeccolournums.push_back(new ptr_vector< ptr_vector<int> >); // List item for each species

for (unsigned int maxcol = 0; maxcol < balls.size(); maxcol++) {
(listspeccolournums.back()).push_back(new ptr_vector<int>);

for (unsigned int elev = 0; elev < ((*its)[graph_bundle].maxlevel + 1); elev++) {
((listspeccolournums.back()).back()).push_back(new int (1));
((listspeccolournums.back()).back()).back() = 0;
}
}

for (unsigned int evert = 0; evert < num_vertices(*its); evert++) {
(listspeccolournums.back())[(*its)[evert].colour][(*its)[evert].level] += 1;
}

itlist = listspeccolournums.begin();
advance(itlist,t);
if (accumulate((*itlist)[0].begin(),(*itlist)[0].end(),(double)0) != num_vertices(*its)) {
cout << accumulate((*itlist)[0].begin(),(*itlist)[0].end(),(double)0) << endl;
cout << num_vertices(*its) << endl;
cout << "ERROR!" << endl;
}
}
else {
speccolournums.push_back(new ptr_vector<int>);
for (unsigned int bl = 0; bl < balls.size(); bl++) {
(speccolournums.back()).push_back(new int (1));
(speccolournums.back()).back() = 0;
}
for (unsigned int bc = 0; bc < num_vertices(*its); bc++) {
(speccolournums.back())[(*its)[bc].colour] += 1;
}

if (accumulate(speccolournums.back().begin(), speccolournums.back().end(), (double) 0) != num_vertices(*its)) {
cout << "ERROR!" << endl;
}

}

if (ages == 0) {
/* Exponetial death rate */
```

```
/*double param = 0;
param = (double)(*its)[graph_bundle].maxlevel
/ (double)(num_edges(*its)
+ num_vertices(*its));
double deathrate5 = 1 - (1 / exp(param));
for (int z = 0; z < counter; z++) {
deathprob = rn();
if (deathprob < deathrate5) {
(*its)[graph_bundle].copyno -= 1;
}
}
*/
/* Fixed death rate
for (int z = 0; z < counter; z++) {
deathprob = rn();
if (deathprob < deathrate) {
(*its)[graph_bundle].copyno -= 1;
}
}
*/

//Death rate dependent on number of nodes
/* double deathrate2 = 1 / exp((double)num_vertices(*its));
for (int z = 0; z < counter; z++) {
deathprob = rn();
if (deathprob < deathrate2) {
(*its)[graph_bundle].copyno -= 1;
}
}


/* Death rate dependent on number of levels
double deathrate3 = 1 / ((*its)[graph_bundle].maxlevel + num_vertices(*its));
for (int z = 0; z < counter; z++) {
deathprob = rn();
if (deathprob < deathrate3) {
(*its)[graph_bundle].copyno -= 1;
}
}
*/

/* Death rate dependent on both levels and nodes
double deathrate4 = 1/sqrt((*its)[graph_bundle].maxlevel*(*its)[graph_bundle].maxlevel)+(num_vertices(*its)*num_vertices(*its)));
for (int z = 0; z < counter; z++) {
deathprob = rn();
if (deathprob < deathrate4) {
(*its)[graph_bundle].copyno -= 1;
}
}
*/
} else {
int countmax = counter;
ptr_vector<int>::iterator itage;
int agecount = 1;
if ((*its)[graph_bundle].age.size() != countmax) {
cout << "ERROR" << endl;
cout << (*its)[graph_bundle].age.size() << endl;
}

for (int z = 0; z < countmax; ) {
deathprob = rn();
if (deathprob < (1 - (1/(((*its)[graph_bundle].age[z] +1))))) {
(*its)[graph_bundle].copyno -= 1;
itage = (*its)[graph_bundle].age.begin();
advance(itage,z);
(*its)[graph_bundle].age.release(itage);
} else {
z++;
}
if (agecount == countmax) {
z = countmax;
}
```

```
++agecount;
}
}

if (breakup == 1) {
if ((*its)[graph_bundle].copyno < 2) {
if ((*its)[graph_bundle].copyno < 0) {
(*its)[graph_bundle].copyno = 0;
}

for (int evert = 0; evert < num_vertices(*its); evert++) {
balls[(*its)[evert].colour][0] += counter;
}

if (ages == 1) {
(*its)[graph_bundle].age.release();
}

survivorlength.push_back(new int (1));
survivorlength.back() = y - (*its)[graph_bundle].tstart;
pop.release(its);
} else {
for (int evert = 0; evert < num_vertices(*its); evert++) {
balls[(*its)[evert].colour][0] += (counter - (*its)[graph_bundle].copyno);
}
++t;
}
} else {
if ((*its)[graph_bundle].copyno < 2) {
for (int evert = 0; evert < num_vertices(*its); evert++) {
balls[(*its)[evert].colour][(*its)[evert].level] += (*its)[graph_bundle].copyno;
}

if (ages == 1) {
(*its)[graph_bundle].age.release();
}

survivorlength.push_back(new int (1));
survivorlength.back() = y - (*its)[graph_bundle].tstart;
pop.release(its);
} else {
for (int evert = 0; evert < num_vertices(*its); evert++) {
balls[(*its)[evert].colour][(*its)[evert].level] +=  (counter - (*its)[graph_bundle].copyno);
}

++t;
}

}

if (dcount == tmax) {
t = tmax;
}

++dcount;
}

int totalballs = 0;

for (its = pop.begin(); its != pop.end(); its++) {
totalballs += (*its)[graph_bundle].copyno * num_vertices(*its);
}

for (unsigned int bc = 0; bc < balls.size(); bc++) {
totalballs += accumulate(balls[bc].begin(), balls[bc].end(), (double) 0);
}

if (totalballs != 1050) {
if (fluctuations == 0) {
cout << "total balls is " << totalballs << endl;
cout << "BALLS NOT CONSERVED" << endl;
```

```
}

}

//cout << totalballs << endl;
/* Maintenance step */
ptr_vector<int> permvec;
int speccounter = 0;

// Random maintenance //
/*
for (its = pop.begin(); its != pop.end(); its++) {
for (int bz = 1; bz < (*its)[graph_bundle].copyno + 1; bz++) {
permvec.push_back(new int (1));
permvec.back() = speccounter;
}
speccounter++;
}

random_shuffle(permvec.begin(), permvec.end());
*/

// Edge first maintenance //

ptr_vector<int> permvecinit;

for (its = pop.begin(); its != pop.end(); its++) {
for (int z = 1; z < (*its)[graph_bundle].copyno + 1; z++) {
permvecinit.push_back(new int (1));
permvecinit.back() = speccounter;
}
speccounter++;
}

random_shuffle(permvecinit.begin(), permvecinit.end());

int itrackmax = permvecinit.size();
int count = 1;
double randm = 0;
ptr_vector<int>::iterator itperm;

for (int t = 0; t < itrackmax;) {
its = pop.begin();
advance(its, permvecinit[t]);
randm = rn();
if (randm > 1 - (num_edges(*its)/num_vertices(*its))){
permvec.push_back(new int (1));
permvec.back() = permvecinit[t];
itperm = permvecinit.begin();
advance(itperm, t);
permvecinit.erase(itperm);
} else {
t++;
}

if (count == itrackmax) {
t = itrackmax;
}
++count;
}

for (itperm = permvecinit.begin(); itperm != permvecinit.end(); itperm++) {
permvec.push_back(new int(1));
permvec.back() = (*itperm);
}

if (itrackmax != permvec.size()) {
cout << "ERROR!" << endl;
cout << permvec.size() << endl;
}

// Edge first //
```

```
/*
ptr_vector<int> edgeordering;

for (its = pop.begin(); its != pop.end(); its++) {
edgeordering.push_back(new int(1));
edgeordering.back() = num_edges(*its);
}

int maxe = 0;
int pnum = 0;
ptr_vector<int>::iterator ite;
int tracker99 = 0;

for (unsigned int uuc = 0; uuc < pop.size(); uuc++) {
// Find maximum edge number
maxe = 0;
for (unsigned int uc = 0; uc < edgeordering.size(); uc++) {
if (edgeordering[uc] >= maxe) {
maxe = edgeordering[uc];
pnum = uc;
}
}
its = pop.begin();
advance(its,pnum);

for (int u = 1; u < ((*its)[graph_bundle].copyno + 1); u++) {
permvec.push_back(new int (1));
permvec.back() = pnum;
}

tracker99 += (*its)[graph_bundle].copyno;

if (tracker99 != permvec.size()) {
cout << "ERROR!" << endl;
}

if (pnum > edgeordering.size()) {
cout << "ERROR!" << endl;
}

ite = edgeordering.begin();
advance(ite,pnum);
edgeordering.release(ite);

if (edgeordering.size() != (pop.size() - uuc - 1)) {
cout << "ERROR!" << endl;
}

}
*/
ptr_vector< ptr_vector<int> > mainballs;

for (unsigned int mb = 0; mb < balls.size(); mb++) {
mainballs.push_back(new ptr_vector<int>);
for (unsigned int mb2 = 0; mb2 < balls[mb].size(); mb2++) {
mainballs.back().push_back(new int (0));
mainballs.back().back() = balls[mb][mb2];
}
}

int tracker = permvec.size();
int ballcheck;
int dvalue = 0;
ptr_vector<int>::iterator itperm2;

for (int z = 0; z < tracker; z++) {
its = pop.begin();
advance(its, permvec[z]);
ballcheck = 0;

if (breakup == 1) {
for (unsigned int bb = 0; bb < balls.size(); bb++) {
```

```
if (speccolournums[permvec[z]][bb] > (unsigned) balls[bb][0]) {
ballcheck = 1;
bb = balls.size();
}
}

if (ballcheck == 1) {
dvalue = permvec[z];

itperm2 = permvec.begin() + z;

if (*itperm2 != dvalue) {
cout << "ERROR!" << endl;
} else {
(*its)[graph_bundle].copyno -= 1;
if (ages == 1) {
//cout << (*its)[graph_bundle].age.size() << endl;
(*its)[graph_bundle].age.release((*its)[graph_bundle].age.begin());
}
tracker -= 1;
for (unsigned int bb = 0; bb < balls.size(); bb++) {
mainballs[bb][0] += speccolournums[permvec[z]][bb];
}

permvec.release(permvec.begin() + z);
}

if (tracker != permvec.size()) {
cout << "ERROR!" << endl;
}

for (int g = z; g < tracker;) {
if (permvec[g] == dvalue) {
if ((*its)[graph_bundle].copyno == 0) {
permvec.release(permvec.begin() + g); // As future individuals of species z will not be able to maintain themselves either
tracker -= 1;
for (unsigned int bb2 = 0; bb2 < balls.size(); bb2++) {
mainballs[bb2][0] += speccolournums[dvalue][bb2];
}
} else {
permvec.release(permvec.begin() + g); // As future individuals of species z will not be able to maintain themselves either
tracker -= 1;
(*its)[graph_bundle].copyno -= 1;
if (ages == 1) {
//cout << (*its)[graph_bundle].copyno << endl;
//cout << (*its)[graph_bundle].age.size() << endl;
(*its)[graph_bundle].age.release((*its)[graph_bundle].age.begin());
}
for (unsigned int bb2 = 0; bb2 < balls.size(); bb2++) {
mainballs[bb2][0] += speccolournums[dvalue][bb2];
}
}
} else {
g++;
}
}
} else {
for (unsigned int bb = 0; bb < balls.size(); bb++) {
balls[bb][0] -= speccolournums[permvec[z]][bb];
//cout << balls[0][0] << endl;
}
}
} else {
unsigned int bbmax = listspeccolournums[permvec[z]].size();

for (unsigned int bb = 0; bb < bbmax; bb++) { // For each colour (should only be one colour)
unsigned int bb2max = listspeccolournums[permvec[z]][bb].size();
for (unsigned int bb2 = 0; bb2 < bb2max; bb2++) { // For each level
if (listspeccolournums[permvec[z]][bb][bb2] > (unsigned) balls[bb][bb2]) {
ballcheck = 1;
bb2 = bb2max;
bb = bbmax;
}
}
}

if (ballcheck == 1) {

(*its)[graph_bundle].copyno -= 1;
if (ages == 1) {
(*its)[graph_bundle].age.release((*its)[graph_bundle].age.begin());
}

for (int evert2 = 0; evert2 < num_vertices(*its); evert2++) {
mainballs[(*its)[evert2].colour][(*its)[evert2].level] += 1;
}

for (int g = (z+1); g < tracker;) {
if (permvec[g] == permvec[z]) {
permvec.erase(permvec.begin() + g); // As future individuals of species z will not be able to maintain themselves either
tracker -= 1;
(*its)[graph_bundle].copyno -= 1;
if (ages == 1) {
(*its)[graph_bundle].age.release((*its)[graph_bundle].age.begin());
}
for (int evert2 = 0; evert2 < num_vertices(*its); evert2++) {
mainballs[(*its)[evert2].colour][(*its)[evert2].level] += 1;
}
} else {
g++;
}
}
} else {
for (int evert2 = 0; evert2 < num_vertices(*its); evert2++) {
balls[(*its)[evert2].colour][(*its)[evert2].level] -= 1;
}
}

}

}

if (breakup == 1) {
for (unsigned int bb = 0; bb < balls.size(); bb++) {
mainballs[bb][0] -= balls[bb][0];
}
} else {
for (unsigned int bb = 0; bb < balls.size(); bb++) {
for (unsigned int bb2 = 0; bb2 < balls[bb].size(); bb2++) {
mainballs[bb][bb2] -= balls[bb][bb2];
}
}
}

totalballs = 0;

for (its = pop.begin(); its != pop.end(); its++) {
totalballs += (*its)[graph_bundle].copyno * num_vertices(*its);
}

for (unsigned int bc = 0; bc < balls.size(); bc++) {
totalballs += accumulate(balls[bc].begin(), balls[bc].end(), (double) 0);
totalballs += accumulate(mainballs[bc].begin(), mainballs[bc].end(), (double) 0);
}

if (totalballs != 1050) {
if (fluctuations == 0) {
cout << "total balls is " << totalballs << endl;
cout << "BALLS NOT CONSERVED" << endl;
}

}
```

```
/* Removing any species with less than 2 copynumber */
int countmax = pop.size();
int bcount = 1;

if (breakup == 1) {
ptr_vector<ptr_vector<int> >::iterator itbv;
for (int c = 0; c < countmax;) {
its = pop.begin();
advance(its, c);
itbv = speccolournums.begin();
advance(itbv, c);

if ((*its)[graph_bundle].copyno < 2) {
for (int evert = 0; evert < num_vertices(*its); evert++) {
mainballs[(*its)[evert].colour][0] += (*its)[graph_bundle].copyno;
}

if (ages == 1) {
(*its)[graph_bundle].age.release();
}
survivorlength.push_back(new int (1));
survivorlength.back() = y - (*its)[graph_bundle].tstart;

pop.release(its);
speccolournums.erase(itbv);
for (unsigned int u = 0; u < permvec.size(); u++) {
if (permvec[u] > c) {
permvec[u] -= 1;
}
}
} else {
++c;
}

if (bcount == countmax) {
c = countmax;
}

++bcount;
}
} else {
ptr_vector< ptr_vector<ptr_vector<int> > >::iterator itbv;
for (int c = 0; c < countmax;) {
its = pop.begin();
advance(its, c);
itbv = listspeccolournums.begin();
advance(itbv, c);

if ((*its)[graph_bundle].copyno < 2) {
for (int evert = 0; evert < num_vertices(*its); evert++) {
balls[(*its)[evert].colour][(*its)[evert].level] += (*its)[graph_bundle].copyno;
}

if (ages == 1) {
(*its)[graph_bundle].age.release();
}
survivorlength.push_back(new int (1));
survivorlength.back() = y - (*its)[graph_bundle].tstart;

pop.release(its);
listspeccolournums.erase(itbv);
for (unsigned int u = 0; u < permvec.size(); u++) {
if (permvec[u] > c) {
permvec[u] -= 1;
}
}
} else {
++c;
}

if (bcount == countmax) {
c = countmax;
```

```
}

++bcount;
}

}

if (fluctuations == 1) {
int chosen = 0;
while ((mainballs[0][0] + balldiff) < 0) {
chosen = floor(rn()*permvec.size());
its = pop.begin();
advance(its, permvec[chosen]);
(*its)[graph_bundle].copyno -= 1;
mainballs[0][0] += speccolournums[permvec[chosen]][0];
permvec.release(permvec.begin() + chosen);
if (ages == 1) {
(*its)[graph_bundle].age.release((*its)[graph_bundle].age.begin());
}
}

mainballs[0][0] += balldiff;
}



int ballrepro = 0;
tracker = permvec.size();
int checkz = 0;

int totalballs2 = 0;

for (its = pop.begin(); its != pop.end(); its++) {
totalballs2 += (*its)[graph_bundle].copyno * num_vertices(*its);
}

for (unsigned int bc = 0; bc < balls.size(); bc++) {
totalballs2 += accumulate(balls[bc].begin(), balls[bc].end(), (double) 0);
totalballs2 += accumulate(mainballs[bc].begin(), mainballs[bc].end(), (double) 0);
}

if (totalballs2 != 1050) {
if (fluctuations == 0) {
cout << "total balls is " << totalballs2 << endl;
cout << "BALLS NOT CONSERVED" << endl;
}

}

//cout << totalballs2 << endl;

/* Reproduction step */
if (breakup == 1) {
int sumballs = 0;
for (int trub3 = 0; trub3 < balls.size(); trub3++) {
sumballs += balls[trub3].front();
}
while (sumballs > 0 && checkz < tracker) {
for (checkz = 0; checkz < tracker; checkz++) {
ballrepro = 0;
its = pop.begin();
advance(its, permvec[checkz]);

for (unsigned int bb = 0; bb < speccolournums[permvec[checkz]].size(); bb++) {
if (speccolournums[permvec[checkz]][bb] <= balls[bb][0]) {
ballrepro += 1;
}
}

if (ballrepro == speccolournums[permvec[checkz]].size()) {
for (unsigned int br = 0; br < speccolournums[permvec[checkz]].size(); br++) {
balls[br][0] -= speccolournums[permvec[checkz]][br];
}
(*its)[graph_bundle].copyno += 1;
if (ages == 1) {
(*its)[graph_bundle].age.push_back(new int (0));
//cout << (*its)[graph_bundle].age.size() << endl;
}
} else {
for (int g = checkz; g < tracker;) {
if (permvec[g] == permvec[checkz]) {
permvec.erase(permvec.begin() + g);
tracker -= 1;
} else {
g++;
}
}
}
}

sumballs = 0;
for (int trub = 0; trub < balls.size(); trub++) {
sumballs += balls[trub].front();
}

}

int totalballs3 = 0;

for (its = pop.begin(); its != pop.end(); its++) {
totalballs3 += (*its)[graph_bundle].copyno * num_vertices(*its);
}

for (unsigned int bc = 0; bc < balls.size(); bc++) {
totalballs3 += balls[bc][0];
totalballs3 += accumulate(mainballs[bc].begin(), mainballs[bc].end(), (double) 0);
}

if (totalballs3 != 1050) {
if (fluctuations == 0) {
cout << "total balls is " << totalballs3 << endl;
cout << "BALLS NOT CONSERVED" << endl;
}

}
//cout << mainballs[0][0] << endl;
for (unsigned int br = 0; br < balls.size(); br++) {
balls[br][0] += mainballs[br][0];
}
} else {
int sumballs = 0;
for (int trub4 = 0; trub4 < balls.size(); trub4++) {
sumballs += accumulate(balls[trub4].begin(), balls[trub4].end(), (double) 0);
}

while (sumballs > 0 && checkz < tracker) {
for (checkz = 0; checkz < tracker; checkz++) {
if (checkz == tracker) {
cout << "ERROR!" << endl;
}
ballrepro = 0;
its = pop.begin();
advance(its, permvec[checkz]);

if ( checkz > permvec.size()) {
cout << "ERROR!" << endl;
}

for (unsigned int bb = 0; bb < listspeccolournums[permvec[checkz]].size(); bb++) {
for (unsigned int bb2 = 0; bb2 < listspeccolournums[permvec[checkz]][bb].size(); bb2++) {
if (listspeccolournums[permvec[checkz]][bb][bb2] < (unsigned) balls[bb][bb2]) {
ballrepro += 1;
}
}
```

```
}

if (ballrepro == num_vertices(*its)) {
for (unsigned int br = 0; br < listspeccolournums[permvec[checkz]].size(); br++) {
for (unsigned int br2 = 0; br2 < listspeccolournums[permvec[checkz]][br].size(); br2++) {
balls[br][br2] -= listspeccolournums[permvec[checkz]][br][br2];
}
}
(*its)[graph_bundle].copyno += 1;
if (ages == 1) {
(*its)[graph_bundle].age.push_back(new int (0));
}
} else {
for (int g = checkz; g < tracker;) {
if (permvec[g] == permvec[checkz]) {
permvec.erase(permvec.begin() + g);
tracker -= 1;
} else {
g++;
}
}
}
}

sumballs = 0;
for (int trub = 0; trub < balls.size(); trub++) {
sumballs += accumulate(balls[trub].begin(), balls[trub].end(), (double) 0);
}

}

for (unsigned int br = 0; br < balls.size(); br++) {
for (unsigned int br2 = 0; br2 < balls[br].size(); br2++) {
balls[br][br2] += mainballs[br][br2];
}
}

}


/* Check for ball conservation */
if (breakup == 1) {
int totalballs = 0;

for (its = pop.begin(); its != pop.end(); its++) {
totalballs += (*its)[graph_bundle].copyno * num_vertices(*its);
}

for (unsigned int bc = 0; bc < balls.size(); bc++) {
totalballs += balls[bc][0];
}

if (totalballs != 1050) {
if (fluctuations == 0) {
cout << "total balls is " << totalballs << endl;
cout << "BALLS NOT CONSERVED" << endl;
}

}
} else {
int totalballs = 0;
for (its = pop.begin(); its != pop.end(); its++) {
totalballs += (*its)[graph_bundle].copyno * num_vertices(*its);
}
for (unsigned int brg = 0; brg < balls.size(); brg++) {
totalballs += accumulate(balls[brg].begin(), balls[brg].end(), (double) 0);
}

if (totalballs != 1050) {
if (fluctuations == 0) {
cout << "BALLS NOT CONSERVED" << endl;
}
```

```
cout << "total balls is " << totalballs << endl;
}
}

permvec.erase(permvec.begin(), permvec.end());
listspeccolournums.erase(listspeccolournums.begin(), listspeccolournums.end());
}

int main() {

//normal_distribution<> rdist(0.0, 0.2);
//boost::variate_generator<boost::mt19937, boost::normal_distribution<> > generator(
//boost::mt19937(time(0)), rdist);

Engine eng;
eng.seed(time(0));
Normal normdist(0.0, 0.2);

int systemsize;
double lambda;
if (analytics == 1) {
cout << "ANALYTICS == 1" << endl;
}

ptr_vector<double> sizediversity, degdiv, counter, ext, divdev2, clustercont, levs; // Used for diversity calculations.

/* Lambda*system size = mutation rate mean. */

/* Species diversity data collection */
ofstream ancestry;
ancestry.open("parents.m");

ofstream diversity;
diversity.open("div.m");

ofstream standevdiv;
standevdiv.open("devdiv.m");

ofstream standevcopy;
standevcopy.open("devcopy.m");

ofstream avnodes;
avnodes.open("nodesav.m");

ofstream avedges;
avedges.open("edgesav.m");

ofstream avlevels;
avlevels.open("levelsav.m");

ofstream sdnodes;
sdnodes.open("nodesdev.m");

ofstream sdedges;
sdedges.open("edgesdev.m");

ofstream sdlevels;
sdlevels.open("levelsdev.m");

ofstream abilities;
abilities.open("ab.m");

ofstream bodyplans;
bodyplans.open("body.dot");

ofstream copynos;
if (dynamics == 1 || resources == 1) {
copynos.open("copy.m");
}

ofstream degrees;
degrees.open("deg.m");
```

```
ofstream extinctions;
extinctions.open("ext1.m");

ofstream morphospace;
morphospace.open("morpho.m");

ofstream copytracker;
copytracker.open("ctrack.m");

//ofstream degreedistrib;
//degreedistrib.open("degdistrib.m");

ofstream genolength;
genolength.open("avgeno.m");

ofstream cluster;
cluster.open("avclustcoeff.m");

ofstream genotypes;
genotypes.open("genomes.txt");

ofstream survivals;
survivals.open("survlength.m");

ofstream nodesperlev;
nodesperlev.open("npl.m");

ofstream muttracker;
muttracker.open("mtrack.m");

//unsigned int levelmax;

morphospace << "morph = [";
if (dynamics == 1 || resources == 1) {
copynos << "vunc = [";
copytracker << "copytrack = [";
}
muttracker << "mut = [";
ancestry << "ancestors = [";
diversity << "popunc = [";
standevdiv << "divdev = [";
standevcopy << "copydev = [";
abilities << "abunc = [";
degrees << "deg1 = [";
extinctions << "ext = [";
//degreedistrib << "degdis = [";
genolength << "geno = [";
cluster << "clustercoeff = [";
survivals << "survdistrib = [";
sdnodes << "ndev = [";
sdedges << "edev = [";
sdlevels << "ldev = [";
avnodes << "nav = [";
avedges << "eav = [";
avlevels << "lav = [";
nodesperlev << "npl = [";

//ptr_vector<double> leveldiversity;
ptr_vector< ptr_vector<int> > cnumticker;

for (int si = 0; si < tsteps; si++) {
clustercont.push_back(new double (1) );
clustercont.back() = 0;
sizediversity.push_back(new double (1) );
sizediversity.back() = 0;
divdev2.push_back(new double (1) );
divdev2.back() = 0;
degdiv.push_back(new double (1) );
degdiv.back() = 0;
counter.push_back(new double (1) );
counter.back() = 0;
```

```
ext.push_back(new double (1) );
ext.back() = 0;

levs.push_back(new double (1));
levs.back() = 0;
}

ptr_vector<double> degdis;
ptr_vector<int> survivorlength;

int speciesticker;

if (genotype == 0) {
for (int x = 0; x < runs; x++) {
ptr_vector<double> copdev2;

speciesticker = 1;
systemsize = 1000;
lambda = 0.00001;
ptr_vector<ptr_vector<int>> balls;
for (int numcol = 0; numcol < totrescol; numcol++) {
balls.push_back(new ptr_vector<int>);
balls.back().push_back(new int (1));
balls.back().back() = 1050/(double)totrescol;
if (fluctuations == 1) {
balls.back().back() = 6000;
}
}

int realsize;

/* Population initialization */
Population pop;
Population::iterator itp;
double prob1 = 0;

//double dd = 0;

for (int p = 0; p < totrescol; p++) {
pop.push_back(new Bodyplan(1));
(pop.back())[graph_bundle].maxlevel = 0;
(pop.back())[graph_bundle].identity = 1;
(pop.back())[graph_bundle].ancestor = 0;
ancestry << (pop.back())[graph_bundle].ancestor << ", ";
(pop.back())[graph_bundle].copyno = floor(
(double) systemsize / (double) totrescol);
for (int tulip = 0; tulip < (pop.back())[graph_bundle].copyno; tulip++) {
(pop.back())[graph_bundle].age.push_back(new int (0));
}
if ((pop.back())[graph_bundle].age.size() != (pop.back())[graph_bundle].copyno) {
cout << "ERROR AT INITIALISATION" << endl;
}

for (unsigned int q = 0; q < num_vertices(pop.back()); q++) {
if (p == 0) {
pop.back()[q].colour = 0;
} else {
pop.back()[q].colour = 1;
}
}

balls[0][p] -= (pop.back())[graph_bundle].copyno
* num_vertices(pop.back());

if (resources == 0) {
if (mult == 0) {
(pop.back())[graph_bundle].abtotal = 0;
} else {
(pop.back())[graph_bundle].abtotal = 1;
}
if (maintenance == 1) {
(pop.back())[graph_bundle].maintotal = 0;
}

for (unsigned int r = 0; r < num_vertices(pop.back());
r++) {
(pop.back())[r].level = 0;
(pop.back())[r].ability = normal_gen(eng, normdist);
if ((pop.back())[r].ability < 0) {
(pop.back())[r].ability = 0
- (pop.back())[r].ability;
}

if (mult == 1) {
(pop.back())[r].ability += 1;
(pop.back())[graph_bundle].abtotal *=
pop.back()[r].ability;
} else {
(pop.back())[graph_bundle].abtotal +=
pop.back()[r].ability;
}

if (maintenance == 1) {
(pop.back())[r].maintain = normal_gen(eng,
normdist);
//if (rmain == 1) {
//while ((pop.back())[r].maintain > (pop.back())[r].ability) {
//(pop.back())[r].maintain = normal_gen(eng, normdist);
//}
//}
if ((pop.back())[r].maintain < 0) {
(pop.back())[r].maintain = 0
- (pop.back())[r].maintain;
}

(pop.back())[graph_bundle].maintotal += pop.back()[r].maintain;
}
}
}

}

cnumticker.push_back(new ptr_vector<int>);
for (int nmp = 0; nmp < tsteps; nmp++) {
(cnumticker.back()).push_back(new int (0));
}
(cnumticker.back()).front() = (pop.back())[graph_bundle].copyno;

(pop.back())[graph_bundle].tstart = -1;

if (norm == 1 && peakselect == 0) {
popnorm(pop);
} else {
if (analytics != 1) {
degcalc(pop, degdiv, degdis, 0, counter);
}
}

for (int y = 0; y < tsteps; y++) {

prob1 = rn();

int totalballs = 0;

Population::iterator its;

for (its = pop.begin(); its != pop.end(); its++) {
totalballs += (*its)[graph_bundle].copyno * num_vertices(*its);
}

for (unsigned int bc = 0; bc < balls.size(); bc++) {
totalballs += accumulate(balls[bc].begin(), balls[bc].end(), (double) 0);
}
```

```
if (totalballs != 1050) {
if (fluctuations == 0 && resources == 1) {
cout << "total balls is " << totalballs << endl;
cout << "BALLS NOT CONSERVED" << endl;
}

}

if (prob1 < (lambda * systemsize)
|| (dynamics == 0 && comparison == 0) || y == 0) {
mutation(pop, eng, normdist, balls, y);
if (norm == 1 && peakselect == 0 ) {
popnorm(pop);
}
speciesticker += 1;
(pop.back())[graph_bundle].identity = speciesticker;
ancestry << (pop.back())[graph_bundle].ancestor << ", ";
muttracker << y << ", ";
cnumticker.push_back(new ptr_vector<int>);
for (int nmp = 0; nmp < tsteps; nmp++) {
(cnumticker.back()).push_back(new int (0));
}
}



totalballs = 0;

for (its = pop.begin(); its != pop.end(); its++) {
totalballs += (*its)[graph_bundle].copyno * num_vertices(*its);
}

for (unsigned int bc = 0; bc < balls.size(); bc++) {
totalballs += accumulate(balls[bc].begin(), balls[bc].end(), (double) 0);
}

if (totalballs != 1050) {
if (fluctuations == 0 && resources == 1) {
cout << "total balls is " << totalballs << endl;
cout << "BALLS NOT CONSERVED" << endl;
}

}

if (ages == 1) {
for (Population::iterator its = pop.begin(); its != pop.end(); its++) {
if ((*its)[graph_bundle].age.size() != (*its)[graph_bundle].copyno) {
cout << "ERROR" << endl;
cout << (*its)[graph_bundle].age.size() << endl;
cout << (*its)[graph_bundle].copyno << endl;
}
}
}

if (analytics == 1) {
pop.pop_front();
}

degcalc(pop, degdiv, degdis, y, counter);

if (resources == 0) {
if (dynamics == 1) {
popgrowth(pop, systemsize, survivorlength, y);
} else {
if (analytics == 0 && peakselect == 0){
selection(pop, realsize, y, survivorlength);
}
}

if (peakselect == 1) {
fitnessfunct(pop,y,survivorlength);
}
Population::iterator itpop;

for (itpop = pop.begin(); itpop != pop.end();
itpop++) {
cnumticker[(*itpop)[graph_bundle].identity - 1][y] = (*itpop)[graph_bundle].copyno;
}


/*
if (dynamics == 1) {
if (x == 0) {
Population::iterator itpop;

for (itpop = pop.begin(); itpop != pop.end();
itpop++) {
copynos << (*itpop)[graph_bundle].copyno << " ";
}
copytracker << pop.size() << " ";
}
}*/
} else {
resourcealloc(pop, balls, y, survivorlength);

Population::iterator itpop;
for (itpop = pop.begin(); itpop != pop.end(); itpop++) {
copynos << (*itpop)[graph_bundle].copyno << ", ";
copdev2.push_back(new double (1));
copdev2.back() = (*itpop)[graph_bundle].copyno;
}

copytracker << pop.size() << " ";
}

if (comparison != 1) {
sizediversity[y] += pop.size();
divdev2[y] = pop.size();
} else {
sizediversity[y] += realsize;
}

totalballs = 0;

for (its = pop.begin(); its != pop.end(); its++) {
totalballs += (*its)[graph_bundle].copyno * num_vertices(*its);
}

for (unsigned int bc = 0; bc < balls.size(); bc++) {
totalballs += accumulate(balls[bc].begin(), balls[bc].end(), (double) 0);
}

if (totalballs != 1050) {
if (fluctuations == 0 && resources == 1) {
cout << "total balls is " << totalballs << endl;
cout << "BALLS NOT CONSERVED" << endl;
}

}

Population::iterator itpop;

//for (itpop = pop.begin(); itpop != pop.end(); itpop++) {
//morphospace << " " << num_vertices(*itpop) << " "
//<< num_edges(*itpop) << " "
//<< (*itpop)[graph_bundle].maxlevel << ";";
//cout << "Abtotal is " << (*itpop)[graph_bundle].abtotal << "\n";
//cout << "Maintotal is " << (*itpop)[graph_bundle].maintotal << "\n";
//cout << "Fitness is " << (*itpop)[graph_bundle].abtotal / (*itpop)[graph_bundle].maintotal << "\n";
//}

//cout << pop.size() << endl;
```

```
/*if (analytics == 0) {
for (itpop = pop.begin(); itpop != pop.end(); itpop++) {
for (unsigned int jj = 0; jj < num_vertices(*itpop);
jj++) {
degreedistrib << out_degree(jj, (*itpop)) << ", ";
}
}
}*/

/*if (pop.size() != 0) {
double cc = 0;
for (itpop = pop.begin(); itpop != pop.end(); itpop++) {
ClusteringContainer coefs(num_vertices(*itpop));
ClusteringMap cm(coefs, *itpop);
cc += all_clustering_coefficients(*itpop, cm);
}
cc /= pop.size();
clustercont[y] += cc;
}*/

if (analytics == 0) {
if (pop.size() != 0) {
double sumn = 0;
double sume = 0;
double suml = 0;

for (itpop=pop.begin(); itpop != pop.end(); itpop++) {
sumn += num_vertices(*itpop);
sume += num_edges(*itpop);
suml += (*itpop)[graph_bundle].maxlevel;
}

sumn /= (double)pop.size();
sume /= (double)pop.size();
suml /= (double)pop.size();

avnodes << sumn << ", ";
avedges << sume << ", ";
avlevels << suml << ", ";

double divn = 0;
double dive = 0;
double divl = 0;


double sumdivn = 0;
double sumdive = 0;
double sumdivl = 0;

for (itpop=pop.begin(); itpop != pop.end(); itpop++) {
divn = (num_vertices(*itpop) - sumn);
divn *= divn;
dive = (num_edges(*itpop) - sume);
dive *= dive;
divl = ((*itpop)[graph_bundle].maxlevel - suml);
divl *= divl;

sumdivn += divn;
sumdive += dive;
sumdivl += divl;
}

sumdivn /= (double)pop.size();
sumdive /= (double)pop.size();
sumdivl /= (double)pop.size();

sumdivn = sqrt(sumdivn);
sumdive = sqrt(sumdive);
sumdivl = sqrt(sumdivl);

sdnodes << sumdivn << ", ";
sdedges << sumdive << ", ";
sdlevels << sumdivl << ", ";
} else {
for (int f = 0; f < (tsteps - y); f++) {
sdnodes << "NaN, ";
sdedges << "NaN, ";
sdlevels << "NaN, ";
avnodes << "NaN, ";
avedges << "NaN, ";
avlevels << "NaN, ";
}

for (int ff = 1; ff < pop.size(); ff++) {
survivorlength[ff] = tsteps;
}
}
}
if (pop.size() == 0) {
cout << "EXTINCT" << endl;
ext[y] += 1;
y = tsteps;
} else {
if (x == 0) {
abilities << pop.back()[graph_bundle].abtotal << ", ";
}
}


cout << x << "." << y << endl;

}

if (analytics == 0) {
if (pop.size() > 0) {
write_graphviz(bodyplans, pop.front(),
my_node_writer(pop.front()));
}
}

/*for (int g = 0; g < clustercont.size(); g++) {
cluster << clustercont[g] << ',';
}*/

/*if (analytics == 1) {
int degreedis;

for (unsigned int j = 0; j < num_vertices(pop.back()); j++) {
degreedis = 0;
degreedis = out_degree(j, pop.back());
degreedistrib << degreedis << ',';
}
}*/

if (pop.size() > 0) {
double meandivdev = accumulate(divdev2.begin(), divdev2.end(),(double) 0);
meandivdev /= (double)divdev2.size();
double diffdd;
for (int u = 0; u < divdev2.size(); u++){
diffdd = divdev2[u] - meandivdev;
divdev2[u] = diffdd*diffdd;
}
diffdd = accumulate(divdev2.begin(), divdev2.end(), (double) 0);
diffdd /= (double)divdev2.size();
standevdiv << sqrt(diffdd) << ", ";

double meancopdev = accumulate(copdev2.begin(), copdev2.end(), (double) 0);
meancopdev /= (double)copdev2.size();
for (int uu = 0; uu < copdev2.size(); uu++){
diffdd = copdev2[uu] - meancopdev;
copdev2[uu] = diffdd*diffdd;
}
diffdd = accumulate(copdev2.begin(), copdev2.end(), (double) 0);
diffdd /= (double)copdev2.size();
```

```
standevcopy << sqrt(diffdd) << ", ";
} else {
standevdiv << "NaN" << ", ";
standevcopy << "NaN" << ", ";
}

if (analytics == 1) {
Population::iterator itpop;
for (unsigned int n = 0; n < num_vertices(pop.back()); n++) {
levs[(pop.back())[n].level] += 1;
}
}


pop.release();
copdev2.release();
}

} else {
for (int x = 0; x < runs; x++) {
double prob1 = 0;

Genopop pop;

systemsize = 1000;
lambda = 0.0001;
int balls = 1;

int specext = 0;

//pop.push_back(new string("[{0}{0}{0}][{111}]"));
//string first = "[{0}{0}{0}][{111}]";
string first = "[{0}]";
//pop.push_back(new string ("[{0}]"));
pop.push_back(new string);
pop.back() = first;

ptr_vector<double> copynums;
copynums.push_back(new double (1) );
copynums.back() = systemsize;

doublelists agevec;

doublelists ablist, mainlist;

ablist.push_back(new ptr_vector<double>);
if (ages == 1) {
agevec.push_back(new ptr_vector<double>);
for (int tic = 0; tic < copynums.back(); tic++) {
(agevec.back()).push_back(new double);
(agevec.back()).back() = 0;
}
}


if (resources != 1) {
for (string_find_iterator It = make_find_iterator(pop.back(),
first_finder("{0}", is_iequal()));
It != string_find_iterator(); ++It) {
(ablist.back()).push_back(new double (1) );
(ablist.back()).back() = normal_gen(eng, normdist);
if ((ablist.back()).back() < 0) {
(ablist.back()).back() = 0 - (ablist.back()).back();
}

}
/*
(ablist.back()).push_back(new double (1) );
(ablist.back()).back() = std::accumulate(
(ablist.back()).begin(), (ablist.back()).begin() + 3,
(double) 0);
*/
```

```
if (maintenance == 1) {

mainlist.push_back(new ptr_vector<double>);

for (string_find_iterator It = make_find_iterator(
pop.back(), first_finder("{0}", is_iequal()));
It != string_find_iterator(); ++It) {
(mainlist.back()).push_back(new double (1) );

(mainlist.back()).back() = normal_gen(eng, normdist);
if ((mainlist.back()).back() < 0) {
(mainlist.back()).back() = 0
- (mainlist.back()).back();
}
}
/*
(mainlist.back()).push_back(new double (1) );
(mainlist.back()).back() = std::accumulate(
(mainlist.back()).begin(),
(mainlist.back()).begin() + 3, (double) 0);
*/
}

if (norm == 1) {
popnorm(pop, ablist, mainlist);
}

}

for (int y = 0; y < tsteps; y++) {

prob1 = rn();

if (prob1 < (lambda * systemsize)
|| (dynamics == 0 && resources == 0) || y == 0) {
mutation(pop, ablist, mainlist, copynums, eng,
normdist, balls, agevec);
//cout << "mutation" << endl;
if (norm == 1 && resources == 0) {
popnorm(pop, ablist, mainlist);
}
//cout << "popnorm" << endl;
if (analytics == 1) {
pop.pop_front();
copynums.erase(copynums.begin());
(ablist.front()).erase((ablist.front()).begin(),(ablist.front()).end());
ablist.pop_front();
if (maintenance == 1) {
mainlist.release(mainlist.begin());
}
}

}

/* Check for ball conservation */
int totalballs = 0;
int numvert = 0;
int t = 0;
for (Genopop::iterator its = pop.begin(); its != pop.end(); its++) {
numvert = 0;
for (string_find_iterator It = make_find_iterator((*its),
first_finder("{", is_iequal())); It != string_find_iterator();
++It) {
numvert += 1;
}
totalballs += copynums[t] * numvert;
t++;
}

totalballs += balls;

if (analytics != 1) {
```

```
if (totalballs != 1001) {
cout << "BALLS NOT CONSERVED" << endl;
cout << "total balls is " << totalballs << endl;
}
}
degcalc(pop, degdiv, degdis, y, counter, copynums);
cout << "deg calc" << endl;
if (analytics == 0) {
if (resources == 0) {
if (dynamics == 1) {
popgrowth(pop, ablist, mainlist, copynums,
systemsize, specext, agevec);
} else {
selection(pop, ablist, mainlist);
}

if (dynamics == 1) {
if (x == 0) {
ptr_vector<double>::iterator itcop;
for (itcop = copynums.begin();
itcop != copynums.end(); itcop++) {
copynos << (*itcop) << " ";
//cout << (*itcop) << endl;
}
copytracker << pop.size() << " ";
}
}
} else {
balls = resourcealloc(pop, copynums, balls, agevec);
for (unsigned int r = 0; r < pop.size(); r++) {
copynos << copynums[r] << ", ";
}

copytracker << pop.size() << " ";
}
}
sizediversity[y] += pop.size();

Population::iterator itpop;

if (pop.size() == 0) {
cout << "EXTINCT" << endl;
ext[y] += 1;
y = tsteps;
} else {
if (x == 0 && resources == 0) {
cout << "ablist size is " << ablist.size() << endl;
abilities
<< accumulate((ablist.back()).begin(),
(ablist.back()).end(), (double) 0)
<< ", ";
}
}

cout << x << "." << y << endl;
}

double avgen = 0;

for (Genopop::iterator itpop = pop.begin(); itpop != pop.end();
itpop++) {
avgen += (*itpop).length();
}

/*if (analytics == 1) {
int degreedis;
char_separator<char> sep("[{}]");
tokeniser tokens(pop.back(), sep);

for (tokeniser::iterator node = tokens.begin();
node != tokens.end(); node++) {
degreedis = 0;
```

```
string tokstr = *node;
for (string_find_iterator It = make_find_iterator(tokstr,
first_finder("1", is_iequal()));
It != string_find_iterator(); ++It) {
degreedis += 1;
}
degreedistrib << degreedis << ',';
}
}*/

for (Genopop::iterator itpop = pop.begin(); itpop != pop.end();
itpop++) {
genotypes << *itpop << endl;
}

avgen /= (double)pop.size();

genolength << avgen << ", " << endl;

cout << specext << endl;

pop.erase(pop.begin(), pop.end());
ablist.erase(ablist.begin(), ablist.end());
mainlist.erase(mainlist.begin(), mainlist.end());
copynums.erase(copynums.begin(), copynums.end());

if (dynamics == 1 || resources == 1) {
copynos << endl;
copytracker << endl;
}
}

}

//dd += ((double) num_edges(branches)
/// (double) num_vertices(branches));

//write_graphviz(branching, branches);

/* Level data collection */
//Population::iterator itsl;
//levelmax = levelcount(pop);
/*
if (leveldiversity.size() < levelmax) {
int diff = levelmax-leveldiversity.size();
for (int i = 0; i < diff; i++) {
leveldiversity.push_back(new double (1) );
leveldiversity.back() = 0;
sizecounter.push_back(new double (1) );
sizecounter.back() = 0;
}

}

for (itsl = pop.begin(); itsl != pop.end(); itsl++) {
leveldiversity[(*itsl)[graph_bundle].maxlevel - 1] += 1;
sizecounter[(*itsl)[graph_bundle].maxlevel - 1] += 1;
}*/

for (unsigned int lev = 0; lev < levs.size(); lev++) {
levs[lev] = levs[lev]/(double)runs;
nodesperlev << levs[lev] << ", ";
}

for (unsigned int le = 0; le < ext.size(); le++) {
extinctions << ext[le] << ", ";
}


for (int d = 0; d < survivorlength.size(); d++) {
survivals << survivorlength[d] << ", ";
}
```

```
double opdiv;
for (unsigned int lic = 0; lic < sizediversity.size(); lic++) {
opdiv = sizediversity[lic] / (double)runs;
diversity << opdiv << ", ";
}

double opdiv2;
for (unsigned int r = 0; r < degdiv.size(); r++) {
if (degdiv[r] != 0) {
opdiv2 = degdiv[r] / counter[r];
} else {
opdiv2 = 0;
}

degrees << opdiv2 << ", ";
}

for (unsigned int ctick = 0; ctick < cnumticker.size(); ctick++) {
for (unsigned int ref = 0; ref < tsteps; ref++) {
copynos << cnumticker[ctick][ref] << " ";
}
copynos << "; ";
}

degdiv.release();

sizediversity.release();

ext.release();
//diversity << "]; " << endl;
//fitot.release();
//leveldiversity.release();
//cout << zz << "." << z << endl; // Inner loop counter
//}
ancestry << "];" << endl;
nodesperlev << "];" << endl;
survivals << "];" << endl;
standevcopy << "];" << endl;
standevdiv << "];" << endl;
extinctions << "];" << endl;
copynos << "];" << endl;
diversity << "];" << "\n";
abilities << "];" << "\n";
degrees << "];" << "\n";
morphospace << "];";
copytracker << "];" << endl;
//degreedistrib << "];" << endl;
genolength << "];" << endl;
cluster << "];" << endl;
muttracker << "];" << endl;
sdnodes << "];";
sdedges << "];";
sdlevels << "];";
avnodes << "];";
avedges << "];";
avlevels << "];";
//}
muttracker.close();
ancestry.close();
nodesperlev.close();
avnodes.close();
avedges.close();
avlevels.close();
sdnodes.close();
sdedges.close();
sdlevels.close();
survivals.close();
standevcopy.close();
standevdiv.close();
cluster.close();
```

```
copytracker.close();
morphospace.close();
copynos.close();
abilities.close();
diversity.close();
degrees.close();
//degreedist << "];";
//degreedist.close();
//branching.close();
bodyplans.close();
extinctions.close();
//degreedistrib.close();
genolength.close();
genotypes.close();
//novertices.close();
//nolevels.close();

return 0;
}
```

Bibliography

Peter A Abrams and Hiroyuki Matsuda. Fitness minimization and dynamic instability as a consequence of predator–prey coevolution. *Evolutionary Ecology*, 11 (1):1–20, 1997.

Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

Réka Albert, Hawoong Jeong, and Albert-László Barabási. Internet: Diameter of the world-wide web. *Nature*, 401(6749):130–131, 1999.

R. Amundson. *The changing role of the embryo in evolutionary thought: roots of evo-devo*. Cambridge Univ Pr, 2005a. ISBN 0521806992.

R. Amundson. *The changing role of the embryo in evolutionary thought: roots of evo-devo*. Cambridge Univ Press, 2005b.

M Madan Babu, Nicholas M Luscombe, L Aravind, Mark Gerstein, and Sarah A Teichmann. Structure and evolution of transcriptional regulatory networks. *Current opinion in structural biology*, 14(3):283–291, 2004.

Albert-László Barabási and Zoltan N Oltvai. Network biology: understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.

H. Barringer, D. Rydeheard, B. Warboys, and D. Gabbay. A revision-based logical framework for evolvable software. In *Proceedings of the 25th conference on IASTED International Multi-Conference: Software Engineering*, pages 78–83. ACTA Press, 2007.

L.J. Billera, S.P. Holmes, and K. Vogtmann. Geometry of the space of phylogenetic trees* 1. *Advances in Applied Mathematics*, 27(4):733–767, 2001.

M. Björklund. The importance of evolutionary constraints in ecological time scales. *Evolutionary Ecology*, 10(4):423–431, 1996.

AJ Boucot. So-called background extinction rate is a sampling artifact. *Palaeoworld*, 15(2):127–134, 2006.

John FY Brookfield. Evolution and evolvability: celebrating darwin 200. *Biology letters*, 5(1):44–46, 2009.

N.J. Butterfield. Macroevolution and macroecology through deep time. *Palaeontology*, 50(1):41–55, 2007.

N.J. Butterfield. Animals and the invention of the phanerozoic earth system. *Trends in ecology & evolution*, 26(2):81–87, 2011.

Gregory Chaitin. *Proving Darwin: Making Biology Mathematical*. Random House Digital, Inc., 2012.

Peter Chesson and Nancy Huntly. The roles of harsh and fluctuating conditions in the dynamics of ecological communities. *The American Naturalist*, 150(5): 519–553, 1997.

K. Christensen, S.A. Di Collobiano, M. Hall, and H.J. Jensen. Tangled nature: a model of evolutionary ecology. *Journal of theoretical biology*, 216(1):73–84, 2002.

Philippe Claeys, Jean-G Casier, and Stanley V Margolis. Microtektites and mass extinctions- evidence for a late devonian asteroid impact. *Science*, 257(5073): 1102–1104, 1992.

J.H. Costello, S.P. Colin, and J.O. Dabiri. Medusan morphospace: phylogenetic constraints, biomechanical solutions, and ecological consequences. *Invertebrate Biology*, 127(3):265–290, 2008.

V Courtillot, JJ Jaeger, Zhenyu Yang, G Feraud, and C Hofmann. The influence of continental flood basalts on mass extinctions: where do we stand? *Geological Society of America Special Papers*, 307:513–525, 1996.

C. Darwin. On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life. *New York: D. Appleton*, 1859.

L. Demetrius. The origin of allometric scaling laws in biology. *Journal of theoretical biology*, 243(4):455–467, 2006.

Jared M Diamond. Biogeographic kinetics: estimation of relaxation times for avifaunas of southwest pacific islands. *Proceedings of the National Academy of Sciences*, 69(11):3199–3203, 1972.

J.A. Dunne, R.J. Williams, and N.D. Martinez. Network structure and biodiversity loss in food webs: robustness increases with connectance. *Ecology Letters*, 5(4): 558–567, 2002a.

Jennifer A Dunne, Richard J Williams, and Neo D Martinez. Food-web structure and network theory: the role of connectance and size. *Proceedings of the National Academy of Sciences*, 99(20):12917–12922, 2002b.

M. Durinx and J.A.J. Metz. *Multi-type branching processes and adaptive dynamics of structured populations*, volume 5. Cambridge Univ Pr, 2005.

N. Eldredge and S.J. Gould. Punctuated equilibria: an alternative to phyletic gradualism. *Models in paleobiology*, 82:115, 1972.

D H Erwin. Macroevolution is more than repeated rounds of microevolution. *Evolution & development*, 2(2):78–84, 2000. ISSN 1520-541X. URL http://www.ncbi.nlm.nih.gov/pubmed/11258393.

D.H. Erwin and E.H. Davidson. The evolution of hierarchical gene regulatory networks. *Nature Reviews Genetics*, 10(2):141–148, 2009. ISSN 1471-0056.

Douglas H Erwin. Early introduction of major morphological innovations. *Acta Palaeontologica Polonica*, 38(3):4, 1994.

Douglas H. Erwin. Novelties that change carrying capacity. *Journal of Experimental Zoology Part B: Molecular and Developmental Evolution*, 318 (6):460–465, 2012. ISSN 1552-5015. doi: 10.1002/jez.b.21429. URL http://dx.doi.org/10.1002/jez.b.21429.

F. Galis, J.J.M. van Alphen, and J.A.J. Metz. Why five fingers? evolutionary constraints on digit numbers. *Trends in Ecology & Evolution*, 16(11):637–646, 2001.

T. Gánti. Theoretical deduction of the function and structure of the genetic material. *Biológia*, 22:17–35, 1974.

T. Gánti and I. Koch. *A theory of biochemical supersystems and its application to problems of natural and artificial biogenesis*. University Park Press, 1979. ISBN 0839114117.

John Gerhart and Marc Kirschner. The theory of facilitated variation. *Proceedings of the National Academy of Sciences*, 104(suppl 1):8582–8589, 2007.

S.A.H. Geritz, É. Kisdi, G. Meszéna, and J.A.J. Metz. Adaptive dynamics of speciation: ecological underpinnings. *Adaptive Speciation*, pages 54–75, 2004.

S.J. Gould. A developmental constraint in cerion, with comments of the definition and interpretation of constraint in evolution. *Evolution*, pages 516–539, 1989.

S.J. Gould and R.C. Lewontin. The spandrels of San Marco and the Panglossian paradigm: a critique of the adaptationist programme. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 205(1161):581–598, 1979. ISSN 0080-4649.

Stephen Jay Gould. The paradox of the first tier: an agenda for paleobiology. *Paleobiology*, pages 2–12, 1985.

Stephen Jay Gould and Elisabeth S Vrba. Exaptation-a missing term in the science of form. *Paleobiology*, pages 4–15, 1982.

Anthony Hallam and Paul B Wignall. *Mass extinctions and their aftermath*. Oxford University Press, USA, 1997.

Thomas F Hansen. Is modularity necessary for evolvability?: Remarks on the relationship between pleiotropy and evolvability. *Biosystems*, 69(2):83–94, 2003.

Thomas F Hansen, Christophe Pélabon, and David Houle. Heritability is not evolvability. *Evolutionary Biology*, 38(3):258–277, 2011.

Alan R Hildebrand, Glen T Penfield, David A Kring, Mark Pilkington, Antonio Camargo, Stein B Jacobsen, and William V Boynton. Chicxulub crater: a possible cretaceous/tertiary boundary impact crater on the yucatan peninsula, mexico. *Geology*, 19(9):867–871, 1991.

JP Hodych and GR Dunning. Did the manicouagan impact trigger end-of-triassic mass extinction? *Geology*, 20(1):51–54, 1992.

John H. Holland. *Hidden order: how adaptation builds complexity*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1995. ISBN 0-201-40793-0.

I.G. Johnston, S.E. Ahnert, J.P.K. Doye, and A.A. Louis. Evolutionary dynamics in a simple model of self-assembly. *Physical Review E*, 83(6):066105, 2011.

T.H. Jukes and M. Kimura. Evolutionary constraints and the neutral theory. *Journal of molecular evolution*, 21(1):90–92, 1984.

S. Kauffman and S. Levin. Towards a general theory of adaptive walks on rugged landscapes. *Journal of theoretical Biology*, 128(1):11–45, 1987. ISSN 0022-5193.

S.A. Kauffman and R.G. Smith. Adaptive automata based on Darwinian selection. *Physica D: Nonlinear Phenomena*, 22(1-3):68–82, 1986. ISSN 0167-2789.

G Keller. Impacts, volcanism and mass extinction: random coincidence or cause and effect? *Australian Journal of Earth Sciences*, 52(4-5):725–757, 2005.

Irene Keller, Douda Bensasson, and Richard A Nichols. Transition-transversion bias is not universal: a counter example from grasshopper pseudogenes. *PLoS genetics*, 3(2):e22, 2007.

M. Kimura. Evolutionary rate at the molecular level. *Nature*, 217(5129):624–626, 1968.

James W Kirchner and Anne Weil. Delayed biological recovery from extinctions throughout the fossil record. *Nature*, 404(6774):177–180, 2000.

Mikko Kuussaari, Riccardo Bommarco, Risto K Heikkinen, Aveliina Helm, Jochen Krauss, Regina Lindborg, Erik Öckinger, Meelis Pärtel, Joan Pino, Ferran Rodà, et al. Extinction debt: a challenge for biodiversity conservation. *Trends in Ecology & Evolution*, 24(10):564–571, 2009.

Lawrence R Lawlor and J Maynard Smith. The coevolution and stability of competing species. *American Naturalist*, pages 79–99, 1976.

O. Leimar. Evolutionary change and darwinian demons. *Selection*, 2(1):65–72, 2002.

Seth Lloyd and Heinz Pagels. Complexity as thermodynamic depth. *Annals of Physics*, 188(1):186–213, 1988.

Oliver Mason and Mark Verwoerd. Graph theory and networks in biology. *Systems Biology, IET*, 1(2):89–119, 2007.

Robert M May and Robert H Mac Arthur. Niche overlap as a function of environmental variability. *Proceedings of the National Academy of Sciences*, 69(5): 1109–1113, 1972.

J. Maynard Smith and E. Szathmáry. *The major transitions in evolution.* Oxford University Press, USA, 1997. ISBN 019850294X.

George R McGhee. *The geometry of evolution: adaptive landscapes and theoretical morphospaces.* Cambridge University Press New York, 2007.

Daniel W McShea. Functional complexity in organisms: Parts as proxies. *Biology and Philosophy*, 15(5):641–668, 2000.

Daniel W McShea. The evolution of complexity without natural selection, a possible large-scale trend of the fourth kind. *Paleobiology*, 31(2):146–156, 2005.

D.W. McShea. Perspective: Metazoan complexity and evolution: Is there a trend? *Evolution*, pages 477–492, 1996.

Géza Meszéna, István Czibula, and Stefan Geritz. Adaptive dynamics in a 2-patch environment: a toy model for allopatric and parapatric speciation. *Journal of Biological Systems*, 5(02):265–284, 1997.

J.A.J. Metz. Thoughts on the geometry of meso-evolution: collecting mathematical elements for a post-modern synthesis. *The Mathematics of Darwins Legacy. Birkhauser, Basel.*, 2011.

J.A.J. Metz, RM Nisbet, and SAH Geritz. How should we define "fitness" for general ecological scenarios? *Trends in Ecology & Evolution*, 7(6):198–202, 1992.

J.A.J. Metz, S.A.H. Geritz, G. Meszéna, F.J.A. Jacobs, and JS Van Heerwaarden. Adaptive dynamics, a geometrical study of the consequences of nearly faithful reproduction. *Stochastic and spatial structures of dynamical systems*, pages 183–231, 1996.

J. M. Montoya and R.V Solé. Small world patterns in food webs. *Journal of Theoretical Biology*, 214(3):405 – 412, 2002. ISSN 0022-5193. doi: 10.1006/jtbi.2001.2460. URL `http://www.sciencedirect.com/science/article/pii/S0022519301924609`.

José M Montoya, Stuart L Pimm, and Ricard V Solé. Ecological networks and their fragility. *Nature*, 442(7100):259–264, 2006.

C. Mora, D.P. Tittensor, S. Adl, A.G.B. Simpson, and B. Worm. How many species are there on earth and in the ocean? *PLoS biology*, 9(8):e1001127, 2011.

HJ Muller. The gene material as the initiator and the organizing basis of life. *The American Naturalist*, 100(915):493–517, 1966. ISSN 0003-0147.

Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

K. Niklas. Evolutionary walks through a land plant morphospace. *Journal of Experimental Botany*, 50(330):39, 1999.

RM Nisbet, EB Muller, K. Lika, and S. Kooijman. From molecules to ecosystems through dynamic energy budget models. *Journal of Animal Ecology*, 69(6):913–926, 2000.

MD Pagel and Paul H Harvey. Comparative methods for examining adaptation depend on evolutionary models. *Folia Primatologica*, 53(1-4):203–220, 2008.

Michael E Palmer and Marcus W Feldman. Survivability is more fundamental than evolvability. *PloS one*, 7(6):e38025, 2012.

R. Pastor-Satorras, E. Smith, R.V. Solé, et al. Evolving protein interaction networks through gene duplication. *Journal of Theoretical biology*, 222(2):199–210, 2003.

M.R. Pie and J.S. Weitz. A null model of morphospace occupation. *The American Naturalist*, 166(1):E1–E13, 2005.

KR Popper. The logic of scientific discovery. 1968.

Rudolf A Raff. The shape of life: genes, development, and the evolution of animal form. 1996.

DA Rand and HB Wilson. Using spatio-temporal chaos and intermediate-scale determinism to quantify spatially extended ecosystems. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 259(1355):111–117, 1995.

David M Raup. A kill curve for phanerozoic marine species. *Paleobiology*, pages 37–48, 1991.

D.M. Raup. Extintion: bad genes or bad luck? *Acta geológica hispánica*, 16(1): 25–33, 1981.

Erzsébet Ravasz, Anna Lisa Somera, Dale A Mongru, Zoltán N Oltvai, and A-L Barabási. Hierarchical organization of modularity in metabolic networks. *science*, 297(5586):1551–1555, 2002.

W.E. Reif, T. Junker, and U. Hoßfeld. The synthetic theory of evolution: general problems and the German contribution to the synthesis. *Theory in Biosciences*, 119(1):41–91, 2000. ISSN 1431-7613.

Joseph Reisinger, Kenneth O Stanley, and Risto Miikkulainen. Towards an empirical measure of evolvability. In *Proceedings of the 2005 workshops on Genetic and evolutionary computation*, pages 257–264. ACM, 2005.

J. Rhodes, C.L. Nehaniv, and M.W. Hirsch. *Application of automata theory and algebra.* World Scientific Publishing Co., 2010.

Robert A Rohde and Richard A Muller. Cycles in fossil diversity. *Nature*, 434(7030): 208–210, 2005.

Charles Albert Sandberg, Willi Ziegler, Roland Dreesen, and Jamie L Butler. Late frasnian mass extinction: conodont event stratigraphy, global changes, and possible causes. *LPI Contributions*, 673:160, 1988.

Erwin Schrödinger. What is life? the physical aspect of the living cell, 1944.

Peter Schulte, Laia Alegret, Ignacio Arenillas, José A Arz, Penny J Barton, Paul R Bown, Timothy J Bralower, Gail L Christeson, Philippe Claeys, Charles S Cockell, et al. The chicxulub asteroid impact and mass extinction at the cretaceous-paleogene boundary. *Science*, 327(5970):1214–1218, 2010.

P. Schuster. Mathematical modeling of evolution. solved and open problems. *Theory in Biosciences*, pages 1–19, 2010.

Virgil L Sharpton, G Brent Dalrymple, Luis E Marín, Graham Ryder, Benjamin C Schuraytz, and Jaime Urrutia-Fucugauchi. New links between the chicxulub impact structure and the cretaceous/tertiary boundary. *Nature*, 359(6398):819–821, 1992.

Eugene M Shoemaker and Ruth F Wolfe. Mass extinctions, crater ages and comet showers. In *The galaxy and the solar system*, volume 1, pages 338–386, 1986.

Ricard V Sole and José M Montoya. Complexity and fragility in ecological networks. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 268 (1480):2039–2045, 2001.

Ricard V Solé, Pau Fernández, and Stuart A Kauffman. Adaptive walks in a gene network model of morphogenesis: insights into the cambrian explosion. *International journal of developmental biology*, 47(7):685–693, 2003.

R.V. Solé and S. Valverde. Spontaneous emergence of modularity in cellular networks. *Journal of the Royal Society Interface*, 5(18):129–133, 2008.

Bärbel MR Stadler, Peter F Stadler, Günter P Wagner, and Walter Fontana. The topology of the possible: Formal spaces underlying patterns of evolutionary change. *Journal of Theoretical Biology*, 213(2):241–274, 2001.

SC Stearns. The role of development in the evolution of life histories. *Evolution and development*, pages 237–258, 1982.

Darcy Wentworth Thompson et al. On growth and form. *On growth and form.*, 1942.

Jan D Van Der Laan and Pauline Hogeweg. Predator-prey coevolution: interactions across different timescales. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 259(1354):35–42, 1995.

J. van der Meer. Metabolic theories in ecology. *Trends in ecology & evolution*, 21 (3):136–140, 2006.

A. Vázquez, A. Flammini, A. Maritan, and A. Vespignani. Modeling of protein interaction networks. *Complexus*, 1(1):38–44, 2003.

L Vinicius. Modular evolution: how natural selection produces biological complexity. 2010.

CH Waddington. Paradigm for an evolutionary process. *Towards a theoretical biology*, 2:106–128, 1969.

Andreas Wagner. Robustness and evolvability: a paradox resolved. *Proceedings of the Royal Society B: Biological Sciences*, 275(1630):91–100, 2008.

Günter P Wagner and Peter F Stadler. Quasi-independence, homology and the unity of type: A topological theory of characters. *Journal of Theoretical Biology*, 220(4):505–527, 2003.

G.B. West, J.H. Brown, and B.J. Enquist. A general model for the origin of allometric scaling laws in biology. *Science*, 276(5309):122–126, 1997.

Rosalind V White and Andrew D Saunders. Volcanism, impact and mass extinctions: incredible or credible coincidences? *Lithos*, 79(3):299–316, 2005.

AD Wissner-Gross and CE Freer. Causal entropic forces. *Physical Review Letters*, 110(16):168702, 2013.