On Selection for Evolvability

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Science and Engineering.

2016

Andrew M. Webb

School of Computer Science

Contents

This thesis contains 37792 words.

List of Tables

List of Algorithms

List of Figures

Abstract

On Selection for Evolvability

Andrew M. Webb

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy, 2016

This thesis is about direct selection for evolvability in artificial evolutionary systems. The origin of evolvability—the capacity for adaptive evolution—is of great interest to evolutionary biologists, who have proposed many indirect selection mechanisms. In evolutionary computation and artificial life, these indirect selection mechanisms have been co-opted in order to engineer the evolution of evolvability into artificial evolution simulations. Very little work has been done on direct selection, and so this thesis investigates the extent to which we should select for evolvability. I show in a simple theoretical model the existence of conditions in which selection for a weighted sum of fitness and evolvability achieves greater long-term fitness than selection for fitness alone. There are no conditions, within the model, in which it is beneficial to select more for evolvability than for fitness. Subsequent empirical work compares *episodic group selection for evolvability* (EGS)—an algorithm that selects for evolvability estimates calculated from noisy samples—with an algorithm that selects for fitness alone on four fitness functions taken from the literature. The long-term fitness achieved by EGS does not exceed that of selection for fitness alone in any region of the parameter space. However, there are regions of the parameter space in which EGS achieves greater long-term evolvability. A modification of the algorithm, EGS-AR, which incorporates a recent best-arm identification algorithm, reliably outperforms EGS across the parameter space, in terms of both eventual fitness and eventual evolvability. The thesis concludes that selection for estimated evolvability may be a viable strategy for solving time-varying problems.

Declaration

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

Copyright

Publications

Andrew Webb, Julia Handl, and Joshua Knowles. "How Much Should You Select for Evolvability?" In: Proceedings of the European Conference on Artificial Life, 2015, Vol. 13, pp. 487-494.

Andrew Webb and Joshua Knowles. "Studying the Evolvability of Self-Encoding Genotype-Phenotype Maps". In: Proceedings of the International Conference on the Synthesis and Simulation of Living Systems, 2014, Vol. 14, pp. 79-86.

Acknowledgements

First and foremost, I thank my supervisors, Dr Joshua Knowles and Dr Julia Handl, for the balance they struck between letting me roam free and reining me in. Thank you for the occasional—and necessary—criticism, and for your time, expertise, and support. Thanks also to Dr Joseph Mellor, whose advice has been frequent and invariably helpful, and to the School of Computer Science and EPSRC for their financial support.

Secondly, thank you to the friends I've made, without whom my time at the university would have been much less enjoyable. Particular thanks to Andrew Mundy, Jonathan Heathcote, James Knight, and Bernard D'Antras for the hikes and the lunches; to Andrew Leeming and Michael Lee for the dinners and the cynicism; to Joe Razavi, Henry Reeve, and Salman Aljammaz; and to Francis Southern, Jon Parkinson, Nikos Nikolaou, Steve "Fluff" Miller, Andy Chambers, and the rest of the CDT 2012 cohort.

Finally, thank you to my family, without whose endless encouragement and support this would not have been possible. Thanks to mum and dad for Wednesdays; to Richard, Emma, and Isaac for Fridays; and to George for weekends.

Chapter 1

Introduction

The topic of this thesis is direct selection for evolvability in artificial evolutionary systems, such as evolutionary algorithms.

There are three necessary conditions for evolution by natural selection: heritability, variation, and differential reproductive success (Lewontin, 1970); if a population of organisms differ from each other in heritable traits that affect their fitness—their ability to survive and reproduce—then traits that confer higher fitness will be selected for and spread in the population. However, such selection reduces the amount of variation within a population, and so for evolution to continue there must be mechanisms to introduce new variation. In natural organisms, these mechanisms are mutation and genetic recombination, with mutations being the ultimate source of new genetic material. In artificial evolution, such as evolutionary computation and artificial life simulations, it is up to the designer of the system to define the variation-introducing mechanisms.

It is now acknowledged that adaptive evolution requires the introduction of not only variation, but of potentially adaptive variation; at least some variant offspring must be fitter than their parents. The ability to produce

such adaptive variants is called *evolvability*.

The importance of evolvability has long been understood in the field of evolutionary computation, in which the principles of evolution are used to find solutions to problems. It is up to the designer of an evolutionary algorithm to choose how to represent candidate solutions and how mutations will operate on those representations. For a given problem, it cannot be taken for granted that a given representation will lead to evolvability. As noted by Ciliberti et al. (2007), man-made systems tend to be brittle, with a small disruption of one part leading to a disastrous failure of the whole, and brittle systems are not amenable to adaptive evolution by mutation and selection. As a specific example, in genetic programming (the evolution of computer programs), one possibility is to evolve source code directly, with programs being mutated by replacing each character in the source code with low probability with another character. Nearly all such mutations are catastrophic for fitness.

As noted by Wagner and Altenberg (1996), biologists took longer to ask questions about evolvability and its origins, because they deal with a system that found suitable representations in the distant evolutionary past, and so evolvability within that system could be taken for granted.

It can be the case that individuals within a population differ from each other in their evolvability. In this case, selection for evolvability is possible, particularly if fitness and evolvability are correlated. Mechanisms for indirect selection for evolvability are sought by evolutionary biologists. Such indirect selection mechanisms have been co-opted by researchers in evolutionary computation and artificial life in order to build the evolution of evolvability into artificial systems.

This thesis is about direct selection for evolvability in artificial evolution—

for example, within an evolutionary algorithm—and will mostly be of interest to evolutionary algorithm practitioners. There has been surprisingly little work on this topic. The motivation of this work is two-fold.

- By directly selecting for evolvability, we might achieve higher long-term fitness on some fitness functions.
- By studying direct selection for evolvability, we might better understand the dynamics of selection for evolvability in natural evolution.

1.1 Research Questions

This thesis aims to answer two questions concerning direct selection for evolvability, which are as follows.

1. Under the conditions that evolvability information is accurate and obtained without a cost in terms of fitness evaluations, to what extent should we select for evolvability? This question is addressed in Chapter 3.
2. Under the conditions that evolvability must be estimated, and that there is a cost in terms of fitness evaluations to obtain evolvability estimates, do the evolutionary algorithms described in Chapters 4 and 5, which incorporate selection for evolvability estimates, achieve greater long-term fitness or evolvability than an algorithm that selects for fitness alone? This question is addressed in Chapters 4 and 5.

1.2 Contributions

The research contributions of this thesis are as follows.

Simple Evolvability Model I develop a simple model, designed in order to study selection for evolvability in the case that certain assumptions are satisfied. I ask to what degree we should select for evolvability in order to maximize eventual fitness. The findings are as follows.

1. I obtain an exact expression for the degree to which we should select for evolvability as a function of the model parameters.

2. An examination of the expression reveals that we should never select more strongly for evolvability than we select for fitness, regardless of the model parameters.

3. Taking the limit of the expression as the number of generations goes to infinity reveals that, in order to maximize long-term eventual fitness, we should select equally for fitness and evolvability unless evolvability decays towards some non-zero value.

4. In the case that evolvability does decay towards some non-zero value I find an expression, in terms of a small subset of the model parameters, for the degree to which we should select for evolvability in the limit that the number of generations goes to infinity.

Episodic Group Selection I introduce the *episodic group selection* (EGS) algorithm. A comparison of this algorithm optimizing four time-varying fitness functions to an algorithm which selects for fitness alone yields the following findings.

1. For most pairings of fitness function and evolvability estimation method, we can identify regions of the parameter space in which EGS outperforms selection for fitness alone in terms of the eventual evolvability

achieved, though we cannot identify regions of the parameter space in which EGS outperforms selection for fitness alone in terms of the eventual fitness achieved.

2. The use of a sequential Bayesian filter to estimate population evolvabilities leads to improved performance compared to relying on an evolvability estimate calculated from a single offspring population.

3. Termination heuristics—designed to determine when to stop selecting for evolvability in order to make more efficient use of fitness evaluations—are found to almost universally increase the achieved long-term fitness of the EGS algorithm.

Episodic Group Selection with Asynchronous Reproduction I introduce the *episodic group selection with asynchronous reproduction* (EGS-AR) algorithm. A comparison of EGS-AR with an algorithm which selects for fitness alone and with EGS yields the following findings.

1. For most pairings of fitness function and evolvability estimation method, we can identify regions of the parameter space in which EGS-AR outperforms selection for fitness alone in terms of the eventual evolvability achieved, though we cannot identify regions of the parameter space in which EGS-AR outperforms selection for fitness alone in terms of the eventual fitness achieved.

2. Averaging over the parameter distributions that we sample from, EGS-AR outperforms EGS in terms of both the eventual evolvability and the eventual fitness achieved.

1.3 Thesis Structure

The remainder of this thesis is organized as follows. Chapter 2 gives the necessary background to understand the concept of evolvability, its importance in natural and artificial evolutionary systems, and the wider context of this work. In this chapter, I discuss the various definitions and measures of evolvability that have been used, describe how individuals can differ from each other in their evolvabilities, and discuss prior research that aims either to discover the origins of evolvability in natural evolutionary systems or to engineer the evolution of evolvability into artificial systems. By the end of this chapter, the reader should understand the motivation for the work and be equipped to understand the work itself.

In Chapter 3 I introduce the simple evolvability model. I first list the assumptions of the model. I then derive an expression for the optimal constant selection weight in terms of the model parameters, and comment on some features of this expression. I end with a discussion of the assumptions, and of the consequences if some of them are relaxed.

Chapter 4 introduces the episodic group selection (EGS) algorithm. First, I explore the effect of relaxing the assumption of the simple evolvability model that evolvability measurements are both accurate and precise, and are made without a cost in fitness evaluations. I then describe the EGS algorithm, the evolvability measures that it uses, and the methods the algorithm uses to estimate evolvabilities, including two sequential Bayesian filtering algorithms. I then describe the four time-varying fitness functions on which the algorithm is tested and the experimental design. I compare the performance of the EGS algorithm with an algorithm which selects for fitness alone, and comment upon the results.

Chapter 5 introduces the episodic group selection with asynchronous re-

production (EGS-AR) algorithm. First, I motivate the algorithm by framing the problem of determining the most evolvable population from a sequence of evolvability measurements as a multi-armed bandit problem. I then describe a recent best-arm identification algorithm, *pure exploration Thompson sampling* (PTS), and how the EGS algorithm is modified to use PTS to choose which population will go through a generation of mutation and selection next. I then describe my experimental design and compare the results of the EGS-AR algorithm to those of the EGS algorithm and selection for fitness alone on the same four time-varying fitness functions as the previous chapter.

Chapter 6 describes the existing research most related to my own. I describe five pieces of research related to direct selection for evolvability in artificial evolution. I discuss the novel features and shortcomings of each piece of work. I directly compare the findings of these works to my own, and conclude that some progress has been made.

Chapter 7 concludes the thesis. A summary and analysis of the findings are followed by a list of the limitations of the work. I end with some suggestions for future work, some to address the listed limitations, and some to take the research in new directions.

Chapter 2

Evolvability

The topic of this thesis is artificial selection for evolvability. This chapter gives an overview of previous thought and research on evolvability and selection for evolvability, by biologists and researchers in evolutionary computation and artificial life.

We must first be clear what we mean by 'evolvability', since there are many definitions in use in the literature. Section 2.1 lists some commonly used definitions and identifies the definition that I use in the remainder of this thesis.

In order for selection for evolvability to be possible, it must be possible for individuals to vary in their evolvabilities. In Section 2.2 I describe the factors that can cause individuals in natural and artificial evolutionary systems to differ from one another in their evolvabilities. I explore the commonalities and differences between these factors, and briefly discuss the implications of these differences for the evolution of evolvability.

In order to select for evolvability we must be able to quantify the evolvability of an individual. In Section 2.3 I list some commonly used measures and identify a family of measures that I use in the remainder of this thesis.

Section 2.4 lists the mechanisms that researchers have proposed for selection for evolvability in natural and artificial evolutionary systems. I postpone discussion of research concerning direct, artificial selection for evolvability until Chapter 6, in which I will compare this research with my own.

The reader should bear in mind throughout this chapter that, although much of what is discussed is biological, the ultimate interest of this thesis is selection for evolvability in artificial systems, such as within evolutionary algorithms. The ideas contained in this chapter inform the direct evolvability selection strategies described in Chapters 3 to 5.

2.1 What is Evolvability?

As noted by Sniegowski and Murphy (2006), definitions of evolvability are "almost as numerous as the papers and books that have been written on the subject". Just as the overloading of the term 'complexity'—both in and outside of an evolutionary context—has led to conceptual confusion, Pigliucci (2008) believes that by 'evolvability' we really mean a family of related concepts; we shouldn't ask which definition is correct. We should instead identify the range of concepts covered by the existing definitions, and then ask ourselves which concept is intended whenever we invoke 'evolvability'.

I do not introduce a new definition here, nor do I systematically categorize the definitions that have come before. In this section, I use the taxonomies of evolvability definitions that have been produced by other researchers in order to identify the intended meaning of 'evolvability' in this thesis.

Pigliucci (2008) places definitions of evolvability into three categories. The first, for which he suggests the term 'heritability', is evolvability in the sense of Houle (1992). This is the ability of populations to respond to selec-

tion, and so it is determined by the existing genetic variation in a population. Evolvability in this sense is a property of populations; it is meaningless to ascribe it to individuals. Moreover, selection cannot increase evolvability in this sense, as selection always decreases variation. Pigliucci's second category of evolvability definitions, which he (confusingly) names 'evolvability', is exemplified by Wagner and Altenberg's definition; evolvability is "the genome's ability to produce adaptive variants when acted upon by the genetic system... there must be some small chance of a variant being adaptive[1]" (Wagner and Altenberg, 1996). In contrast to the first category, evolvability in this sense is about the capacity to generate—rather than the prior existence of—suitable genetic variation. Pigliucci's third category, 'innovation', exemplified by Maynard Smith and Szathmáry's *major transitions* (Maynard Smith and Szathmáry, 1997), concerns major changes in the way that variation is generated that allow the exploration of new regions of phenotype space. The transition from unicellular to multicellular life is an example of such a change. Such major transitions open up new regions of phenotype space for exploration by shifting variance from a lower level (e.g., variation between cells) to a higher level (e.g., variation between configurations of cells).

Pigliucci notes that most proposed evolvability definitions fall into his second category. Similarly, Sniegowski and Murphy (2006) observe that most recent work on evolvability is to do with the ability to produce heritable, selectable variation. Gallagher (2009), in his own categorization of evolvability definitions, identifies a distinct school who use the term in the sense originally used by Dawkins (2003). Since for Dawkins, 'evolvability' is to do with the ability to evolve by generating suitable phenotypic variation, this school also falls into Pigliucci's second category.

[1]By 'adaptive', the authors mean 'better', rather than 'able to adapt or to respond.'

For the remainder of this thesis, 'evolvability' is meant in the sense of Pigliucci's second category. Table 2.1 lists some definitions from the literature that belong to this category. If confusion arises concerning what is meant by 'evolvability', the reader should refer back to Table 2.1. Sections 2.2 and 2.3 make these definitions more concrete, by describing the ways in which individuals in natural and artificial systems can differ from one another in their evolvabilities, and by describing how one measures evolvability as it is defined here.

Table 2.1: Definitions of 'evolvability' that fall into Pigliucci's second category.

Hansen (2006)	—	"The ability to produce and maintain potentially adaptive genetic variants."
Kirschner and Gerhart (1998)	—	"The capacity to generate heritable phenotypic variation."
Wagner and Altenberg (1996)	—	"The genome's ability to produce adaptive variants when acted upon by the genetic system."'
Marrow et al. (1999)	—	"The capacity to evolve," "The capacity for evolutionary change."
Dawkins (2003)	—	"A tendency to evolve in certain directions, or even just a tendency to evolve at all."
Conrad (1990)	—	"[Suitability] for self-organisation through... variation and selection."

There has been some disagreement about whether we ought to think of evolvability as a property of individuals, populations, or somewhere in between. It is sometimes argued that since evolvability is, roughly, the capacity to evolve, and individuals do not evolve, it cannot be a property of individuals (Dawkins, 2003; Sniegowski and Murphy, 2006). It seems to follow that it must be a property of populations, and this then leads to a discussion

about whether selection for evolvability must invoke selection on an evolutionary unit higher than the individual, such as clade, kin, or group selection. Alberch (1991) defines evolvability as a property of lineages, since lineage selection can be a stronger force than group selection. Conrad (1979a, 1990) seems to be the first to define evolvability as a property of individuals, and others have followed (Altenberg, 1994, 1995; Kirschner and Gerhart, 1998). We can ascribe evolvability as defined here to individuals if we think if it not as the capacity to evolve, but as the capacity for one's descendant lineage to evolve.

2.2 What Determines the Evolvability of an Individual?

This section examines how two otherwise identical individuals in a natural or artificial evolutionary system might differ in their evolvabilities, i.e., how the kinds of variation that are likely to occur in their descendants might differ such that one lineage is better able to adapt than the other. My approach to modelling and selecting for evolvability, which I describe in Chapters 3 to 5, is agnostic to the reason for differences between individuals' evolvabilities, requiring only that differences exist and are heritable. However, the ideas described in this section informed the choice of environments in which to test my approach.

In this section, I use the following terminology. Whether describing a natural or artificial system, I use 'genome' to mean the directly heritable information of an individual, passed down during reproduction in a possibly mutated form. By 'phenome', I mean the form and function of the individual; the sum of the traits of an individual that are 'visible' to selection, i.e., that

affect fitness. For a natural organism, for example, weight, height, colouring, and the ability to catalyse some reaction are all examples of traits that make up the phenome.

For a natural organism, or a simulated organism within an artificial life simulation, 'fitness' might be defined as the number of offspring of that organism that reach reproductive age. In the context of evolutionary computation, 'fitness' is usually taken to mean a value indicating the quality of an individual or candidate solution, which is then used to determine reproductive success.

An individual exists in the context of an 'environment', by which I mean the sum of all factors affecting an individual's fitness that are not properties of the individual itself, i.e., that are not part of the phenome. For a natural or simulated organism, the 'environment' is the literal biotic and abiotic environment in which the organism finds itself, including any other organisms that it interacts with. For a candidate solution in an evolutionary algorithm, the fitness function is the environment. I will use 'genome-phenome map' to mean a function that determines the phenome of an individual given its genome. I use this terminology in favour of the more common 'genotype', 'phenotype', and 'genotype-phenotype map' in order to avoid the confusion that can arise due to the overloading of terminology; Mahner and Kary (1997) have identified seven conflicting definitions of 'genotype', five of 'genome', and five of 'phenotype'. The phrase 'genotype-phenotype map' can cause particular confusion, as we will see below, and I will avoid using this phrase.

The fitness of an individual, then, is jointly determined by the phenome of the individual and the environment. We can visualize the space of all phenomes, and the mapping from phenomes to fitness determined by the environment, as in Figure 2.1. The nodes on the left of the figure repre-

sent phenomes. The edges between phenomes represent probable mutations. Horizontal lines connect phenomes to their fitness values.

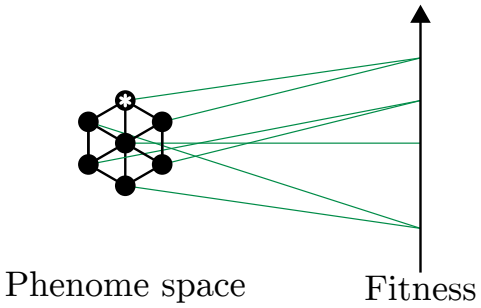

Figure 2.1: The environment determines the mapping from phenome space to fitness. The asterisk indicates the phenome of a particular individual. The nodes on the left-hand side are phenomes, which are connected to each other by likely mutations. The horizontal lines show how the environment maps each phenome onto a fitness value.

When a given individual, with a given phenome, reproduces there will be some probability distribution over the phenome of the offspring. For example, the individual with the phenome marked with an asterisk in Figure 2.1 might have a uniform probability of generating offspring with the phenomes it is connected to in that figure. This offspring phenome distribution of an individual, along with the environment, gives a probability distribution over the fitness of the offspring of the individual. As we will see in Section 2.3, evolvability is often measured in terms of this offspring fitness distribution.

It is mutations that are the ultimate source of this distribution. However, typically, mutation doesn't operate on the phenome directly. In an evolutionary algorithm (EA), mutation typically operates on some kind of data structure, such as strings or trees over some alphabet. Mutation usually operates in a way that is considered 'natural' for that data structure. For example, the mutation operator on a string might replace each character with some low probability with another character drawn uniformly from the

alphabet. During selection, the data structures are interpreted as representing candidate solutions to the problem. It is well known that the success of an evolutionary algorithm in finding a solution to a problem depends on how candidate solutions are encoded. This is known as the *representation problem* (Wagner and Altenberg, 1996). In natural organisms, mutations come in the form of insertions, deletions, inversions and other errors that occur with low probability across the genome. It is the combination of developmental processes and environmental influences that create the organism from its genome.

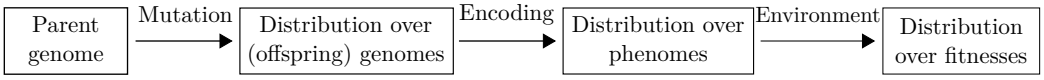

Figure 2.2: The pathway from the parent's genome to the probability distribution over the offspring's genome, phenome, and fitness.

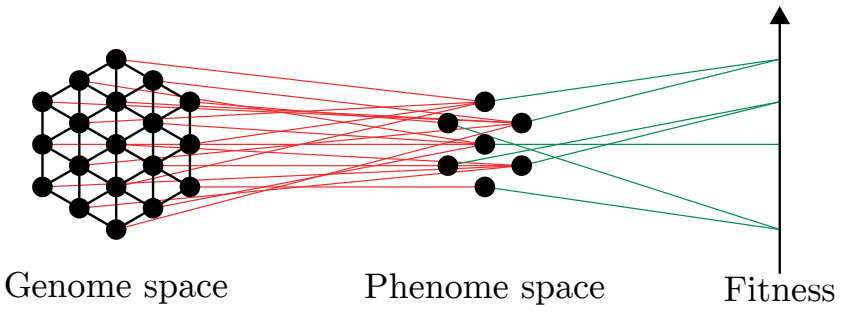

Figure 2.3: Genome space, phenome space, fitness, and the mappings between them determined by developmental systems/solution encodings and the environment. The connections between genomes show likely mutations.

Figure 2.2 shows the pathway from the parent genome to the distribution over offspring fitnesses described above. Figure 2.3 shows an example genome space and its mapping onto phenome space and ultimately onto fitness. The edges between genomes represent mutations that have a non-negligible proba-

bility of occurring. The edges giving the mapping from genomes to phenomes are, in the case of EAs, given by the way in which the algorithm designer chooses to encode candidate solutions, and in the case of natural organisms, are determined by developmental processes. Figures 2.2 and 2.3 are an illustration of what Altenberg (1994) calls the 'transmission function'.

Although the environment (i.e., the mapping from phenomes to fitness) might vary over time, and might (as is common) depend on the current state of the population, in this section we are asking how two individuals with the same phenome can differ in their evolvabilities. If the individuals coexist in the same environment, then they can differ in evolvabilities only if they have different offspring phenome distributions. It was in terms of differing offspring phenome distributions that Kirschner and Gerhart (1998) described evolvability.

Although the probabilities of different mutations (i.e., the edges between genomes in Figure 2.3) can be under evolutionary control—for example, there are mutator genes that increase the probability of mutations in surrounding genes—we will consider this mutational neighbourhood to be fixed. Since we consider the distribution over mutations and the environment to be invariant between individuals, we are left with two ways that individuals can differ in their evolvabilities independently of their phenome; if the mapping from genome to phenome is many-to-one, then changing the genome can leave the phenome unchanged while changing the offspring phenome distribution. Alternatively, the mapping from genome to phenome itself can change under some circumstances. I will now explore these two possibilities.

An evolutionary algorithm practitioner might compare the evolvabilities achieved by individuals when two or more different representations are used. Figure 2.4 illustrates two competing ways of representing phenomes in the

same genome space. The genome and phenome marked with an asterisk indicate the genome and phenome of a particular individual. The right hand side of the figure shows the phenome neighbourhood of that individual, i.e., the phenomes reachable by likely mutations. The figure shows how two representations can give the same genome-phenome pair different phenome neighbourhoods, and therefore differing evolvabilities.

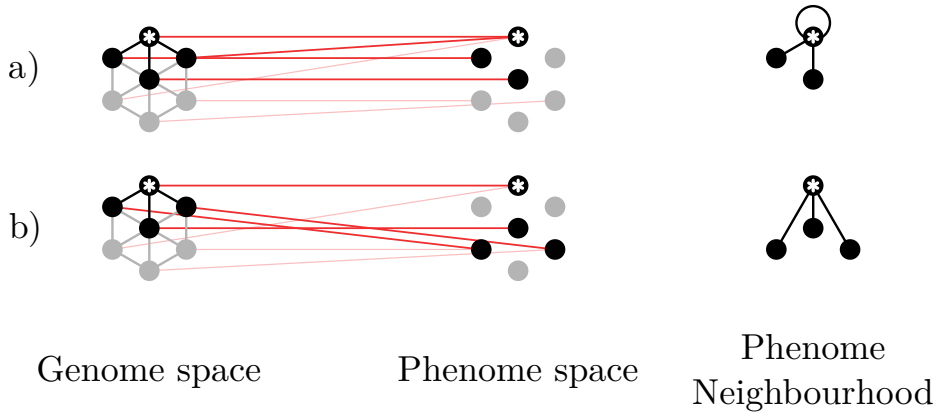

Figure 2.4: An example of a change of genome-phenome map. The asterisk indicates the genome and phenome of a particular individual. The right-hand side of the figure shows the 'phenome neighbourhood' of that individual, i.e., the phenomes that are reachable by likely mutations. a) and b) represent two different genome-phenome maps.

An example of this is Dawkins' exploration of competing developmental systems for the two-dimensional artificial organisms in his *biomorphs* simulation experiments (Dawkins, 2003). This is an example of an outside observer manipulating phenome neighbourhoods, and therefore evolvability, by changing the rules of the system. How might evolvability change in a heritable way, within the rules of the system? Dawkins added genes to his artificial organisms that act as switches between rival developmental systems. The idea was that this would allow individuals to differ in their developmental systems, and therefore their evolvabilities. Similarly, in an evolutionary algo-

rithm, we might use one or more genes to specify how the candidate solutions are encoded in the rest of the genome. This is illustrated in Figure 2.5. This figure represents the same system as Figure 2.4, with a single added gene that switches an individual between the rival mappings. The single vertical line connecting the two halves of the genome space are a visual shorthand for a mutation connecting each corresponding genome. Note, however, that this new representation switching gene is a part of the genome, and considered as a whole, the genome-phenome map is fixed.

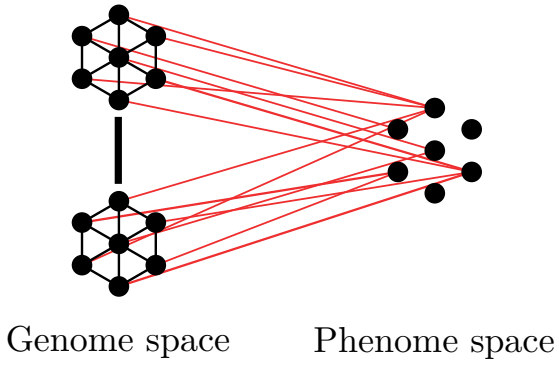

Figure 2.5: An illustration of the action of an extra gene which controls the representation of an evolutionary algorithm or the developmental system of an organism. The thick vertical line is a shorthand for a mutation connecting each node in the upper region of the genome space with the corresponding node in the lower region. Mutations to this extra gene change how the remainder of the genome maps onto the phenome.

This kind of representation switching gene is a particularly straightforward example of a so-called 'neutral' mutation that alters the phenome neighbourhood without changing the phenome. A generalization is shown in Figure 2.6. This figure shows a genome-phenome map that allows such neutral mutations.

Toussaint (2002) identifies some properties that a genome-phenome map would have to have in order that phenotypic variability becomes, in the course of evolution, structured in such a way that phenome space is explored more

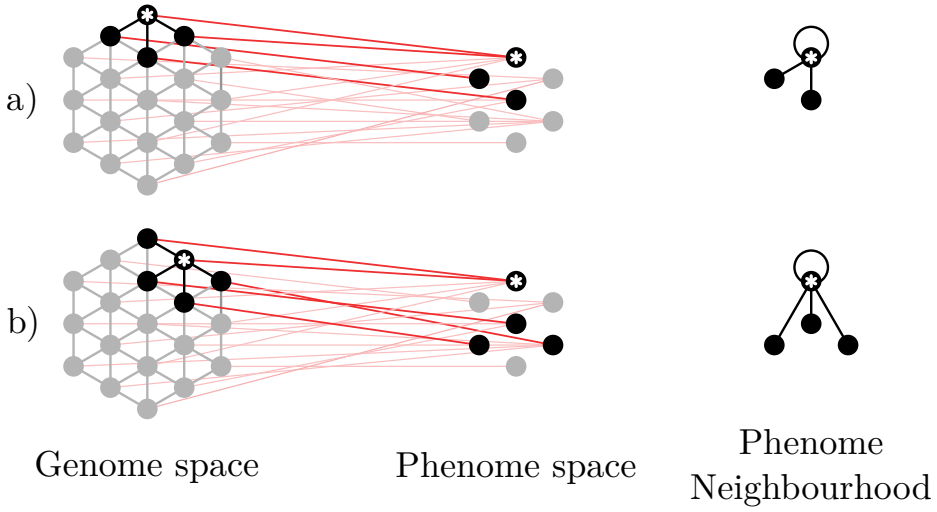

Figure 2.6: A genome-phenome map that allows 'neutral' mutations, which leave the phenome unchanged but alter the phenome neighbourhood. The asterisk indicates the genome and phenome of a particular individual a) before and b) after a neutral mutation.

effectively. One property identified is that it must be the case that in order to understand the trajectory of phenotypic evolution, one must consider neutral traits. Toussaint calls mappings with these properties 'non-trivial'. He points to tertiary structure, i.e., the shapes of proteins after folding, as an example of a mapping from genome (DNA) to phenome (protein shape) that has the desired properties. The genome-phenome map having these properties is one way that individuals can differ in their evolvabilities.

In natural organisms, there are mechanisms of non-genetic inheritance. These are means of transmitting information—of varying reliability—between generations outside of the genome. Examples include DNA methylation, chromatin modifications, and the transmission from parent to developing offspring of hormones, cytokines, and microorganisms (Toth, 2014). We can imagine that some of these factors might influence the developmental process, and therefore seemingly the genome-phenome map. Moreover, these factors

can be 'mutated' by environmental noise. However, this is not qualitatively different from the kind of changes within a 'non-trivial' genome-phenome map described above. It is a matter of terminology; we may include these non-genetic heritable factors in what we call the 'genome'.

I will briefly discuss here a terminological problem caused by the varying uses of the words 'genotype' and 'phenotype' noted above. A neutral mutation such as that shown in Figure 2.6 would be commonly called a change in the 'genotype-phenotype map'. In this usage, 'phenotype' means some subset of the traits of an individual that are currently under consideration, and 'genotype' means the set of genes considered to directly code for those traits. For an example of this usage, consider Wagner and Altenberg (1996), who propose that the genotype-phenotype map may evolve by epistatic mutations or by the creation of new genes. Sometimes, what I have called here the genome-phenome map is called the 'genotype-phenotype map as a whole' (Toussaint, 2002), and is considered to be fixed. However, some researchers, typically computer scientists, use 'genotype-phenotype map' synonymously with our usage of 'genome-phenome map'[2].

It has been claimed that heritable changes to the genome-phenome map, or Toussaint's 'genotype-phenotype map as a whole', are logically inconsistent (Toussaint, 2002). The argument is that heritable information is contained in the genome, and so the genome cannot determine the genome-phenome map in any meaningful way, for if it did then we could infer the map, and therefore the phenome, from the genome alone, and the genome-phenome map would in fact be fixed—a contradiction. This argument can be extended to include the non-genetic inheritance mechanisms discussed above.

However, the argument fails if we consider the dynamic and partially

[2]Lee Altenberg, personal correspondence.

self-referential nature of the developmental systems of organisms. McMullin (2012) has advocated studying artificial evolutionary systems in which the developmental systems of organisms are specified self-referentially, specifically in order to study their capacity for the evolution of evolvability. He motivates his work with a discussion of John von Neumann's *Universal Constructor* (Von Neumann and Burks, 1966), which was a sort of proof by construction that machines are capable of producing machines of greater complexity than themselves, and that evolution could lead to machines of ever greater complexity. It will aid us to briefly consider the operation of von Neumann's Constructor. The Constructor is a configuration within a 2D cellular automaton. The Constructor consists of the following.

- A one dimensional structure called the 'tape', which is inactive; by itself it does nothing.
- An active 'machine', which amongst other components contains
  - A 'tape copier', which causes a copy of the tape to be produced, offset from the original by some distance
  - A 'decoder', which consults the tape, interprets it as specifying a two-dimensional structure, and constructs that structure adjacent to the copied tape.

The Constructor is self-reproducing in the case that the tape, when interpreted by the decoder, specifies the Constructor itself, including its decoder. The 'meaning' of the tape, i.e., the configuration that it specifies, depends upon the decoder that is interpreting it. This is also true of the part of the tape that specifies the decoder itself. This is the self-referential property referred to above. Von Neumann noted that a mutation in the part of the tape

that specifies the decoder would result in an offspring Constructor that would interpret its tape differently from its parent. He discounted such mutations as inevitably leading to infertility. However, as McMullin (2000) notes, not all such offspring would be infertile. A fertile mutant with a modified decoder corresponds to a change in the developmental system. Moreover, such a change might not be reversible by simply reversing the mutation.

Consider the following toy example of a non-reversible change of this kind. Suppose we have an organism that has two possible developmental systems, D_1 and D_2, and that the specification of the developmental system is contained in a gene with two alleles, g_1 and g_2. Suppose that the *meaning* of this gene in the offspring of an individual is conditional on the developmental system implemented by the parent in the following way.

$$\begin{aligned} D_1(g_1) &= D_1, \quad D_2(g_1) = D_2, \\ D_1(g_2) &= D_2, \quad D_2(g_2) = D_2. \end{aligned} \tag{2.1}$$

This means that if the parent implements developmental system D_1, then an offspring with gene g_i will implement developmental system D_i. However, if the parent implements developmental system D_2, the offspring will also implement D_2 regardless of the allele of the gene that is present.

An illustrative—but rather implausible—biological example of a change to the developmental system of this kind would be a change to the genetic code via modifications to *transfer RNA* (tRNA). The final stage of the translation from DNA to protein is accomplished by matching up triplets of nucleotides to amino acids. Which triplets match with which amino acids is determined by the shape of the tRNA in the cell; each tRNA molecule contains the complement of a nucleotide triplet on one end (and so has an affinity for the triplet itself), and is shaped on the other end to have an affinity for

a particular amino acid. In a sense, the tRNA determines the 'meaning' of the DNA currently being translated. However, the tRNA is itself specified in DNA, and is constructed in the same manner as any other protein structure; this aspect of development is self-referential in the sense described above. We can imagine an unlikely mutation that causes altered tRNA to be constructed such that the genetic code is changed, and further that this change is what McMullin calls 'backward compatible', i.e., that the new tRNA, when reading the mutated tRNA genes, cause themselves to be constructed.

This kind of change to the genome-phenome map seems to be qualitatively different to the neutral mutations illustrated by Figure 2.6. One of the major differences is that this kind of mutation can produce individuals that are not self-reproducers. Above, we discussed systems where in the absence of mutations offspring resemble parents. However, consider the system which is as follows.

$$\begin{aligned} D_1(g_1) &= D_1 \\ D_1(g_2) &= D_2 \\ D_2(g_2) &= D_3 \\ D_3(g_2) &= D_4 \\ &\vdots \end{aligned} \tag{2.2}$$

In this system, an organism with developmental system D_1 is a self-reproducer in the sense that, in the absence of mutations, the offspring and parent are identical. However, a mutation to allele g_2 moves us away from this fixed point, and the 'offspring' (if the word is appropriate under these circumstances) do not resemble the parents at all. Figure 2.7 is an illustration of this kind of evolutionary system. Each of the three networks in the figure represent a genome. A mutation takes an individual from one node

to the corresponding node in another network. The shape of the nodes represent phenomes, and the arrows represent parent-offspring relationships in the absence of mutations. Note that only one individual, in the left-hand network is depicted as self-reproducing. The middle network depicts a four generation cycle, which switches between two phenomes.

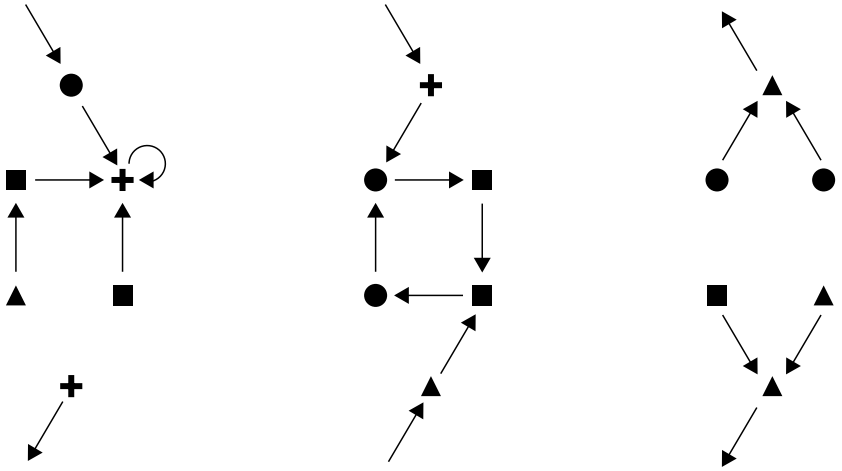

Figure 2.7: Each network represents a genome. Mutations take an individual from one node to the corresponding node in a different network. Shapes represent phenomes. Arrows represent parent-offspring relationships in the absence of mutations.

Dawkins (2003) finds it useful to make a distinction between two kinds of mutation: ordinary changes within an existing genetic system, and changes to the genetic system itself. Mutations that affect development as described above seem to be a good candidate for the latter.

I have described in this section two ways in which individuals can differ in their offspring phenome distributions, and therefore in their evolvabilities. The first was neutral mutations in what Toussaint calls a 'non-trivial' genome-phenome map. The second was a heritable change to the genome-phenome map itself.

2.3 Measures of Evolvability

This section reviews proposed measures of evolvability from the literature, focusing on measures that roughly capture evolvability as defined in Section 2.1.

In Chapters 3 to 5, I use measures of evolvability that are functions of the *offspring fitness distribution* (OFD). For this reason I categorize the measures described here into those that are and are not functions of the OFD.

2.3.1 Measures not related to the offspring fitness distribution

Since evolvability, as defined in Section 2.1, is related to the capacity for adaptive evolution, one obvious way to measure it (with respect to some environment) is to measure the adaptation of a population to an environment or environments over time. Van Belle and Ackley (2002) evolve programs to approximate a periodically changing function. They measure the mean per-generation fitness increase within each function epoch. This measures how effective the population is at finding a good approximation to the function presented within each epoch. In another paper in which the target changes gradually, the same authors measure the online fitness (i.e., the average fitness so far) of a population (Van Belle and Ackley, 2003). If evolvability can change over time, then a suitable extension would be to measure the average fitness over a sliding window. A longer window would give more information about how effective the evolutionary search is within the window, but the measurement would be less precise in time.

Draghi and Wagner (2009) use a similar method to measure the evolvability of an individual (rather than a population) with respect to a set of

environments. They report the mean amount of adaptation (averaging over trials and environments) towards the (known) target optimal phenotype in a given number of generations after seeding the initial population with copies of the individual. They measure adaptation by how much closer the evolved phenotypes are to the known target phenotype than the ancestral phenotypes were. Quayle and Bullock (2006) use a similar measure to the above, but report the time taken to hit a (known) phenotypic target rather than the fitness achieved by a given time.

Reisinger et al. (2005) have a population evolve in a periodically switching environment, and they distinguish between a training and test phase. They measure the correlation between the amount of time spent in the training phase and the performance in the testing phase. By doing so, they measure the degree to which the population retains and generalizes information learned about the domain. Note that according to the definition of evolvability we are using, Reisinger et al. measure the *evolution of evolvability* rather than evolvability itself.

Some measures are not directly related to fitness or adaptation, while still capturing evolvability as defined in Section 2.1. Palmer and Feldman (2012) calculate the predicted probability of a lineage avoiding extinction for k generations. They argue that while the probability of beneficial mutations occurring—and the degree of fitness increase when they do occur—is important, the incidence of severely deleterious mutations is important too. They argue that their *k-survivability* measure gives more of an indication of long-term evolutionary success than more standard evolvability measures such as those described in this chapter.

Some measures require being able to identify traits or features of organisms. Hansen et al. (2011) measure a trait's response to selection per strength

of selection. The *evolutionary activity measure* of Bedau and Packard (1996) measures the rate at which adaptive features are introduced. Adaptive features are defined as those that survive for longer than some threshold amount of time. Features can be traits, genes, or any other feature of an individual that can influence fitness—and thus can differ in their ability to survive—and can be identified.

2.3.2 Measures related to the offspring fitness distribution

The measures of evolvability described in this section are functions of the *offspring fitness distribution* (OFD). Section 2.2 gives some idea of the determining factors of this distribution. The measures described here depend on the pre-selection introduction of variation, and are primarily properties of individuals.

During natural or artificial reproduction, mutations occur with some probability across the whole genome, and the genetic material of two parents might be combined in a random fashion to produce the genome of the offspring. The end result of this large number of random events is the offspring, which has some fitness value; we can treat the fitness of the (unknown, perhaps yet unborn) offspring of a parent or parent(s) as a real-valued random variable. Altenberg (1994) is a proponent of quantifying evolvability in terms of this fitness distribution of the offspring, and we will return to his proposed measure shortly. This distribution is a property of the parent, and the fitnesses of any offspring the parent actually has are samples from this distribution.

Figure 2.8 illustrates what the (relative) offspring fitness distribution might look like due to a single random mutation. The distribution over

fitness given that a single mutation has occurred is sometimes called the *distribution of fitness effects of a mutation* in the literature. A relative fitness of 0 in the figure represents a mutation that is either lethal or renders the offspring infertile. A relative fitness of 1 means that the mutation has no effect on fitness. Mutations giving relative fitness less than 1 are detrimental, and those giving relative fitness greater than 1 are beneficial. The shown distribution is realistic for some natural organisms, with most mutations being either lethal or nearly neutral (Eyre-Walker and Keightley, 2007).

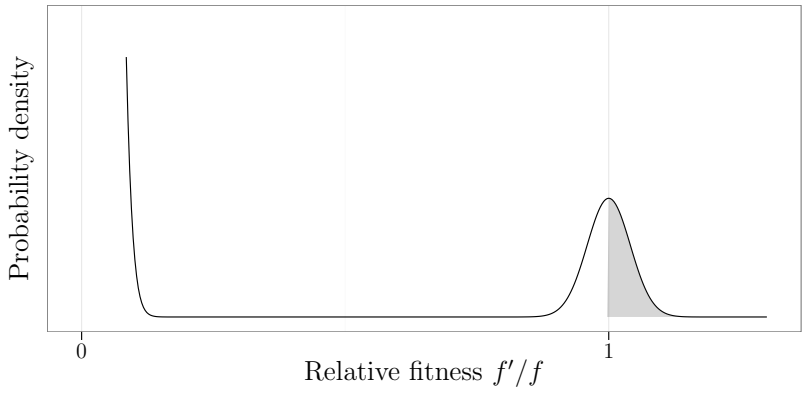

Figure 2.8: An example probability distribution of the fitness effect of a mutation. The horizontal axis shows the relative fitness of the offspring, i.e., the fitness of the offspring divided by the fitness of the parent. The example distribution has most probability mass around zero (representing lethal mutations) and one (representing neutral or nearly neutral mutations).

In practice, the true OFD of an individual will be hidden from us, and the samples we have (the fitnesses of the offspring the individual actually has) will be the only information we gain about the distribution. The measures of evolvability described below, being functions of the OFD, must be *estimated* using the fitnesses of the actual offspring. These estimates of evolvability are themselves random variables, since their values depend on the observed offspring fitnesses. Some of the novelty of my own work, described in Chap-

ters 4 and 5 is in the combination of estimates of evolvability from subsequent generations to produce a less noisy estimate of evolvability.

Smith et al. use as a measure of evolvability the probability of a beneficial mutation. This corresponds to the shaded area in Figure 2.8. They also measure the average fitness of the top Cth percentile of the offspring fitness distribution, which takes into account the shape of the upper tail (Smith et al., 2002). Altenberg also takes into account the shape of the distribution, measuring the correlation between the parent's fitness and the upper tail of the offspring fitness distribution (Altenberg, 1994). Other proposed measures include the variance or standard deviation of fitness, or the maximum offspring fitness (amongst N offspring) (Gallagher, 2009, p. 42), and the expected fitness after selection (Turney, 1999).

2.4 Evolvability in Natural and Artificial Systems

This section reviews proposed explanations for the evolvability observed in natural systems (in Section 2.4.1) and proposed mechanisms to build evolvability—or the evolution of evolvability—into artificial systems (in Section 2.4.2).

For natural systems, this is a question of how it came to be that the kinds of variation induced in natural organisms as a result of mutation are in some way well suited to the environment, such that adaptation can proceed by evolution. For artificial systems, it is a question of how we can build systems such that that is true. The approach taken by researchers answering this question depends on their motivations and goals. Artificial life researchers, while dealing with artificial simulations and models, are ultimately trying to

understand life as a general phenomenon, or *life as it could be*. Artificial life researchers are therefore likely to restrict themselves to studying biologically plausible mechanisms to encourage evolvability. Researchers in Evolutionary Computation (EC), on the other hand, are interested in building algorithms to solve problems. Although evolutionary algorithms are inspired by evolution, and researchers may use the explanations proposed by biologists for the evolvability observed in the natural world in order to build better algorithms (and vice versa), EC researchers need not constrain themselves to biologically plausible mechanisms.

2.4.1 Explanations for evolvability in natural systems

As noted by Wagner and Altenberg (1996), biologists did not initially look for explanations for evolvability; the existence of life on Earth, and the variety of life, was evidence that natural organisms were undergoing adaptive mutations at a sufficient rate. It was easy to overlook the question of how life attained this evolvability. Wagner and Altenberg point to Levinton (1988) as an early acknowledgement that biologists had until that point studied "the fate of variability but not the production of variability"[3].

It has long been clear in Evolutionary Computation (EC) that the success of an evolutionary algorithm in solving a problem depends on how candidate solutions are represented or encoded; it is easy to design evolutionary systems which are not evolvable. Indeed, early acknowledgement in the field of biology for the need for an explanation for evolvability in nature came from a biologist experimenting with artificial evolution simulations. Dawkins (2003)

[3]We have here another problem of terminology. Wagner and Altenberg, and later authors, use 'variation' to mean actual, present differences between individuals in a population, and 'variability' as the production of such differences. For these, Levinton uses 'variability' and 'the production of variability' respectively.

(published in its original form in 1989) studies the evolution of *biomorphs*. These are two dimensional patterns intended to mimic the kind of body patterns seen in various plants and animals. Evolution of biomorphs proceeds interactively, with the researcher choosing for reproduction those biomorphs which most resemble an existing body plan. What Dawkins found was that the success of this procedure depends crucially on the kind of development that the biomorphs undergo, restricting or heavily biasing the phenotypes available. If the biomorphs undergo a developmental stage that ensures certain symmetries, it is much easier to evolve familiar body plans than if the biomorph genome is unstructured, with each gene specifying (for example) the end-points of a line.

Once it became clear that evolvability required an explanation, some researchers argued that evolutionary theory as it stood did not explain the variational properties of natural organisms (Sniegowski and Murphy, 2006). There was disagreement about whether there was any need to invoke the *evolution of evolvability*, and if so, if there could be *selection for evolvability*.

Amongst those who believe that evolvability has evolved, there is ongoing debate about the mechanisms involved (Pigliucci, 2008), although Hansen (2006) argues that it is naive to expect that any single mechanism alone explains the evolution of evolvability. Proposed mechanisms can be roughly separated into those that involve more or less direct selection for evolvability as a result of the changes in fitness that it leads to over subsequent generations, and those in which the increase of evolvability is a side-effect of selection for other traits. This separation corresponds to the categorization by Visser et al. (2003) of explanations for the evolution of phenotypic robustness into *adaptive* and *congruent* hypotheses, later co-opted by Hansen (2006) in order to categorize explanations for the evolution of evolvability.

So-called *congruent* hypotheses for the evolution of evolvability posit that evolvability increases as a result of selection for robustness to environmental noise. The idea is that variation due to mutations often affects organisms in similar ways to variation due to environmental noise. As a result, organisms that can survive environmental variation, by reducing or otherwise shaping its effects, will be better able to survive genetic variation (Hansen, 2006). Since environmental variation is more prevalent than genetic variation, and the ability to survive environmental variation has a direct effect on fitness, this creates strong selection for the ability to survive genetic variation. For the remainder of this section I focus on adaptive explanations for the evolution of evolvability, i.e., those that require selection for evolvability.

Selection for evolvability as a result of the fitness differences that result has been controversial for two reasons. The first is the problem of teleology. Researchers have noted that selection for evolvability seems to require selection for traits not because they increase fitness, but for the future effects those traits have on fitness within a lineage (Sniegowski and Murphy, 2006; Pigliucci, 2008). Dawkins (2003) is careful to be clear that he isn't making an argument that requires such foresight in evolution. But as Gallagher (2009, p.16) notes, this argument of a problem of teleology is put forward most often by researchers in order to rebut it. Selection for evolvability requires the same condition as selection for any other trait; that there exist variations in evolvability within a population. All that is required additionally is that differences in evolvability are sustained for long enough that correlated differences in fitness can result.

The second perceived problem with selection for evolvability as a result of the resulting fitness differences is that it seems to require selection on a level above the individual, since evolvability is often seen as a property of

a lineage or clade. Selection operating at levels higher than the individual have become unpopular, and higher-order selection is known to be a weaker force (Sniegowski and Murphy, 2006). However, as we saw in Section 2.3, evolvability can be seen as a probabilistic property of an individual. Individuals with high evolvability are likely to have ancestors with high evolvability, and are likely to be highly fit because of the beneficial mutations that have occurred in between those generations. Of course, in the context of evolutionary computation, any controversy surrounding selection at levels higher than the individual disappears; an evolutionary computation practitioner cares less about the biological plausibility of an algorithm and more about its performance, and may explicitly build group selection into an algorithm.

A commonly studied mechanism for selection for evolvability is selection for *mutator* genes, which affect parameters like the mutation or recombination rates of the surrounding genes or the genome as a whole. In a changing environment, a mutation to such a mutator gene in an organism that causes the mutation rate to transition from low to high may allow the descendants of that organism to adapt more quickly. Since those organisms also inherit the modified mutator gene, the modified (evolvable) form is selected for. On the other hand, in sexually reproducing organisms the link between the modified mutator gene and the adaptations it causes can be broken by recombination (Sniegowski and Murphy, 2006; Pigliucci, 2008); organisms can inherit the beneficial mutations independently of the modified mutator gene, leading to lower selection for the modified mutator gene.

Conrad (1979b) gave an early argument for a mechanism for the evolution of evolvability. He notes that for adaptation through evolution to work a *gradualism* property is necessary; small changes in the genome (mutations) need to lead to small changes in function, and therefore in fitness. He gives

as an example enzymes—small changes to enzymes leads to small changes in their affinities. Conrad shows that there will be strong selection for any change to the genetic system that creates gradual paths between peaks in the fitness landscape. Central to the argument is that if two adaptive peaks are separated by a distance of m (i.e., if m independent mutations have to happen to get from one peak to the other), and if all of the intermediate mutants are significantly less fit than either peak, then all of the m necessary mutations would have to happen simultaneously in order to get from one peak to the other—an extremely unlikely event.

If, however, just one of the $m!$ paths from one peak to the other were such that each intermediate mutant were not significantly less fit than the one before, then the required mutations need not occur simultaneously, significantly increasing the probability of traversing between the peaks. Conrad calculates, for two settings of the various parameters (such as protein length and the distance between the adaptive peaks), a reduction by a factor of 10^9 and 10^{18} respectively in the amount of time taken to traverse from one peak to the other if such a smooth path is created. Any such change to the genetic system within a lineage is likely to survive by virtue of that lineage reaching adaptive peaks not available to other lineages. Conrad notes that this effect is much stronger than the hitchhiking effect of mutator genes discussed above, by virtue of the magnitude of the evolvability increase that results. Conrad quantifies Maynard Smith's earlier observation that natural selection can only lead from one protein to another if they are connected by a network of functional proteins; if the only way to mutate from one protein to another is via an intermediate, nonfunctional protein, then natural selection is unlikely to lead from one to the other, because the pair of mutations required are unlikely to occur together (Maynard Smith, 1970).

Conrad (1979a) also suggested that selection for evolvability might be more intense in changing environments. Similarly, Wagner and Altenberg (1996) argues that pleiotropic effects of mutations can become decoupled if one trait is under stabilizing selection while another is under directional selection, which may happen due to perturbations in the environment. Under these circumstances the genetic system may change to allow traits that correspond to probable changes in the environment to vary independently from those that do not.

There have been simulation studies of the effects of fluctuating environments on selection for evolvability (Earl and Deem, 2004; Draghi and Wagner, 2008). Often in these simulation experiments the environment changes in a *modular* way. For example, organisms may have to evolve to solve some problem that can be broken down into sub-problems, and it is the sub-problems that change over time. Often, what is shown is that the genetic system changes such that mutations alter the organism in a similarly modular way (Kashtan and Alon, 2005; Kashtan et al., 2007); mutations can alter performance on one sub-problem without affecting performance on another. This of course is a specific case of what it means for an organism to become more evolvable in a fluctuating environment; that the kinds of changes that the organism can undergo due to mutation correspond to probable changes in the environment.

Palmer and Feldman (2011) study in simulation the effect of environments that vary in space rather than in time, and find that such variation creates selection for evolvability. In these simulations, there is a spatial structure on the population, and organisms may migrate slowly. They also find that periodic extinction events in regions of the space intensify selection for evolvability—corroborating earlier arguments by Dawkins and by Kirschner

and Gerhart on the contributions of extinction and migration to the evolution of evolvability—as organisms migrating into newly vacated regions tend to be equally sub-optimal in fitness, but may differ in their ability to adapt to the local environment. Lehman and Miikkulainen (2015) also study the effect of periodic extinctions in a spatially varying environment, and find selection for evolvability results.

As noted above, Dawkins (2003), after experimenting with artificial evolution, was led to the realization that rival embryologies can give vastly different capacities for adaptive evolution. This leads him to conclude that more or less drastic changes to embryology have occurred and been selected for by virtue of their evolvability. He notes that it may seem that selection for evolvability requires group or species selection. However, like Aboitiz (1991), he suggests a kind of lineage selection; an individual with a modified embryology may have offspring that can quickly fill vacant niches. As an example he uses the emergence of segmentation as a significant innovation in embryology; in organisms with segmentation, the form of the segment is specified only once in the genome. Dawkins argues that the first segmented organism was probably less fit as a result, but that it was fit enough to survive, and that segmentation opened up phenotypic possibilities such that its descendants could quickly diverge and occupy vacant niches. As other examples of innovations in embryologies, he gives various types of symmetry and recursion. This argument is similar to that given by Conrad; these changes in embryology survive by virtue of the fitness-increasing mutations they allow within that lineage. Selection for an optimal offspring sex ratio is another example of lineage selection.

Alberch (1991) gives a similar argument; that certain changes in development, such as the emergence of multicellularity or segmentation, open up

regions of phenotype space pregnant with possibility, and that such changes, as long as they survive for long enough, then propagate by virtue of the adaptations they allow. Alberch believes that selection between what he calls 'pattern generating systems' occurred mostly in the pre-Cambrian and Cambrian periods, and that since then the broad details of development have been largely fixed.

Kirschner and Gerhart (1998) provide a two-part explanation for the evolution of evolvability. The first part is essentially the congruent hypothesis; organisms that can tolerate variability in the environment can tolerate the variability caused by mutations. The second part of their explanation is that since such individuals can carry more mutations non-lethally, there will be a greater degree of genetic variance in populations of such individuals. After environmental change, extinctions, or the emergence of new niches, such populations are more likely to contain variants that are well suited to the environment.

2.4.2 Building evolvability into artificial systems

That evolutionary success depends on the generation of suitable genetic variation has long been known in the Evolutionary Computation (EC) community. Every time an evolutionary algorithm is applied to a problem, a practitioner must explicitly decide how potential solutions will be represented and how mutation will operate on represented solutions. That is, the practitioner decides how variation will be introduced, and he presumably hopes that the introduction of variation is such that evolution can solve the problem at hand. In this sense, every EC researcher and practitioner is concerned with how to build evolvability into evolutionary algorithms. Therefore, in this section I will focus on attempts to build the evolution of evolvability into artificial

systems, either by co-opting natural explanations like those described in the previous section, or by more direct means.

Placing the degree of mutational variation under evolutionary control has been studied by Eiben et al. (1999), mostly from an empirical perspective, though important theoretical work in this area has also been done (Rudolph, 2001).

There has been experimental work (in simulation) showing that fluctuating environments, which change over time in some structured way, can lead to the evolution of evolvability (Turney, 1999; Reisinger and Miikkulainen, 2006; O'Neill et al., 2011). Clune et al. (2013) couple a modularly time-varying environment with a fitness function which penalizes neural networks proportionally to the total length of the connections in the network. They show in simulation that this leads to modularity in the networks that mimics the modularity of the problem—and hence leads to evolvability.

In recent simulation experiments in artificial life, the underlying encodings of self-replicators (i.e., the way in which the replicators are encoded in their heritable genetic information) has been allowed to evolve. The hope is that more evolvable encodings (with respect to the environment) will emerge. Many of these simulations were implemented in Avida and its variants. Avida is an artificial life platform in which assembly-like computer programs self replicate (Ofria and Wilke, 2004).

Baugh and McMullin (2013) and Hasegawa and McMullin (2013) have, respectively, designed replicators for the Tierra and Avida self-replication platforms in which part of the self-replicating program is interpreted as genetic information, and another part is interpreted as a decoding mechanism, that decodes the genetic information. By allowing both to evolve, the way that the replicators encode themselves can change over time. Egri-Nagy and

Nehaniv (2003) have implemented a variant of Avida in which each replicating program has its own, different, instruction set, which itself can evolve. The goal is similar; the way that a replicator's behaviour is encoded can change over time, and more evolvable encodings might be discovered.

Webb and Knowles (2014) aimed to study the differing capacities to evolve evolvability between 'non-self-encoding' and 'self-encoding' replicators[4]. In both cases, each replicator implements a decoder that interprets its genetic information, and the decoder itself can evolve over time. In 'self-encoders', the decoder determines the way in which it *itself* is encoded in the genetic information, as illustrated by Figure 2.7. The authors conclude that there may be insufficient selection for evolvability in their simulations to distinguish between the two types of replicator.

Reisinger and Miikkulainen (2006) list some ways in which evolvability has been allowed to evolve in evolutionary algorithms. For example, Ebner et al. (2002) study the evolution of evolvability in neutral networks, in which neighbouring genotypes can encode the same phenotype. Neutral mutations, whose relevance to evolvability was first outlined by Maynard Smith (1970), are those that leave the phenotype unchanged, while possibly changing the phenotypic neighbourhood. Reisinger and Miikkulainen give as other examples evolutionary algorithms with indirect encodings (Stanley and Miikkulainen, 2003) and Estimation-of-Distribution algorithms (Pelikan et al., 2002).

Altenberg (1994) shows that in genetic programming, evolvability can evolve by the implicit selection of blocks of code for what he calls their 'constructional selection', or their ability to improve programs in the population when inserted into them.

[4]Note that this work is not discussed further within this thesis.

There has been surprisingly little effort to directly select for evolvability in evolutionary algorithms and artificial life simulations. An early suggestion to incorporate information about evolvability during selection comes from Reisinger and Miikkulainen (2006). However, the only example that they give is *Estimation of Distribution Algorithms* (EDAs) (Pelikan et al., 2002). I do not believe EDAs select for evolvability in the sense meant in this thesis. EDAs use the current population to try to build a probabilistic model of highly-fit solutions. In a sense, EDAs try to improve the way in which they search the space of solutions based on the solutions and their fitnesses observed so far, but they do not attempt to determine the evolvability of individuals and reproduce those that are more evolvable. I describe research that is more closely related to my own, in that it studies this kind of selection for evolvability, in Chapter 6.

2.5 Summary

The purpose of this chapter has been to give an overview of previous thought on the topic of evolvability in natural and artificial systems. The reader should now have sufficient background information in order to understand the work described in Chapters 3 to 5, which is concerned with explicit selection for evolvability.

We have seen a wide range of definitions and measures of evolvability, a brief discussion of the factors that might cause one individual to differ in its heritable evolvability from another, and a range of explanations for the evolution of evolvability in natural evolutionary systems. We also briefly discussed previous work that aims to build the evolution of evolvability into artificial evolutionary systems. We defer until Chapter 6 a discussion of pre-

vious work that aims to build explicit selection for evolvability into artificial evolutionary systems, as this work is closely related to that of Chapters 4 and 5, and warrants a more detailed discussion and comparison.

Chapter 3

The Simple Evolvability Model

In this chapter, I study selection for evolvability in what I will call the *simple evolvability model*. This work is an extended form of that presented by Webb et al. (2015). I first introduce the model in Section 3.1, list the assumptions of the model in Section 3.2, and then answer the question of the extent to which one should select for evolvability within the model in order to maximize eventual fitness in Section 3.3. The chapter concludes with a list of the limitations of the model.

3.1 The Model

In the model we have a population of N individuals, each with two traits A and B. Trait A is a real-valued number, and trait B is a positive real-valued number. The A value of an individual represents fitness in some environment, and as such it is a value to be maximized. The B value of an individual represents the "evolvability with respect to trait A"; it is the key factor in determining the rate of increase of A in the course of evolution. Here, that means that the B value of an individual determines the standard

deviation of mutations affecting the A value.

In each generation, we rank the population by the value $\gamma A + (1 - \gamma)B$, where the As and Bs have first been normalized by dividing by the standard deviations of those traits in the population. The highest-ranking proportion p become the parents of the next generation; we select with replacement from the set of parents to form the next generation[1]. The weighting parameter γ is under our control, and takes values in the range $[0, 1]$. This parameter represents a trade-off between selecting for fitness and for evolvability. The normalization step describe above is justified in Section 3.A, which is at the end of this chapter.

After the selection step, we mutate the A and B values as follows. We add Gaussian noise to each individual's B value with a constant standard deviation, β. We add Gaussian noise to each individual's A value with standard deviation αB (i.e., a constant times that individual's B value). Since standard deviations must be positive, we prevent the Bs from taking negative values; when a mutation makes a B value negative, we set it to a small positive value ϵ.

In each generation, after mutation, the A and B values decay towards some baseline values. Algorithm 1 describes the process of evolution in this model. Table 3.1 lists the parameters and their roles.

Our question, stated in terms of the model, is as follows. What value of the parameter γ maximizes, at some particular future time t_{end}, the expected mean value of A, the expected mean fitness? I answer this question exactly in the special case that the population size $N = 2$ and the proportion selected as the parents of the next generation $p = 1/2$. I consider this special case in

[1]This truncation selection is employed for analytical convenience only. Note that, depending on the value of the parameter p, this can provide much stronger selection than, for example, roulette wheel selection.

order to remove the effects of indirect selection, as discussed in Section 3.4.

3.2 Assumptions

The simple evolvability model makes the following assumptions.

- Fitness and evolvability can increase without limit, and the probability distribution over the effect of mutations on fitness and evolvability are independent of time and of the current fitness and evolvability values.

- Fitness and evolvability mutations are normally distributed, with zero mean.

- We have perfect knowledge of the evolvability of each individual. In practice, as discussed in Section 4.1, we would have to rely on estimates of the evolvability of an individual or lineage, derived from observations of the effects of past mutations.

- There is no indirect selection for evolvability when selecting for fitness. I have forced this to be the case by considering only the situation where a single parent is selected in each generation. In general, with larger parent population sizes, there would be correlation between fitness and evolvability. In Chapter 4, where we periodically select between populations for an estimate of evolvability, and in between select for fitness within populations, there may be indirect selection for evolvability within the populations.

Algorithm 1 The simple evolvability model

Initialize vector $A(t = 0)$ with N elements of value A_0
Initialize vector $B(t = 0)$ with N elements of value B_0
for each $t \leftarrow 0..t_{end}$ **do**
    $A' \leftarrow A(t)/\text{std\_dev}(A(t))$
    $B' \leftarrow B(t)/\text{std\_dev}(B(t))$
    Sort $A(t)$, $B(t)$ by the corresponding value $\gamma A' + (1 - \gamma)B'$
    for each $n \leftarrow 1..N$ **do**
        $i \leftarrow$ random variate drawn from the discrete uniform distribution $[1, \lfloor pN \rfloor]$
        $M_A \sim \mathcal{N}(0, \alpha^2 B(t)_i^2)$
        $M_B \sim \mathcal{N}(0, \beta^2)$
        $A(t + 1)_n \leftarrow k_A(A(t)_i + M_A) + (1 - k_A)R_A$
        $B(t + 1)_n \leftarrow k_B(B(t)_i + M_B) + (1 - k_B)R_B$
        if $B(t + 1)_n < 0$ **then**
            $B(t + 1)_n \leftarrow \epsilon$

3.3 Optimal Constant γ

In this section I find the constant value of the selection trade-off parameter γ which maximizes the expected eventual fitness value. Since we restrict ourselves to the special case that the population size $N = 2$ and the proportion kept during selection $p = 1/2$ (i.e., there is a single parent in each generation), we can restate the problem so that we only have to keep track of one A value and one B value per generation; let $A(t), B(t)$ be the A and B values of the *parent* for generation t, with $A(0) = A_0$, $B(0) = B_0$.

In each generation, we duplicate the parent, mutate both copies, and then, after normalizing by dividing the As and Bs by their population standard deviations, we select as the parent for the next generation the individual with the maximum value of $\gamma A + (1 - \gamma)B$. Since the parents are identical before mutation, we are essentially selecting between rival mutations.

The normalization step means that, as far as the selection operator is concerned, all mutations have a standard deviation of 1. We can achieve

Table 3.1: Parameters of the model.

N	—	The population size. Here we set $N = 2$.
p	—	The proportion of individuals chosen by truncation selection as parents of the next generation. Here we set $p = 1/2$.
A_0	—	The initial value of the trait A in the population.
B_0	—	The initial value of the trait B in the population. Non-negative.
α	—	A parameter adjusting the standard deviation of A mutations. Non-negative.
β	—	A parameter adjusting the standard deviation of B mutations. Non-negative.
k_A	—	The A decay rate.
R_A	—	The value towards which A decays.
k_B	—	The B decay rate.
R_B	—	The value towards which B decays.
t_{end}	—	The time at which we want to maximize the expected mean value of A, with respect to the parameter γ. Positive integer.
γ	—	A parameter under our control representing a trade-off between selection for the traits A and B. In the range $[0, 1]$.
ϕ	—	An alternative parameter giving the trade-off between selecting for the traits A and B. In the range $[0, \frac{\pi}{2}]$.

the same result by drawing four 'normalized' mutations from the standard normal distribution $\mathcal{N}(0, 1)$. M_{A_1} and M_{A_2} are the normalized mutations affecting the As, and M_{B_1} and M_{B_2} are the normalized mutations affecting the Bs. We will then select the pair of mutations with the largest value of

$\gamma M_A + (1-\gamma)M_B$, and multiply each by the desired mutational standard deviation, undoing the normalization step. We then apply these mutations to the parent to get the A and B values of the parent of the next generation.

Let $\langle M_A^+, M_B^+ \rangle$ be the $\langle M_A, M_B \rangle$ pair with the maximum value of $\gamma M_A + (1-\gamma)M_B$. That is,

$$\langle M_A^+, M_B^+ \rangle = \langle M_{A_m}, M_{B_m} \rangle, \text{ where} \tag{3.1}$$
$$m = \underset{i \in \{1,2\}}{\operatorname{argmax}} \big(\gamma M_{A_i} + (1-\gamma)M_{B_i}\big).$$

The two components in the sum in Equation (3.1) are distributed as

$$\gamma M_{A_{1,2}} \sim \mathcal{N}(0, \gamma^2) \tag{3.2}$$
$$(1-\gamma)M_{B_{1,2}} \sim \mathcal{N}(0, (1-\gamma)^2). \tag{3.3}$$

Using Equations (3.24) and (3.25) from the Section 3.B, we obtain the expected values of these components in the pair with the maximum sum, which are

$$E[\gamma M_A^+] = \frac{\gamma^2}{\sqrt{\gamma^2 + (1-\gamma)^2}\sqrt{\pi}} \tag{3.4}$$
$$E[(1-\gamma)M_B^+] = \frac{(1-\gamma)^2}{\sqrt{\gamma^2 + (1-\gamma)^2}\sqrt{\pi}}. \tag{3.5}$$

Due to mutation and selection, A will increase by $\alpha B(t) M_A^+$, and B will increase by βM_B^+. These are the 'un-normalized' mutations, which have the

expected values (taking the expectation over the possible values of M_A^+, M_B^+)

$$E[\alpha B(t) M_A^+] = \frac{\gamma}{\sqrt{\gamma^2 + (1-\gamma)^2}\sqrt{\pi}} \alpha B(t) \tag{3.6}$$

$$E[\beta M_B^+] = \frac{1-\gamma}{\sqrt{\gamma^2 + (1-\gamma)^2}\sqrt{\pi}} \beta. \tag{3.7}$$

As a result of mutation and selection, A and B each increase by a non-linear function of γ multiplied by the standard deviation of A and B mutations. This function is shown in Figure 3.1, and is given by

$$f(\gamma) = \frac{\gamma}{\sqrt{\gamma^2 + (1-\gamma)^2}}. \tag{3.8}$$

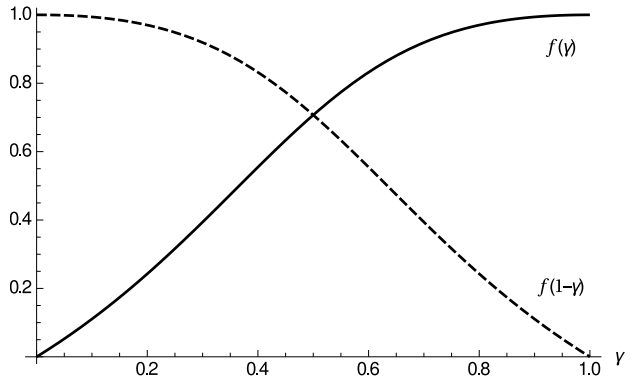

Figure 3.1: The trade-off between selecting for traits A and B in a population of size 2. The function $f(\gamma)$ gives the expected increase in trait A due to selection, in units of the standard deviation of A mutations. The function $f(1-\gamma)$ plays the same role for the expected increase in trait B.

Since $\alpha B(t) M_A^+$ and βM_B^+ represent the change in A and B due to mutation and selection, once we account for fitness and evolvability decay we obtain the following recurrence relations relating A and B in generation t to

generation $t+1$.

$$A(t+1) = k_A(A(t) + \alpha B(t) M_A^+(t)) + (1 - k_A)R_A \tag{3.9}$$

$$B(t+1) = k_B(B(t) + \beta M_B^+(t)) + (1 - k_B)R_B. \tag{3.10}$$

The expected value of $A(t)$ and $B(t)$ (now taking the expectation over the mutation events in *every* generation) satisfy the recurrence relations

$$E[A(t+1)] = k_A(E[A(t)] + f(\gamma)\frac{\alpha E[B(t)]}{\sqrt{\pi}}) + (1 - k_A)R_A \tag{3.11}$$

$$E[B(t+1)] = k_B(E[B(t)] + f(1-\gamma)\frac{\beta}{\sqrt{\pi}}) + (1 - k_B)R_B. \tag{3.12}$$

In our formulation of the problem so far, we have ranked the population by $\gamma A + (1-\gamma)B$ during selection. An alternative and natural parameterization, which I will return to, is to rank the population by $\sin(\phi)A + \cos(\phi)B$, where $\phi \in [0, \frac{\pi}{2}]$, in which case the normalization term (the denominator of Equation (3.8)) disappears. The parameter ϕ is more meaningful than γ, giving the angle in $A - B$ space that the population moves in as a result of selection. With this parameterization, the recurrence relations in Equations (3.11) and (3.12) become

$$E[A(t+1)] = k_A(E[A(t)] + \sin(\phi)\frac{\alpha E[B(t)]}{\sqrt{\pi}}) + (1 - k_A)R_A \tag{3.13}$$

$$E[B(t+1)] = k_B(E[B(t)] + \cos(\phi)\frac{\beta}{\sqrt{\pi}}) + (1 - k_B)R_B. \tag{3.14}$$

Returning to the γ-parameterization, solving the recurrence relations in Equations (3.11) and (3.12) for $E[A(t)]$ and maximizing the resulting expression[2] with respect to γ yields a single solution in the range $[0, 1]$, which

[2]This expression was calculated using the Mathematica symbolic computation software (Wolfram Research, Inc., 2016) and verified by simulating the evolutionary process

is

$$\gamma^* = \frac{3}{4} + \frac{1}{4}\big(z - \sqrt{z+3}\sqrt{z-1}\big), \text{ where} \tag{3.15}$$
$$z = \sqrt{1 + \frac{8}{\pi}w^2}, \text{ where}$$
$$w = \frac{\beta k_B}{(k_B - 1)(-\frac{B_0(k_A-1)(k_B^t - k_A^t)}{k_B + k_B^t(k_A-1) - k_A + k_A^t - k_A k_B t} + k_B R_B)} \quad .$$

We can get a shorter expression by switching to the alternative ϕ-parameterization. In this case, $\phi = \arctan(\frac{\gamma}{1-\gamma})$, so the optimal value of ϕ is

$$\phi^* = \arctan(1 + \frac{4}{z-1}) \quad , \tag{3.16}$$

where z is as defined in Equation (3.15). Note that $z(w)$ takes values in the range $[1, \infty)$, and so γ^* takes values in the range $(\frac{1}{2}, 1]$. In other words, within this model, it's never a good strategy to select more for evolvability than for fitness.

The optimal value of the selection tradeoff parameter, γ^*, does not depend on R_A, the value to which fitness decays. We can see that the optimal value γ^* decreases with β and increases with B_0. In words, for larger initial evolvability values we should select more for fitness, and with a greater capacity to increase evolvability through mutation and selection we should select more for evolvability. The optimal value is a complicated function of the remaining parameters.

In the case that $k_B = 1$ or $R_B = 0$—i.e., if evolvability does not decay, or

described in this chapter for a large number of parameter settings and comparing the mean fitness achieved at time t to the value predicted by that expression.

it decays towards zero—then in the limit as t goes to infinity

$$\lim_{t\to\infty} \gamma^* = \frac{1}{2} \quad . \tag{3.17}$$

That is, unless evolvability decays to a nonzero value, if we care about maximizing the expected value of fitness in the extreme long term, then the optimal value of the selection trade-off parameter is $\frac{1}{2}$.

If evolvability decays towards some non-zero value—if $k_B \neq 1$ and $R_B \neq 0$—then as t goes to infinity

$$\lim_{t\to\infty} w = \frac{\beta}{(k_B - 1)R_B} \quad . \tag{3.18}$$

We can see from this that, in the case that evolvability decays towards some non-zero value, if we care about maximizing the expected value of fitness in the extreme long term, then the optimal value of the selection trade-off parameter does not depend on the rate at which fitness decays, but only upon the standard deviation of evolvability mutations β, the rate at which evolvability decays k_B, and the value towards which it decays R_B. We can see that the long term optimal value decreases, meaning we should select more strongly for evolvability, if the standard deviation of evolvability mutations increases. We should select less for evolvability if evolvability decays towards a larger value R_B, and should select more for evolvability if evolvability decays more quickly.

In the following I discuss the optimal value γ^* in the simple case that $k_A = k_B = 1$, i.e., neither evolvability nor fitness decay. In this case, the optimal value of the trade-off parameter, γ^*, depends only on B_0, β, and t_{end}, and is an increasing function of $B_0/(\beta(t_{end} - 1))$. Figure 3.2 shows γ^* as a function of $B_0/(\beta(t_{end} - 1))$, while Figure 3.3 shows γ^* as a function of

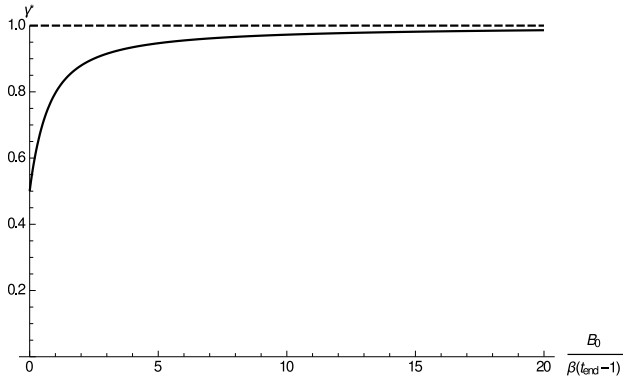

Figure 3.2: The optimal value of the trade-off parameter γ as an increasing function of $B_0/(\beta(t_{end}-1))$. As B_0 becomes large, the optimal value asymptotically approaches 1.

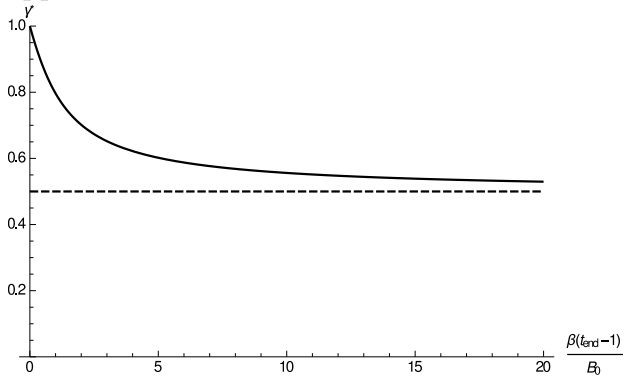

Figure 3.3: The optimal value of the trade-off parameter γ as a decreasing function of $\beta(t_{end}-1)/B_0$. As β or t_{end} become large, the optimal value asymptotically approaches 1/2.

the reciprocal $\beta(t_{end}-1)/B_0$. Both are shown so that both asymptotes are clear.

That γ^* depends on this quantity makes sense; it is the ratio between the initial evolvability B_0 and βt_{end}, which is related to the amount that evolvability can increase in the course of evolution in the time given. If the initial evolvability is large compared to the capacity to increase evolvability, then it pays off to focus more on increasing the trait A. As we look further

to the future and t_{end} becomes large, the initial evolvability value has less of an effect and γ^* tends towards $\frac{1}{2}$.

Figure 3.4 shows the optimal value of γ found by grid search and simulation for a range of values of B_0, β, and t_{end}. For each setting of the parameters, we simulate the evolutionary process of the model for 100 values of γ evenly spaced between zero and one. We plot the value of γ with the highest mean value of A at time t_{end} measured over one hundred thousand trials (the low population size makes the outcome noisy). The simulation results closely agree with the predicted value obtained here, verifying the result[3].

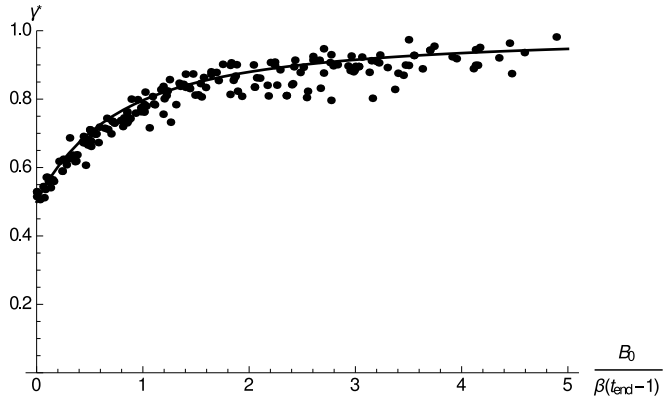

Figure 3.4: A comparison between the optimal value of γ obtained by grid search and simulation and the predicted result. The evolutionary process was simulated for a large number of combinations of values for B_0, β, and t_{end}. For each setting of the parameters, We try 100 values of γ, evenly spaced between zero and one. The scatter plot shows, for each combination of parameter values, the value of γ that achieved the highest mean value (over 100 000 trials) of A at time t_{end}. The line shows the result predicted by the model.

Figure 3.5 shows, for a particular setting of the parameters, the expected

[3]There is a small discrepancy, perhaps because our result does not account for the fact that, after a mutation, we set negative B values to small positive values. This manifests when B_0 is small enough that B is small compared with β in the initial generations.

value of A over time for three strategies; setting $\gamma = 1$ (so that only A is selected for), setting $\gamma = 1/2$ (so that we select equally for A and B), and setting $\gamma = \gamma^*$ (the optimal value). It can be seen that the γ^* strategy dominates. Figure 3.6 shows the same with a different setting of the parameter β. Note that for the γ^* strategy, the plots do not show A over time for a particular value of γ; for each time t, the plot shows the expected value of A when using the (constant) value of γ that maximizes $A(t)$.

Note that although selecting for an optimally weighted sum of fitness and evolvability can lead to increased fitness over selecting for fitness alone by a particular time t—as shown in Figures 3.5 and 3.6—selecting for the weighted sum will lead to *decreased* fitness compared with selection for fitness alone in the short term.

3.4 Limitations and Conclusion

This chapter introduced the *simple evolvability model* (SEM) in order to study the extent to which we ought to select for evolvability in order to maximize long-term fitness. The model assumes that evolvability information is accurate and comes at no cost in terms of fitness evaluations. The findings of this chapter are as follows.

1. I obtain an expression for the optimal selection trade-off parameter γ in terms of the model parameters.

2. The expression reveals that, regardless of the model parameters, it never pays off to select more for evolvability than for fitness.

3. In the limit that the number of generations goes to infinity—i.e., when we are concerned with extreme long-term fitness—in the case that

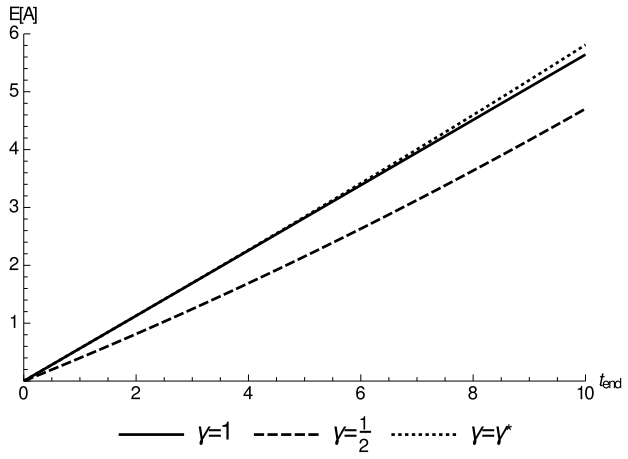

Figure 3.5: The expected value of A over time for three strategies for setting γ. $A_0 = 0, \alpha = 1, B_0 = 1, \beta = 0.1$.

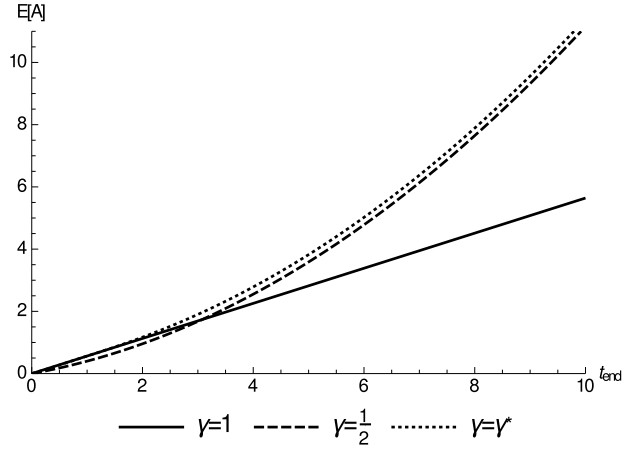

Figure 3.6: The expected value of A over time for three strategies for setting γ. $A_0 = 0, \alpha = 1, B_0 = 1, \beta = 1$.

evolvability does not decay to some non-zero value, then the optimal value of the selection trade-off parameter γ goes to $1/2$, meaning that we should select equally for fitness and evolvability.

4. In the case that evolvability does decay to some non-zero value, I derive an expression for the optimal value of γ in terms of a small subset of

the model parameters in order to maximize achieved extreme long-term fitness.

The assumptions of the model are listed in Section 3.2. In the following I discuss some of the assumptions and the effect on our conclusions if they are relaxed.

In the above we study the simple case of a population size of $N = 2$ to eliminate indirect selection effects. If the population is of size N, and we produce the next generation by selecting with replacement from the top proportion p of individuals, ranked by $\gamma A + (1 - \gamma)B$, then after the first generation, the expected increase in traits A and B due to mutation and selection becomes harder to calculate for two reasons. The first is that the trait variances in the population depend on mutations accumulated over multiple generations; because more than one parent is selected in each generation there will be residual variation from the previous generation. This residual (post-selection) variation will not be normally distributed. The result is that the traits A and B will no longer be normally distributed, but will be skewed by an amount depending on the trade-off parameter γ and the proportion kept during selection p. The trait distributions will change over time, approaching some equilibrium shape.

The second problem is that, because more than one parent is selected in each generation, correlation builds up between the A and B values in the population; individuals selected for having large A values are likely to have inherited large B values. The result of this correlation is that there is indirect selection for trait B when selecting for trait A.

Figure 3.7 shows (with $N = 100, p = 1/2$), as functions of γ, the measured mean increase in traits A and B during selection in generation 10, in units of the mutational standard deviation of A and B, respectively, with neither

fitness or evolvability decaying, i.e., $k_A = 1$ and $k_B = 1$. Figure 3.8 shows the same in generation 50. The asymmetry is due to indirect selection for trait B, and the functions change over time because the population trait distributions and the correlation between the traits are changing over time. Compare these with the stationary (in time) and symmetric functions giving the expected per-generation increase of traits A and B for a population of size two, shown in Figure 3.1. Without an exact expression for the expected increase of traits A and B due to selection in a larger population, we cannot deduce the optimal value of the trade-off parameter γ.

Figure 3.9 shows the optimal value of γ found by grid search and simulation (as in Figure 3.4), with a population size $N = 100$ and $p = 1/2$. The exact predicted result derived for a population size of $N = 2$ is shown for comparison.

It would be interesting to see an analysis of this model which incorporates indirect selection effects.

The model assumes that we know the evolvability of each individual, and that it doesn't cost us any fitness evaluations to get that information. In practice, we can only learn about the evolvability of an individual by sampling from that individual's offspring fitness distribution, i.e., by producing offspring from that individual. We can estimate the evolvability of an individual by calculating the sample standard deviation of the fitness of its offspring. This estimate is noisy, and the variance of the estimate will depend on the size of the offspring population.

The increase in evolvability due to mutation and then selection for evolvability, as stated in this chapter, is an overestimate for two reasons; the first is that part of our fitness evaluation budget must be spent in order to calculate evolvability estimates. The second reason is that due to noise in the

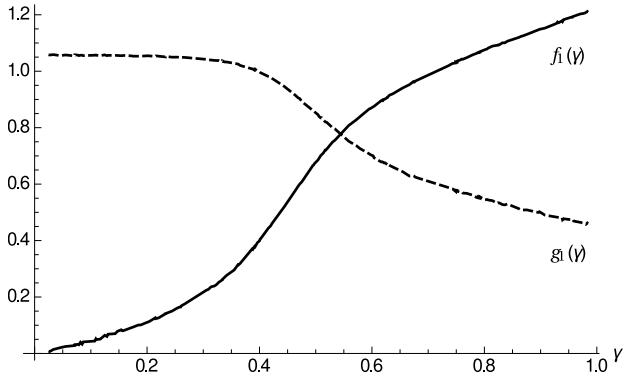

Figure 3.7: The function $f_1(\gamma)$ shows the measured mean increase in trait A in generation 10 in units of the standard deviation of A mutations. The function $g_1(\gamma)$ plays the same role for trait B. The functions are asymmetric; there is indirect selection for B when selecting for A. $N = 100$.

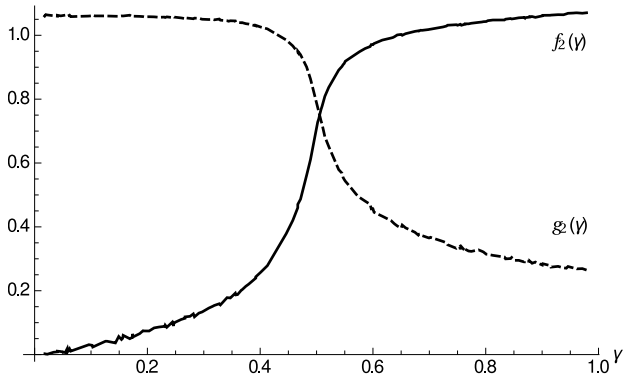

Figure 3.8: The function $f_2(\gamma)$ shows the measured mean increase in trait A in generation 50 in units of the standard deviation of A mutations. Function $g_2(\gamma)$ plays the same role for trait B. The functions are not the same as those for generation 10, and the functions are asymmetric. Compare with Figure 3.1. $N = 100$.

evolvability estimates, when we select for evolvability we may select the less evolvable individual. We will return to these considerations in Chapter 4, in which the effect of noisy estimates is further explored and new evolvability selection strategies are proposed. Although this limitation of the model is the motivation for the work in Chapters 4 and 5, it would be interesting to see an

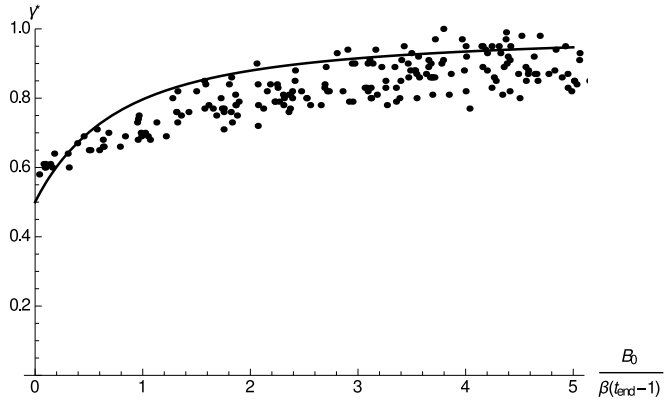

Figure 3.9: A comparison between the optimal value of γ obtained by grid search and simulation (as in Figure 3.4, with population size $N = 100$) and the exact predicted result (with $N = 2$).

extension of the model in which the cost of obtaining accurate evolvability estimates is included.

In the analysis above, we were restricted to considering constant values of the selection trade-off parameter γ. However, the optimal constant value might be sub-optimal, and we might achieve better eventual fitness values if we allow $\gamma(t)$ to vary, being a function of the current generation number t. There are certain properties we should expect from the optimal value of the function $\gamma(t)$; as early increases in evolvability have a greater effect on eventual evolvability than later increases, and because any early increases in evolvability will manifest more strongly whenever we are selecting more strongly for fitness, we might expect $\gamma(t)$ to be monotonically increasing with t. Finding the optimal $\gamma(t)$ might be difficult. Since the problem is that of obtaining a function which maximizes an expression containing that function within a sum, the correct approach may be related to the calculus of variations.

The goal in this chapter has been to maximize the expected population average fitness at time t, and increasing the B value of an individual early on in the simulation increases the expected mean A value of its surviving descendants. However, from the point of view of any given individual, having a large B value is a risky strategy; if an individual produces N offspring, the probability that all of its offspring have fitness values less than or equal to the parent fitness minus $\frac{B}{10}$ is around 0.21 if $N = 2$, 0.1 if $N = 3$, and 0.045 if $N = 4$[4]. If an individual has an extremely large B value (relative to the rest of the population), then with probability roughly equal to 0.5^N where N is the number of offspring, none of those offspring will survive. An individual with a large B value has a greater expected value for the fitness of their surviving offspring, but also has a greater degree of noise in its offspring fitnesses and a larger probability that none of its offspring will survive.

[4]These probabilities are estimates calculated by simulating the process 1000 times.

Chapter Appendix

3.A Justification for the Normalization Step

In the simple evolvability model, the A and B values used during selection are normalized by dividing by the population standard deviation of those values. The reason we include the normalization step is that, without it, as the variance of one of the traits in the population becomes large, evolution stops acting on the other trait, regardless of the value of γ. This is illustrated in Figures 3.10 and 3.11. These show, for a large population with normally distributed A and B traits, the expected increase in the population mean values of A and B if we select the top half (i.e., $p = 0.5$) by value $\gamma A + (1 - \gamma)B$ with $\gamma = 0.5$, and where the standard deviation of the A values is 1 and the standard deviation of the B values is parameterized by β. Without normalization, as Figure 3.10 shows, the expected increase of each trait (denoted ΔA and ΔB) is a nonlinear function of both of the trait standard deviations. As the standard deviation of the Bs, β, tends to infinity, the expected increase in trait A due to selection tends to zero. With normalization, as Figure 3.11 shows, the expected increase of each trait is proportional to the standard deviation of just that trait.

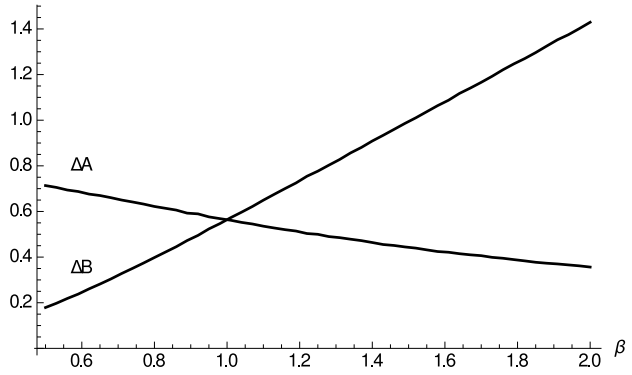

Figure 3.10: Without the normalization step, when we select for $A + B$, the expected increase in the mean value of each trait (ΔA and ΔB) is a nonlinear function of the standard deviations of both traits in the population. As the standard deviation of the Bs, β, tends to infinity, trait A stops being selected for.

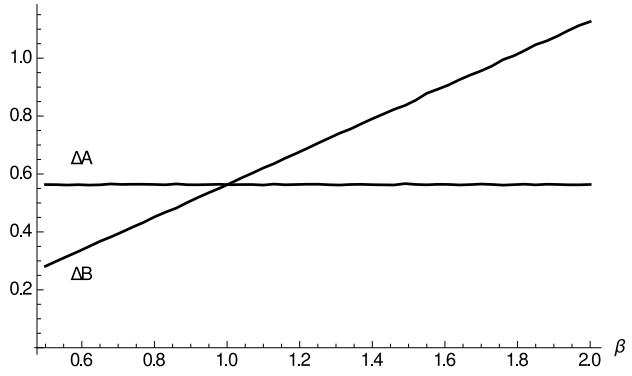

Figure 3.11: With the normalization step, when we select for $A + B$, the expected increase in the mean value of each trait (ΔA and ΔB) is a linear function of the standard deviation of just that trait in the population.

3.B The Expected Maximum Values of the Maximum-Sum Pair

In this section I obtain the result used in order to obtain Equations (3.4) and (3.5). If we have two random variables A and B both distributed as

$\mathcal{N}(0, \sigma^2)$, then the maximum of A and B has the expected value

$$\begin{aligned} E[max(A,B)] &= \int_{-\infty}^{\infty} a\phi(a) \int_{-\infty}^{a} \phi(b)\,\mathrm{d}b\,\mathrm{d}a \\ &\quad + \int_{-\infty}^{\infty} b\phi(b) \int_{-\infty}^{b} \phi(a)\,\mathrm{d}a\,\mathrm{d}b \\ &= 2\int_{-\infty}^{\infty} a\phi(a) \int_{-\infty}^{a} \phi(b)\,\mathrm{d}b\,\mathrm{d}a \quad , \end{aligned} \tag{3.19}$$

where $\phi(x)$ is the pdf of the distribution. This can be understood as follows. We integrate over the possible values of A, multiplying the probability of getting that value by the probability that the B value is less than it (i.e., the probability that the A is the maximum of the pair). For each possibility we multiply by the value of A to get the expected value. We then do the same thing for the case where the B value is the greater of the pair. Because A and B have the same distributions, these integrals are equal, so we evaluate it once and double the result. Evaluating the integral gives the result

$$E[max(A,B)] = \frac{\sigma}{\sqrt{\pi}} \quad . \tag{3.20}$$

In the previous section we make use of the following result giving the expected values of the pair of numbers (out of two pairs) that has the maximum sum. Suppose we have four normal random variables distributed as

$$\begin{aligned} A_1 &\sim \mathcal{N}(0, \sigma_A^2), \quad B_1 \sim \mathcal{N}(0, \sigma_B^2) \\ A_2 &\sim \mathcal{N}(0, \sigma_A^2), \quad B_2 \sim \mathcal{N}(0, \sigma_B^2) \quad . \end{aligned} \tag{3.21}$$

Let $\langle A_m, B_m \rangle$ be the $\langle A, B \rangle$ pair with the maximum sum. That is,

$$m = \underset{i \in \{1,2\}}{\operatorname{argmax}} \, (A_i + B_i) \quad . \tag{3.22}$$

The expected value of A_m is given by

$$E[A_m] = 2\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} a\phi_A(a)\phi_B(b)\Phi(a+b)\,\mathrm{d}a\,\mathrm{d}b, \tag{3.23}$$
$$\text{where } \Phi(a+b) = \int_{-\infty}^{a+b} \phi_{A+B}(c)\,\mathrm{d}c \quad ,$$

and where $\phi_A(x)$ is the pdf of each of $A_{1,2}$, $\phi_B(x)$ is the pdf of each of $B_{1,2}$, and ϕ_{A+B} is the pdf of each of $A_1 + B_1$ and $A_2 + B_2$.

In words, we integrate over the possible values of A_1 and B_1, multiplying the joint probability of getting those values by the probability that the sum of the *other* $\langle A, B\rangle$ pair takes a value less than $A_1 + B_1$, and we multiply by the A_1 value to get its expected value. We then integrate over the possible values of A_2 and B_2 (for the case where the sum of the second pair is greater than the sum of the first), which gives the same integral again. Adding the two integrals together gives the expected value of A_m.

Evaluating the above integral gives the expression[5]

$$E[A_m] = \frac{\sigma_A^2}{\sqrt{\sigma_A^2 + \sigma_B^2}\sqrt{\pi}} \quad , \tag{3.24}$$

and by symmetry the expected value of B_m is

$$E[B_m] = \frac{\sigma_B^2}{\sqrt{\sigma_A^2 + \sigma_B^2}\sqrt{\pi}} \quad . \tag{3.25}$$

[5]This expression was calculated using the Mathematica symbolic computation software (Wolfram Research, Inc., 2016) and verified by simulating the process described in this section for a large number of combinations of values for σ_A and σ_B, and comparing the exact value obtained for the expected values of the components of the maximum-sum pair with the mean value obtained in simulation.

Chapter 4

Episodic Group Selection for Evolvability

The previous chapter explores the question of how to select optimally for evolvability when certain assumptions are satisfied. One assumption of that chapter is that we know the evolvabilities of individuals exactly. As Section 3.4 briefly discussed, in practice we will usually have to rely on an estimate of the evolvability of an individual calculated from that individual's offspring fitnesses. The algorithms compared in this chapter will be given a fixed budget of fitness evaluations, in order that they may be compared, and in order to select for an estimate of evolvability an algorithm must use some of its fitness evaluation budget. Depending on the size of the offspring population this estimate can be poor and the result of selecting for poor estimates of evolvability is that evolvability increases much more slowly.

This chapter describes and evaluates an algorithm that attempts to use its fixed budget of fitness evaluations more efficiently to calculate and select for estimates of evolvability. I explore in Section 4.1 how the results obtained in Chapter 3 are affected by selection for estimates of evolvability that are

noisy due to sampling errors. I introduce in Section 4.2 a general strategy, *episodic group selection*, in which multiple populations are maintained. Selection for fitness occurs within those populations, with selection for evolvability occurring periodically between the populations based on sequences of noisy evolvability estimates calculated from offspring fitnesses. The novel aspect of this algorithm is that the sequence of noisy estimates of evolvability are combined with a model of how evolvability changes over time in order to produce less noisy estimates. I describe in Section 4.5 some heuristics for deciding when to stop selecting for evolvability—and stop maintaining multiple populations—in order to use the fitness evaluation budget more efficiently. I compare the EGS algorithm to selection for fitness alone on four fitness functions in experiments described in Sections 4.7 and 4.8.

The reader may wish to refer ahead to Chapter 6, which reviews work that is closely related to that described in this chapter, before or concurrently with with chapter.

4.1 The Effect of Sampling Noise on Evolvability Selection

In Chapter 3 we consider a selection strategy in which, in each generation, we select for a weighted sum of fitness and evolvability. We assume in that chapter that we know the evolvabilities of each individual exactly. In this section, we consider the consequences for the results obtained in that chapter if we must select for an estimate of the evolvability of each individual calculated from its offspring fitnesses. We will consider a modified version of the model described in Chapter 3, in which we select for a weighted sum of fitness and *estimated* evolvability, where the true evolvability of an individual is

the standard deviation of its offspring fitness distribution and the estimated evolvability of an individual is the sample standard deviation of the fitnesses of its offspring. We will find that, unless the offspring population size is very large, evolvability estimates are noisy, as are rankings of individuals by their estimated evolvabilities, and so evolvability increases more slowly as a result of selecting for estimates.

In order to calculate evolvability estimates, we produce a *poll offspring population* of size N' for each individual, and calculate from it the standard deviation of the offspring fitnesses. (It is a 'poll' offspring population because the individuals in the population can never be selected for; they are only used to estimate the evolvability of their parents.) Figure 4.1 illustrates this selection strategy with a population size of two.

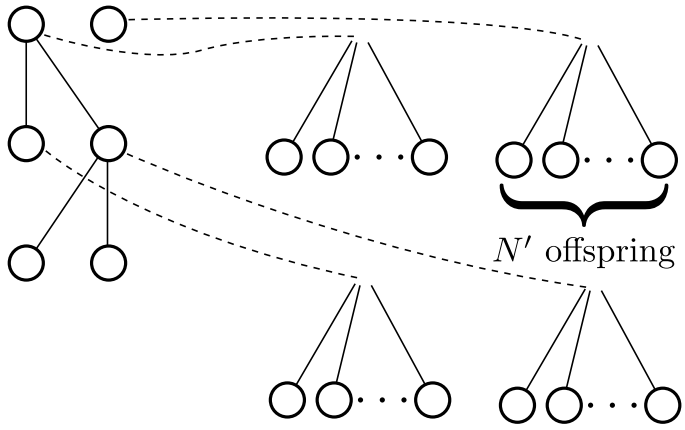

Figure 4.1: The left-hand side of the figure illustrates a population of size two over three generations. Solid lines represent parent-offspring relationships, with offspring appearing below parents. In each generation, we create a 'poll' offspring population for each individual of size N'. Each parent is connected to its poll offspring population in the figure by a dotted line. This offspring population is used to estimate the evolvability of the parent and is then discarded. We then select for a weighted sum of fitness and estimated evolvability.

Under these circumstances, the expected increase of the population mean

value of evolvability due to selection as reported in Chapter 3 is optimistic for two reasons. The first is that because offspring populations are produced to estimate the evolvabilities of the parents, some of our fixed budget of fitness evaluations must be used in order to make these estimates; the number of fitness evaluations per generation has increased from N to NN', where N is the population size and N' is the poll offspring population size. The second reason is that, depending of the poll offspring population size, the evolvability estimates can be poor.

In Chapter 3, I noted that if we select purely for evolvability by selecting the maximum from a population size of two, where the two evolvability values are drawn from a normal distribution with the same mean and with variance β^2, then the expected increase of the population mean value of evolvability due to selection is $\frac{1}{\sqrt{\pi}}\beta$.

What happens if instead of selecting the individual with the greater evolvability, we select the individual with the greater estimated evolvability? The sample variance of a sample of size N from a normal distribution with variance σ^2 is distributed as

$$\frac{N-1}{\sigma^2}S^2 \sim \chi^2_{N-1} \quad . \tag{4.1}$$

This shows that a constant ($\frac{N-1}{\sigma^2}$) multiplied by the sample variance follows a chi-square distribution with parameter $k = N-1$ (Knight, 2000, Proposition 2.11). It follows from this that the sample variance itself follows a gamma distribution as follows.

$$S^2 \sim \Gamma\left(\frac{N-1}{2}, \frac{2\sigma^2}{N-1}\right) \quad . \tag{4.2}$$

If one individual has evolvability $B_1 = B$ and another has evolvability

$B_2 = B + \Delta$ (with Δ positive), then the sample variances $\hat{B}_1^2$, $\hat{B}_2^2$ of fitness from offspring populations of size N' are therefore random variables following gamma distributions as follows.

$$\hat{B}_1^2 \sim \Gamma\left(\frac{N'-1}{2}, \frac{2B^2}{N'-1}\right) \tag{4.3}$$

$$\hat{B}_2^2 \sim \Gamma\left(\frac{N'-1}{2}, \frac{2(B+\Delta)^2}{N'-1}\right) \quad . \tag{4.4}$$

We can obtain an expression for the probability p that the second sample variance is larger than the first by computing the integral

$$p = \int_0^\infty \int_0^{v_2} f_1(v_1) f_2(v_2)\, \mathrm{d}v_1\, \mathrm{d}v_2 \quad , \tag{4.5}$$

where f_1 and f_2 are the probability density functions of the sample variances. This is also the probability that the second sample standard deviation is larger than the first, and therefore that the individual with the greater estimated evolvability—the individual selected—has the greater true evolvability.

This probability is given by

$$\frac{2^{(N'-2)}}{\sqrt{\pi}}\left(1+\frac{\Delta}{B}\right)^{(N'-1)}\Gamma\left(\frac{N'}{2}\right)\ {}_2\tilde{\mathrm{F}}_1\left(\frac{N'-1}{2}, N'-1; \frac{N'+1}{2}; -\left(1+\frac{\Delta}{B}\right)^2\right) \quad , \tag{4.6}$$

where ${}_2\tilde{\mathrm{F}}_1(a, b; c; z)$ is the regularized Gaussian hypergeometric function[1].

We can see that this probability is a function of two variables: the poll

[1]This expression was calculated using the Mathematica symbolic computation software (Wolfram Research, Inc., 2016) and verified by simulating the process described in this section for a large number of combinations of the parameters N, B, and Δ. For each setting of the parameters, over a large number of trials, the proportion of times that the individual selected has the greater true evolvability value in simulation agrees with the exact value obtained here for the probability of that being the case.

population size N' and the ratio $\frac{\Delta}{B}$ between the difference in evolvabilities and the baseline evolvability. The probability p takes values in the range $[0.5, 1.0)$. Figures 4.2 and 4.3 show the probability p as a function of N' for three fixed values of $\frac{\Delta}{B}$. Figures 4.4 and 4.5 show p as a function of $\frac{\Delta}{B}$ for three fixed values of N'.

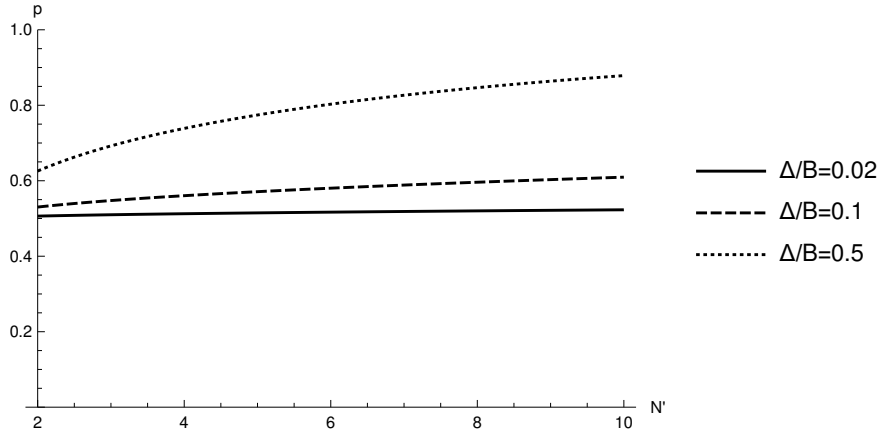

Figure 4.2: The probability of correctly selecting the individual with the greater true evolvability based on an estimate calculated from a population size of N', plotted as a function of N' for three fixed values of the relative evolvability difference between the two individuals.

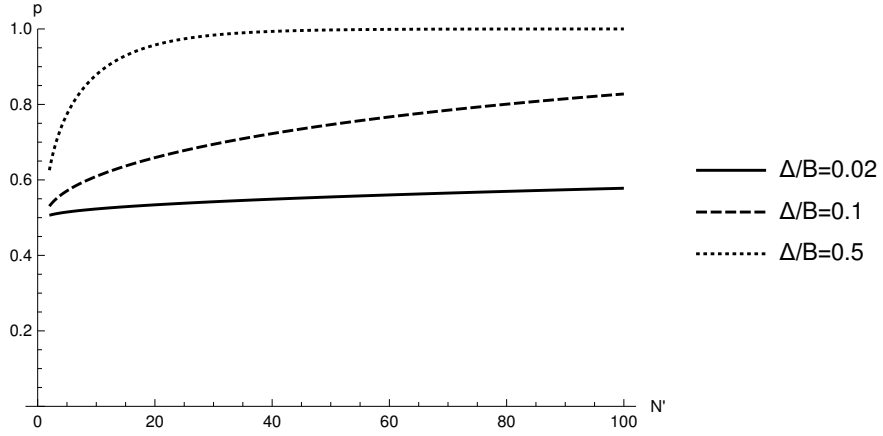

Figure 4.3: The same as Figure 4.2, shown over a larger range of N'.

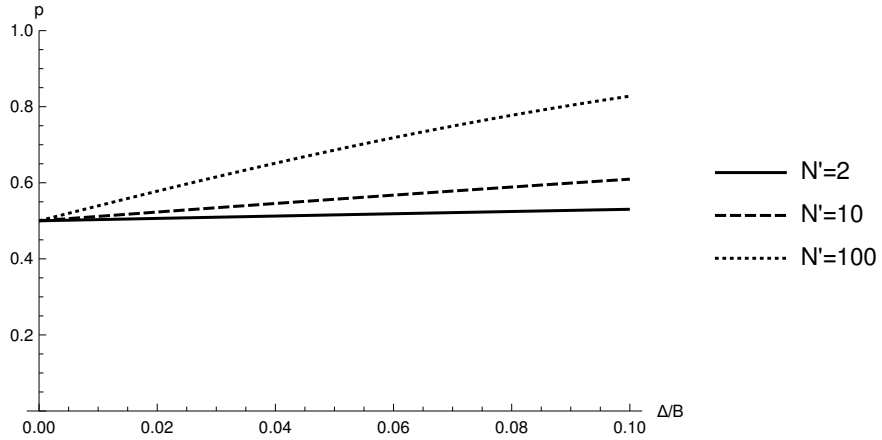

Figure 4.4: The probability of correctly selecting the individual with the greater true evolvability, plotted as a function of the relative fitness difference $\frac{\Delta}{B}$ for three fixed values of N'.

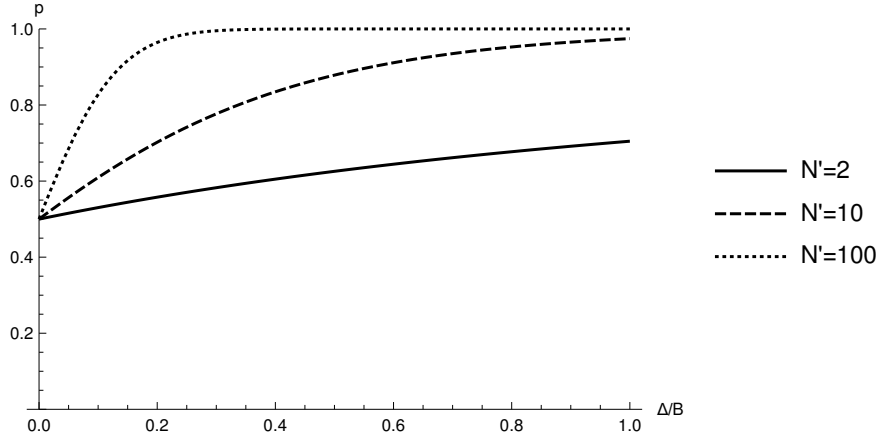

Figure 4.5: The same as Figure 4.4, shown over a greater range of the relative fitness difference $\frac{\Delta}{B}$.

We can see from these plots that, for example, if two individuals have a relative difference in evolvability of $\frac{\Delta}{B} = 0.1$, then for a poll population size of $N' = 10$, we only have a probability of 0.6 of correctly selecting the individual with the greater true evolvability, despite using a factor of ten times more fitness evaluations than in the analysis in Chapter 3. Increasing N' by another factor of ten, to 100, increases p to around 0.8.

In the limit as $\frac{\Delta}{B}$ goes to infinity, for $N' \geq 2$, p goes to 1. In words, as the difference in evolvabilities grows larger, even for small sample sizes our probability of correctly determining which individual has the greater evolvability asymptotically approaches 1. In the limit as N' goes to infinity, for any $\frac{\Delta}{B} > 0$, p goes to 1; any difference in evolvability can be detected if the sample sizes are large enough. The sample size N' required such that $p \geq p'$, for any $p' > 0.5$, goes to infinity as $\frac{\Delta}{B}$ goes to zero.

How can we interpret the value of p and its effect on the expected population mean value of evolvability due to selection? If $p = 1$, then we select the individual with the greater true evolvability with certainty. If $p = 0.5$, then we select randomly with respect to evolvability.

Prior to selection, the mean evolvability of our two individuals is $B + \frac{\Delta}{2}$. If we select correctly with certainty (i.e., $p = 1$), then the expected evolvability after selection is $B + \Delta$ and the expected increase in evolvability is $\frac{\Delta}{2}$. On the other hand, if we select randomly (i.e., $p = 0.5$), then the expected evolvability after selection is $B + \frac{\Delta}{2}$, and there is no expected increase in evolvability.

4.2 Episodic Group Selection

The previous section considered a selection strategy like that described in Chapter 3, modified so that we select for a weighted sum of fitness and *estimated* evolvability in each generation, with the estimate being calculated from a 'poll' offspring population. We saw that the offspring population must be large if the difference between evolvabilities is small in order that the probability of correctly selecting the individual with the greater true evolvability is significantly larger than 0.5.

Note that the individuals in any given generation are closely related, and are likely to have similar evolvabilities. This means that, unless the poll offspring populations are very large (using more of our fixed budget of fitness evaluations per generation), the probability of selecting the individual with the greatest true evolvability will be low, and the expected increase of evolvability due to selection will be close to that when we select randomly, i.e., zero.

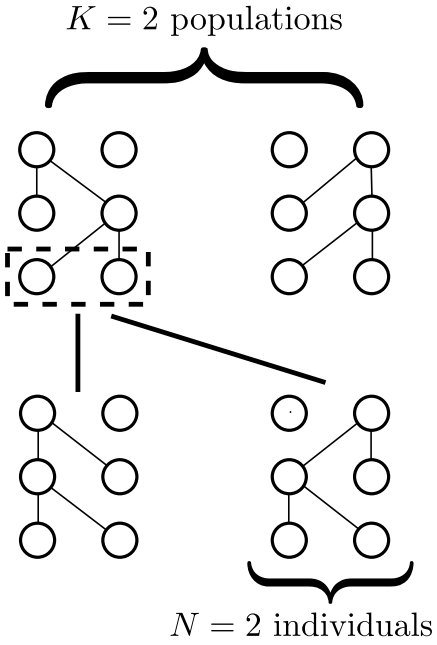

Figure 4.6: Episodic group selection for evolvability estimates. Two populations of two individuals each proceed through three generations of selection for fitness alone, followed by one step of group selection, during which the left-hand population (highlighted by the dashed square) is duplicated to replace the other population. The time at which group selection for evolvability occurs depends upon the evolvability estimation method. These methods are discussed in Section 4.4.

For that reason, the strategies described in this section do not select for evolvability in every generation. Instead, they implement *episodic group selection* (EGS) for evolvability. They maintain K separate populations, select

for fitness within each population in each generation by tournament selection with tournament size k, and periodically select between those populations for evolvability. During evolvability selection the population believed to have the greatest evolvability is duplicated to replace the other $K-1$ populations. The general strategy is illustrated in Figure 4.6.

The motivation for this kind of strategy is two-fold.

1. If the difference in evolvability between two populations is greater, it is easier, for a given number of offspring fitness samples, to determine which has the greater evolvability. If the per-generation change in evolvability due to mutations in each population is drawn independently from the same zero-mean distribution, then the magnitude of the difference in evolvability between any two populations will grow like $\sqrt{M}$, where M is the number of generations since the populations diverged.

2. In every generation, for each of the K populations we produce an offspring population of size N (i.e., the next generation). From these offspring fitnesses, we can calculate noisy estimates of the evolvabilities of the parents, and combine these into estimates of the evolvabilities of each population in that generation. We can combine the sequence of noisy evolvability estimates obtained since the populations diverged, along with a model of the changes in evolvability due to mutations, to produce more accurate evolvability estimates.

The following sections describe the components of this algorithm. Section 4.3 describes the measures of evolvability used, and Section 4.4 describes various methods for producing evolvability estimates. Section 4.5 describes some heuristics that aim to use fitness evaluations more efficiently. Sec-

tion 4.6 describes the fitness functions on which the EGS algorithm will be tested against selection for fitness alone, and Sections 4.7 and 4.8 describe the experimental design and report the results of the experiments.

4.3 Evolvability Measures

The previous chapter used a single measure of evolvability; the standard deviation of fitness of that individual's offspring. In the experiments in this chapter, I use that and one other measure, taken from the list of measures from the literature listed in Section 2.3.

I will use the following notation. The fitness of the jth offspring of individual i is a random variable denoted F_i^j. Since offspring fitnesses are assumed to be identically and independently distributed, when not referring to a particular offspring I will sometimes use the notation F_i to refer to the random variable describing the fitness of an offspring of individual i. A particular value of F_i^j, i.e., a random variate giving the fitness value of an actual, instantiated offspring of individual i is denoted f_i^j. The fitness of the parent individual i is denoted ${}^p f_i$.

In this chapter, I use two measures of evolvability, which I will refer to as ϵ_σ and ϵ_{max}^m. The ϵ_σ evolvability measure is the one used in Chapter 3. It is the standard deviation of the fitness of an individual's offspring. The true evolvability of individual i is given by

$$\epsilon_{\sigma,i} = \mathrm{SD}[F_i] \quad . \tag{4.7}$$

We can estimate the $\epsilon_{\sigma,i}$ evolvability of an individual that has n offspring by

$$\hat{\epsilon}_{\sigma,i} = \frac{1}{c_4(n)} \sqrt{\frac{1}{n-1} \sum_{j}^{n} (\bar{f}_i - f_i^j)^2} \quad , \tag{4.8}$$

where $\bar{f}$ is the mean offspring fitness of individual i, and $c_4(n) = \sqrt{\frac{2}{n-1}} \frac{\Gamma(n/2)}{\Gamma((n-1)/2)}$ is a correction term; it makes our evolvability estimate unbiased in the case that the fitnesses are drawn from a normal distribution (Holtzman, 1950).

The ϵ^m_{max} evolvability measure is the expected maximum difference between the offspring fitness and parent fitness in an offspring population of size m. The true value is

$$\epsilon^m_{max,i} = \mathbb{E}[max_{j \in 1..m}(F_i^j - {}^p f_i)] \quad . \tag{4.9}$$

We can estimate the $\epsilon^m_{max,i}$ evolvability of an individual that has m offspring by

$$\hat{\epsilon}^m_{max,i} = max_{j \in \{1,\ldots,m\}}(f_i^j - {}^p f_i) \quad . \tag{4.10}$$

If individual i has more than m offspring, then we can calculate multiple such estimates in each generation. For example, if individual i has Lm offspring, we calculate L evolvability estimates. The lth estimate is

$$\hat{\epsilon}^{m,l}_{max,i} = max_{j \in \{(l-1)m+1,\ldots,lm\}}(f_i^j - {}^p f_i) \quad . \tag{4.11}$$

The experiments in this chapter use the ϵ_σ and ϵ^2_{max} evolvability measures. Some of the evolvability estimation methods described in Section 4.4 combine a sequence of noisy evolvability estimates to produce a better estimate. For clarity, I will hereafter refer to evolvability estimates calculated from a single

offspring population, as described above, as 'evolvability observations', and I will reserve the term 'evolvability estimate' to refer to the output of the methods described in Section 4.4.

These estimation methods take as input, in each generation, one or more evolvability observations and the likelihoods of those observations, i.e., the probability of making that observation as a function of the true evolvability.

4.3.1 Likelihoods

This section gives the likelihood function of an evolvability observation conditional on the true evolvability for each of the evolvability measures ϵ_σ and ϵ^2_{max}. These likelihood functions will be used by the methods described in Section 4.4.2 to infer true evolvability values from evolvability observations. One method described in that section uses the exact likelihood function. The other uses a Gaussian approximation, which we also calculate here.

The ϵ_σ evolvability of an individual is the standard deviation of the distribution of its offspring fitnesses. We make the simplifying assumption here that the offspring fitnesses are sampled independently from a normal distribution. As we saw in Section 4.1, the sample variance S^2 of a normal distribution with standard deviation σ and with N samples follows the gamma distribution

$$S^2 \sim \Gamma\left(\frac{N-1}{2}, \frac{2\sigma^2}{N-1}\right) \quad . \tag{4.12}$$

Through a transformation of variables, we compute the likelihood function $f_S(o|\epsilon_\sigma = \sigma)$ of the uncorrected (biased) sample standard deviation S as a function of the actually observed value o, conditional on the true standard

deviation (the evolvability) being equal to σ as

$$f_S(o|\epsilon_\sigma = \sigma) = 2^{\frac{3-N}{2}} o^{N-2} \left(\frac{\sqrt{N-1}}{\sigma} \right)^{N-1} \exp\Big(-(N-1)\frac{o^2}{2\sigma^2} \Big) \quad . \quad (4.13)$$

A further transformation of variables leads us to the following as the likelihood function of the corrected (unbiased) standard deviation $\hat{S}$.

$$f_{\hat{S}}(o|\epsilon_\sigma = \sigma) = \frac{2}{o}\Gamma\big(N/2\big)^{N-1}\Gamma\left(\frac{N-1}{2}\right)^{(-N)}\Big(\frac{o}{\sigma}\Big)^{N-1} \exp\left(-\frac{\Gamma(N/2)}{\gamma\big((N-1)/2\big)}\frac{o^2}{\sigma^2} \right) \quad . \quad (4.14)$$

The mean value of this likelihood function is the true standard deviation, or true evolvability σ. The variance of the likelihood is

$$\frac{\sigma^2}{2}\left((N-1)\frac{\Gamma\big((N-1)/2\big)^2}{\Gamma\big(N/2\big)} - 2 \right) \quad , \quad (4.15)$$

and so if we wish to use a Gaussian approximation to the likelihood—which may or may not be justified—we can use the Gaussian distribution with this mean and variance.

The ϵ^2_{max} evolvability of an individual is the expected fitness of the fittest of two of that individual's offspring, minus the fitness of the individual. If we assume that offspring fitnesses are sampled independently from a normal distribution centred on the parent fitness, then the distribution of ϵ^2_{max} is the distribution of the maximum of two normally distributed random variables with equal standard deviation and zero mean. According to Nadarajah and Kotz (2008), the likelihood function of the measured maximum $\hat{\epsilon}^2_{max}$ as a function of the value o that it takes, conditional on the true standard

deviation taking the value σ is

$$f_{\hat{\epsilon}^2_{max}}(o|\epsilon_\sigma = \sigma) = 2\phi_\sigma(o)\Phi_\sigma(o) \quad , \tag{4.16}$$

where $\phi_\sigma(o)$ and $\Phi_\sigma(o)$ are the pdf and cdf of the zero mean Gaussian distribution with standard deviation σ.

A change of variables leads us to the likelihood function whose conditional is in terms of the true ϵ^2_{max} value, rather than the true ϵ_σ value. The following is the likelihood function of the observed maximum $\hat{\epsilon}^2_{max}$ in terms of the actually observed value of the maximum o, conditional on the true value of ϵ^2_{max} being m.

$$f_{\hat{\epsilon}^2_{max}}(o|\epsilon^2_{max} = m) = \frac{1}{\sqrt{2\pi}m} \exp\Big(-\frac{o^2}{2\pi m^2}\Big) \operatorname{erfc}\Big(-\frac{o}{\sqrt{2\pi}m}\Big) \quad , \tag{4.17}$$

where $\operatorname{erfc}(x)$ is the complementary error function. The mean value of the likelihood function is the true expected value of ϵ^2_{max}, m, and the variance is $(\pi - 1)m^2$. When needed, we can use as a Gaussian approximation to the likelihood the Gaussian distribution with this mean and variance.

It may seem odd to use two different evolvability measures, ϵ_σ and ϵ^2_{max} when, in computing the likelihood functions, we assume that offspring fitnesses are normally distributed. In the case that offspring fitnesses are normally distributed, the two measures are closely related. When ϵ_σ takes value σ, ϵ^2_{max} takes value $m = \sigma/\sqrt{\pi}$. The reason for the two separate measures is that the fitness functions and mutation operators described in Section 4.6 deviate from the assumption of normally distributed offspring fitness values to varying degrees. Depending on the fitness function, either the ϵ_σ or the ϵ^2_{max} measure may be a better indicator of evolutionary potential.

4.4 Evolvability Estimation Methods

In this section I describe three methods for calculating estimates of evolvability to use in the group selection step. As discussed in Section 4.2, there are two motivating factors for episodic group selection for evolvability. One is that the evolvabilities of the populations diverge and differences become easier to detect. The other is that we can combine a sequence of noisy observations to produce better evolvability estimates. Although all of the strategies I describe in the following sections take advantage of diverging evolvability values, not all attempt to combine observations.

4.4.1 Point-estimate estimation method

In this section I describe the *point-estimate* method for estimating population evolvabilities, which uses as its evolvability estimate for each population the most recent evolvability observation. For example, for evolvability measure ϵ_σ, where we are trying to estimate the expected standard deviation of the fitness of the offspring of a population, this method estimates that by the sample standard deviation of the offspring population immediately prior to evolvability selection. The point-estimate method is the only method used that does not combine the sequence of noisy evolvability observations to produce better evolvability estimates.

The number of generations between evolvability selection events in this strategy is fixed at M. Prior to evolvability selection, a poll offspring population of size N' is produced for each population, and the offspring fitnesses are used to calculate the evolvability observation. Algorithm 2 describes the point-estimate method in full, and it is illustrated by Figure 4.7. We take the mean of the evolvability observations of the parents to calculate the evolvabil-

ity estimate of the population. If we have multiple evolvability observations for a parent, as in the case where we use the ϵ^2_{max} evolvability measure and the parent has more than two offspring, then the mean of these observations is taken to be the evolvability estimate of the parent.

Algorithm 2 EGS with a point-estimate of evolvability

```
1: generation ← 1
2: spent_evaluations ← 0
3: initialize K populations of N individuals
4: while spent_evaluations < evaluation_budget do
5:     for each population do
6:         produce offspring population of size N by tournament selection, with tournament size k
7:         mutate each offspring
8:         pair off offspring, crossover each pair with probability p_c
9:         replace the parent population with the offspring population
10:    spent_evaluations ← spent_evaluations +KN
11:    generation ← generation +1
12:    if generation mod M = 0 then
13:        for each population do
14:            produce offspring population of size N'
15:            calculate evolvability observations of parents
16:        spent_evaluations ← spent_evaluations +KN'
17:        replace each population with the population with the greatest estimated evolvability
```

Because the evolvability measures require that an individual has at least two offspring, the tournament selection method used to select for fitness within each population ensures that each parent that produces offspring produces at least two offspring. This tournament selection is described by Algorithm 3.

This estimation method has two parameters; the number of generations between evolvability selection events, M, and the offspring population size used to calculate the evolvability estimate, N'. In each generation between evolvability selection events, there are KN fitness evaluations, where K is

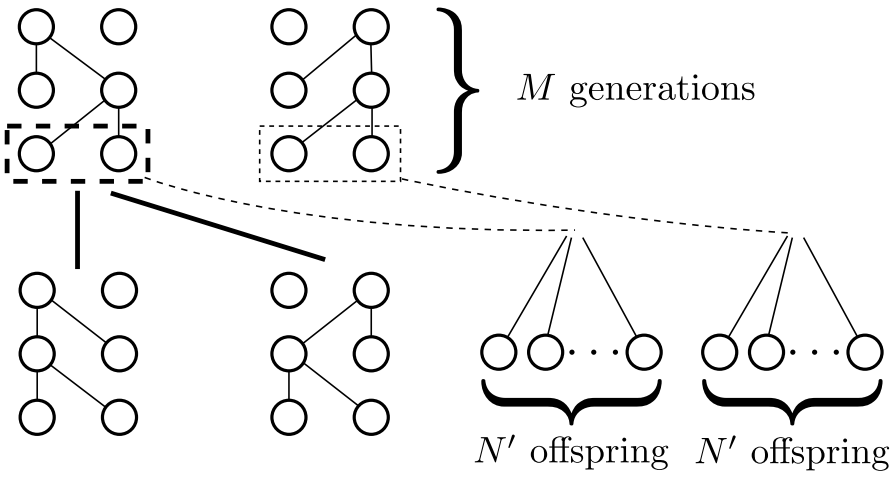

Figure 4.7: Episodic group selection for evolvability estimates using point-estimates of evolvability. Selection for fitness alone proceeds within each population for M generations, by tournament selection with tournament size k. Then, a poll offspring population of size N' is produced for each of the K populations. Evolvability estimates for each population are calculated from these offspring populations, after which the offspring populations are discarded. The population with the greatest evolvability estimate is duplicated to replace each of the other $K - 1$ populations. Each population then goes through another M generations before the next evolvability selection event.

the number of populations. In a generation in which selection for evolvability occurs, there are KN' fitness evaluations. As we saw in Section 4.1, the probability of selecting the population with the larger true evolvability value increases with M and N'. Since we have a fixed budget of fitness evaluations, increasing N' decreases the number of fitness evaluations that can be used for selecting for fitness directly and increasing M decreases the number of times that we can select for evolvability.

Because this estimation method does not combine a sequence of evolvability observations to produce a better estimate, it is not expected to perform as well as the estimation methods described in the following sections.

Algorithm 3 Tournament selection

Input: tournament size k, set of N parents
Output: set of N offspring

1: for each $j \in \{0, \ldots, N/2 - 1\}$ **do**
2: for each $i \in \{1, \ldots, k\}$ **do**
3: $index_i \leftarrow Uniform(\{1, \ldots, N\})$
4: $i^* \leftarrow \text{argmax}_{i \in \{1,\ldots,k\}} parent_{index_i}.fitness$
5: $offspring_{2j} \leftarrow parent_{index_{i^*}}$
6: $offspring_{2j+1} \leftarrow parent_{index_{i^*}}$

4.4.2 Sequential Bayesian filtering

In this section and the following sections, I describe methods for estimating evolvability that combine a sequence of noisy observations made over time to produce better estimates. These methods model the evolvability of each of the K populations and the noisy evolvability observations as a *state space model* (Murphy, 2012, p. 631).

A state space model is a *hidden Markov model* in which the hidden state and observations are continuous. There is a vector representing the hidden state at time t, $\mathbf{z}_t$, and a vector of observations $\mathbf{y}_t$. The model can be written as

$$\mathbf{z}_t = g(\mathbf{u}_t, \mathbf{z}_{t-1}, \boldsymbol{\epsilon}_t) \tag{4.18}$$

$$\mathbf{y}_t = h(\mathbf{z}_t, \mathbf{u}_t, \boldsymbol{\delta}_t), \tag{4.19}$$

where $\mathbf{u}_t$ is an optional control signal. The *transition model* function g tells us how the current state depends on the previous state and the *system noise* $\boldsymbol{\epsilon}_t$. The *observation model* function h tells us how our current observations depend on the current state and the *observation noise* $\boldsymbol{\delta}_t$.

In our application, the hidden state $\mathbf{z}_t$ is the vector of evolvabilities of the K populations. The observation vector $\mathbf{y}_t$ is the vector of evolvability

observations in generation t. I make the assumption that the evolvability of each population changes from one generation to the next by adding a value drawn from a zero-mean normal distribution with variance q^2, i.e.,

$$\mathbf{z}_t = \mathbf{z}_{t-1} + \boldsymbol{\epsilon}_t \quad , \tag{4.20}$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, q\mathbb{1}_k)$ is a multivariate Gaussian random variable. The likelihood of a particular vector $\mathbf{y}_t$ of observations given the current state $\mathbf{z}_t$, $p(\mathbf{y}_t|\mathbf{z}_t)$, depends on which evolvability measure is used. The likelihood functions in Section 4.3 can be used to calculate the likelihood of each observation, and the likelihood of the vector of observations is the product of the individual observations.

A common problem of state space models is to infer the current hidden state $\mathbf{z}_t$ from the sequence of observations made so far, $\mathbf{y}_{1:t}$. This is known as the filtering problem. For our application, that corresponds to inferring the current evolvability values of the K populations from the sequence of evolvability observations made so far.

I will operate in a Bayesian setting, starting with a prior distribution over the initial evolvabilities $p(\mathbf{z}_1)$ and in each generation calculating a posterior distribution $p(\mathbf{z}_t|\mathbf{y}_{1:t}) \propto p(\mathbf{z}_t|\mathbf{y}_{1:t-1})p(\mathbf{y}_t|\mathbf{z}_t)$ using Bayes' rule. I describe in the following sections two well-known Bayesian filtering algorithms for calculating this posterior $p(\mathbf{z}_t|\mathbf{y}_{1:t})$ for a state space model. They are the *Kalman filter* and *particle filter*. Again, in my description of these I closely follow Murphy (2012). Each of these algorithms maintains a probability distribution over the current hidden state, and updates that distribution iteratively in a two-step process.

1. In the *predict* step, the distribution is updated to account for how the

system may have changed between observations. In our case, we simply add the process noise as described in Equation (4.20), and so become less certain about the state of the system.

2. In the *update* step, the distribution is updated to account for the likelihood of the most recent observation, and we become more certain about the state of the system.

In each generation of the EGS algorithm, we calculate an evolvability observation for each parent in the previous population and use these to update the filter's probability distribution of the population evolvabilities in the update step. We make the simplifying assumption that individuals within the same population have the same evolvability, so that we can then treat the evolvability observations of the individuals in the population as multiple observations of this population evolvability. This assumption may be a little strong. However, we can see that the expected within-population variance of evolvability must be less than the between-population variance of evolvability. Immediately after we select for evolvability, the within- and between- population variances of evolvability are the same, since each population contains copies of the same individuals. Until the next evolvability selection event, evolvability values will follow a random walk. This will cause the between-population variance of evolvability to grow indefinitely. The within-population variance will also grow, but towards some finite equilibrium value. We can see that this is true by considering the following. The effect of selection for fitness is that, within each population, all individuals are descended from a single individual some finite number of generations earlier. This puts a bound on the number of evolvability mutations that can have occurred since this most-recent common ancestor, and thus on the expected variance of evolvability within that population. Once the point

is reached that the most-recent common ancestor is found within a generation that occurs after the previous evolvability selection event, the expected within-population variance of evolvability is at some equilibrium value below the between-population variance.

From the current probability distribution over evolvabilities, for each population we can calculate the probability that that population has the highest evolvability. When this probability exceeds P for one of the populations, that population is duplicated to replace the other $K-1$ populations. What happens to our probability distribution over the evolvabilities of the K populations after one population is duplicated in this way? Since the ith dimension of this distribution represents our state of belief of the evolvability of population i, if a particular population j is duplicated to replace all other populations, then the probability distribution over the evolvabilities should be modified such that, in all but the jth dimension, the marginal distribution becomes perfectly correlated with the marginal distribution in the jth dimension.

The EGS algorithm with evolvabilities estimated by a Bayesian filter is described in Algorithm 4 and shown in Figure 4.8. Note that we calculate an evolvability observation for each parent in a population, and these multiple observations are used to update the filter's estimate of the *population* evolvability. Also note that these evolvability observations are calculated prior to crossover.

In the following sections, I describe how the Kalman filter and particle filter encode the probability distribution over the current state, how that distribution is updated during the predict and update steps, and how the distribution is updated when one population is duplicated replacing all others.

Algorithm 4 EGS with evolvabilities estimated by a Bayesian filter

```
spent_evaluations ← 0
initialize K populations of N individuals each
while spent_evaluations < evaluation_budget do
    update filter to account for evolvability changes between observations
    for each population do
        produce offspring population of size N by tournament selection, with tournament size k
        mutate each offspring
        calculate evolvability observations for the parents
        pair off offspring, crossover each pair with probability p_c
        replace the parent population with the offspring population
    update filter using evolvability estimates
    spent_evaluations ← spent_evaluations +KN
    if evolvability of any one population exceeds all others with probability exceeding P then
        replace each population with the population with the greatest estimated evolvability
        update filter to account for group selection
```

4.4.3 Kalman filter estimation method

We can derive the posterior probability distribution of a linear Gaussian state space model exactly (Murphy, 2012, p. 640). A LG-SSM is one such that

$$\mathbf{z}_t = \mathbf{F}\mathbf{z}_{t-1} + \boldsymbol{\epsilon}_t \tag{4.21}$$

$$\mathbf{y}_t = \mathbf{H}\mathbf{z}_t + \boldsymbol{\delta}_t \quad , \tag{4.22}$$

where $\mathbf{z}_t$ is a column vector of size K, $\mathbf{F}$ is a $K \times K$ matrix, $\mathbf{y}_t$ is a column vector of size N, $\mathbf{H}$ is a $N \times M$ matrix, and the process noise $\boldsymbol{\epsilon}_t$ and observational noise $\boldsymbol{\delta}_t$ are zero-mean multivariate Gaussians with covariance matrices $\mathbf{Q}$ and $\mathbf{R}$.

In this case, if our prior distribution over the initial evolvabilities $p(\mathbf{z}_1)$ is Gaussian, then all subsequent distributions over $\mathbf{z}_t$ conditional on the ob-

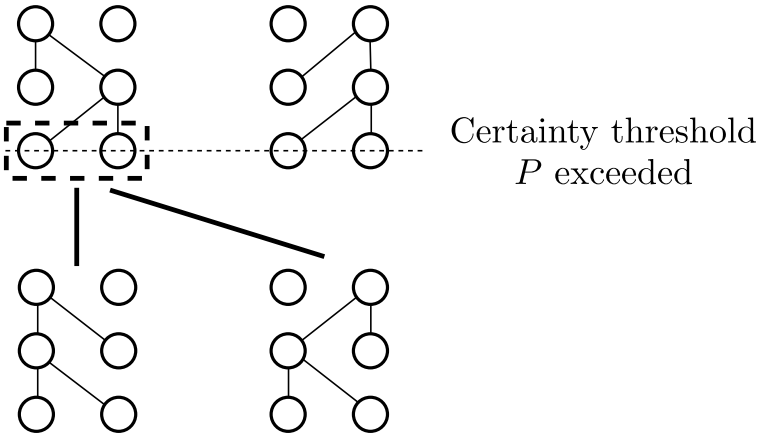

Figure 4.8: EGS with evolvabilities estimated using sequential Bayesian filtering. The N fitness values of each population in each generation are used to calculate evolvability observations (i.e., very noisy estimates of evolvability), which the filter uses along with a model of how the evolvabilities change in each generation due to mutation to improve its current estimates. Once the filter believes with greater than probability P that one particular population has the greatest evolvability, that population is duplicated to replace the $K - 1$ other populations.

servations $\mathbf{y}_{1:t}$, $p(\mathbf{z}_t|\mathbf{y}_{1:t})$, are Gaussian and can be derived using the Kalman filter algorithm. The Kalman filter—named after Rudolf E. Kálmán, one of the main developers of the algorithm (Kálmán, 1960)—maintains a vector $\mathbf{x}$ and matrix $\mathbf{P}$ giving the mean vector and covariance matrix of the multivariate Gaussian representing our current belief of the state of $\mathbf{z}_t$. The predict step, which accounts for how the system may have changed between observations, is shown in Algorithm 5, and the update step, which incorporates information from new observations, is shown in Algorithm 6.

Algorithm 5 Kalman filter predict step

1: $\mathbf{x} \leftarrow \mathbf{F}\mathbf{x} + \mathbf{B}\mathbf{u}_t$
2: $\mathbf{P} \leftarrow \mathbf{F}\mathbf{P}\mathbf{F}^T + \mathbf{Q}$

In our application, the likelihood of the (unbiased) evolvability observa-

Algorithm 6 Kalman filter update step

1: $\tilde{\mathbf{y}} \leftarrow \mathbf{y}_t - \mathbf{H}\mathbf{x}$
2: $\mathbf{S} \leftarrow \mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R}$
3: $\mathbf{K} \leftarrow \mathbf{P}\mathbf{H}^T\mathbf{S}^{-1}$
4: $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{K}\tilde{\mathbf{y}}$
5: $\mathbf{P} \leftarrow (\mathbb{1}_k - \mathbf{K}\mathbf{H})\mathbf{P}$

tion of population i depends only on the true evolvability of population i, so the matrix $\mathbf{R}$ is diagonal, with the variance of the relevant likelihood function from Section 4.3 filling the diagonal entries. The result is that we update our state of belief of the population evolvabilities using Gaussian approximations to the likelihoods of our observations.

Since in our case $\mathbf{F} = \mathbb{1}_k$, $\mathbf{H} = \mathbb{1}_k$, there is no control signal $\mathbf{u}_t$, and we assume that $\mathbf{Q} = q\mathbb{1}_k$, these steps simplify to those shown in Algorithms 7 and 8. The predict and update steps of the Kalman filter are illustrated by Figure 4.9.

Algorithm 7 Simplified Kalman filter predict step

1: $\mathbf{P} \leftarrow \mathbf{P} + \mathbf{Q}$

Algorithm 8 Simplified Kalman filter update step

1: $\tilde{\mathbf{y}} \leftarrow \mathbf{y}_t - \mathbf{x}$
2: $\mathbf{S} \leftarrow \mathbf{P} + \mathbf{R}$
3: $\mathbf{K} \leftarrow \mathbf{P}\mathbf{S}^{-1}$
4: $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{K}\tilde{y}$
5: $\mathbf{P} \leftarrow (\mathbb{1}_k - \mathbf{K})\mathbf{P}$

Since we select for evolvability when we believe with probability greater than or equal to P that one population has greater evolvability than all others, we need to be able to extract this probability from the multivariate Gaussian. For a multivariate Gaussian of dimensionality greater than two, there is no closed form expression for this probability. Instead we take $m =$

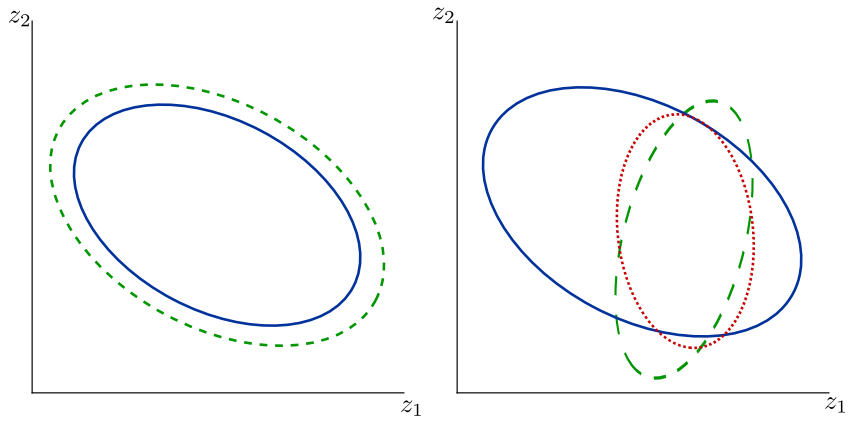

Figure 4.9: The ellipses represent the one standard deviation contour line of a two-dimensional Gaussian distribution. The left-hand side shows the distribution before (solid line) and after (dashed) the predict step of a Kalman filter. Due to the random element of the process model, we are less certain about the state of the system after the predict step. On the right-hand side, the solid line represents our state of belief prior to making an observation. The dashed line represents the likelihood function. The dotted line represents our posterior state of belief, incorporating the latest observation.

1000 samples from the multivariate Gaussian and estimate the probability that population i has the greatest evolvability by

$$p_i = \frac{1}{m} \sum_{j=1}^{m} \mathbb{1}(\underset{k \in \{1..K\}}{\operatorname{argmax}} x_j^k = i) \quad , \tag{4.23}$$

where x_j^k is the kth element of the jth sample from the multivariate Gaussian, and $\mathbb{1}$ is the indicator function, equal to one if its argument is true, zero otherwise. This estimate is the proportion of samples in which population i has the greatest evolvability.

If this probability is greater than or equal to P for the ith population, then the ith population is copied to replace each of the other populations. The Kalman filter is updated after such a duplication as follows. Each element in the mean vector $\mathbf{x}$ is replaced with the value x_i, and each element in the

covariance matrix $\mathbf{P}$ is replaced with the value $P_{i,i}$.

How should we infer the value q, the variance of the process model noise? We could add an extra dimension to our system and infer the value of q, but we would no longer be dealing with a linear-Gaussian state space model, and so couldn't perform exact inference with the Kalman filter. Instead, I use in each step the current maximum-likelihood estimate of q, which is given by

$$\hat{q} = \frac{1}{(g-1)k} \sum_{i=2}^{g} \sum_{k=1}^{k} (x_i^k - x_{i-1}^k)^2 \quad , \tag{4.24}$$

where x_i^k is the kth element of the mean vector of the Kalman filter in generation i, and g is the total number of generations. The maximum likelihood value of q is the mean squared difference between our expected value of the evolvability of a population from one generation to the next.

4.4.4 Particle filter estimation method

The previous section describes the use of the Kalman filter algorithm to predict the current evolvability of each population from the sequence of evolvability observations so far. In order to use the Kalman filter, we have to use Gaussian approximations to the likelihoods of the observations and use a maximum-likelihood value of the process noise variance q. This section describes the use of a particle filter algorithm to track the population evolvabilities. A particle filter is a Monte Carlo (sampling) method for calculating the probability distribution over the current state of a state space model conditional on all observations so far, without the following restrictions, which apply to the Kalman filter (Murphy, 2012, p. 823).

- The process model function, which relates the current state to the previous state, need not be linear.

- The observation model function, which relates the current observation to the current state, need not be linear.

- The process noise and observation noise need not be Gaussian.

- The prior distribution over the initial state $p(\mathbf{z}_1)$ need not be Gaussian.

This means we can use the exact likelihood functions as given in Section 4.3, and to infer the process noise variance q we can add an extra dimension to our system rather than using the maximum-likelihood value as in the previous section.

A particle filter estimates the posterior distribution with a large number of weighted samples. We start with some prior distribution over the initial state of the system $p(\mathbf{z}_1)$, and we take $N_{samples}$ samples, or particles, from this distribution. We associate with each particle a weight, w_i for the ith particle, and initialize each weight to $\frac{1}{N_{samples}}$. In our application, the system is $K+1$ dimensional, with the first K elements of $\mathbf{z}_t$ giving the evolvabilities of the K populations and the $(K+1)$th element giving the process noise variance q. The $K+1$ dimensional vector $\mathbf{x}_i$ represents the position of the ith particle.

As with the Kalman filter, we alternate between predict and update steps. During the predict step, the cloud of particles moves to account for how the system may have changed since our last observation. The position of each particle represents a possible state of the system, and its position after the predict step depends on the dynamics of the system, the initial position, and the process noise. During the update step, the likelihoods of our latest evolvability observations are taken into account to adjust the weights associated with the particles.

In our application, during the predict step each particle moves in each of the first K dimensions by adding noise from a K dimensional multivariate Gaussian. The ith particle moves in the first K dimensions by adding noise from a multivariate Gaussian with covariance matrix $x_i^{K+1}\mathbb{1}_K$. We also add noise to the last element, which represents a possible value of the process noise variance q, drawn from a Gaussian distribution with small variance, for reasons which are explained below.

If the evolvability measure in use is ϵ_σ, after the predict step we prevent any particle from taking a negative value in the first K dimensions, and we always prevent particles from taking negative values in the $(K+1)$th dimension. This is because these variables represent possible standard deviation parameters. They are prevented from taking negative values by setting any negative values to the small positive value 0.001. The weights are left unaffected during the predict step. This predict step is described by Algorithm 9 and is illustrated in Figure 4.10.

Algorithm 9 Particle filter predict step

1: for each particle i **do**
2: for each $j \in \{1..K\}$ **do**
3: $\mathbf{x}_i^j \leftarrow \mathbf{x}_i^j + X, X \sim \mathcal{N}(0, \mathbf{x}_i^{K+1})$
4: $\mathbf{x}_i^{K+1} \leftarrow \mathbf{x}_i^{K+1} + Q, Q \sim \mathcal{N}(0, 0.01)$

During the update step, given our evolvability observations for the current generation $\mathbf{y}_t$, we calculate for each particle the likelihood of having made that observation if the system were in the state that the particle represents. To do so, we calculate

$$p(\mathbf{y}_t|\mathbf{x}_i) = \prod_{j=1}^{K} p(y_t^j|x_i^j) \quad , \tag{4.25}$$

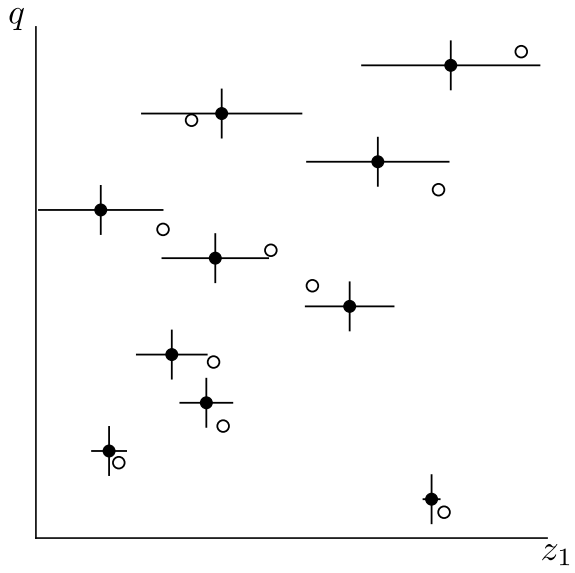

Figure 4.10: Each solid point represents a possible value for the evolvability of a population, z_1, and the process noise variance q. Each unfilled point represents the position of the same particle after the predict step. These values are obtained by adding Gaussian noise with fixed, small standard along the q axis (illustrated by the fixed length vertical lines), and Gaussian noise with standard deviation $\sqrt{q}$ on the z_1 axis (illustrated by the variable length horizontal lines, which are longer towards the top of the figure).

which is simply the product of the likelihood functions $p(y_t^j|x_i^j)$ of the observations for each population as given in Section 4.3. We multiply each weight w_i by the corresponding likelihood, and then normalize such that the sum of the weights is equal to one, as shown in Algorithm 10. Although in practice it is better to work with log-likelihoods to avoid problems of overflow and underflow, in the presentation here we work with likelihoods for clarity.

Algorithm 10 Particle filter update step

1: for each particle i **do**
2: $\quad w_i \leftarrow w_i p(\mathbf{y}_t|\mathbf{x}_i)$
3: $\mathbf{w} \leftarrow \mathbf{w} / \sum_{i=1}^{N_{samples}} w_i$

After the update step, the particles are still samples from the distribution representing our state of belief *prior* to making the observations. However, the particles and the weights together represent a weighted sample from the *posterior* distribution $p(\mathbf{z}_t|\mathbf{y}_{1:t})$. We can calculate an estimate of any statistic of the posterior using the weighted samples. For example, our estimate of the expected value of the evolvability of the jth population is given by

$$\hat{x}^j = \sum_{i=1}^{N_{samples}} w_i x_i^j \quad . \tag{4.26}$$

Note that inference on the process noise variance q is indirect, since we never observe q directly. After several predict-update iterations, it will be the case that the majority of particles have weights close to zero, and that we have very few particles in the regions of state space with non-negligible probability density, leading to large sampling errors. The solution is to *re-sample* such that we obtain unweighted samples from the posterior. To do so, we repeatedly sample each particle proportionally to its weight. There are several ways that this resampling can be done. As part of the EGS algorithm it is done by *systematic resampling*. During systematic resampling, first we calculate the cumulative sums of the weights w_i,

$$c_i = \sum_{j=1}^{i} w_j \quad . \tag{4.27}$$

We then sample $N_{samples}$ from $[0, 1]$, but do so in such a way that the samples are uniformly spaced. This is done by drawing the first sample s_1 uniformly from $[0, 1/N_{samples}]$, and then setting the $s_{i+1} = s_i + 1/N_{samples}$. The new ith particle is a duplicate of the old jth particle if and only if $c_i \geq s_i > c_{i-1}$, where $c_0 = 0$. This process is illustrated by Figure 4.11 and

described by Algorithm 11[2].

Algorithm 11 Particle filter systematic resampling

$r \sim Uniform([0, 1])$
for each i **do**
    $p_i \leftarrow 1/N_{samples} + r$
    $c_i \leftarrow \sum_{j=1}^{i} w_j$
$i, j = 1, 1$
while $I \leq N_{samples}$ **do**
    if $p_i < c_j$ **then**
        ith new particle is duplicate of jth old particle
        $i \leftarrow i + 1$
    else
        $j \leftarrow j + 1$

Figure 4.11: Systematic particle resampling in the case that we initially have six particles with the first weight $w_1 = 0.5$, and the other weights equal to 0.1. After sampling six times from $[0, 1]$ such that our samples are uniformly spaced, and creating duplicates of the particles whose cumulative weights corresponds to the samples, we have three duplicates of the first particle, one of the second, one of the fourth, and one of the fifth, with each weight set to $\frac{1}{6}$. The third and sixth particles were not resampled, and have been discarded.

We could perform resampling after every update step. However, to reduce the computational cost and reduce the sampling error, a common heuristic is to calculate the so-called effective sample size $s_{eff} = \sum_i \frac{1}{w_i^2}$. We resample only if $s_{eff} < N_{samples}/2$.

The reason that noise is added to the q dimension during the predict step, even though the true value is stationary, is that otherwise, after many update-resample iterations, every particle would have the same q value, whichever gave the highest likelihood of the sequence of observations; we could never get any closer to the true value of q.

[2]This algorithm is taken from a tutorial provided by Labbe (2016)

The whole process of sampling particles from a prior, updating the weights of those particles according to the likelihood of an observation, and then resampling such that the particles are unweighted samples from the posterior, is illustrated by Figure 4.12.

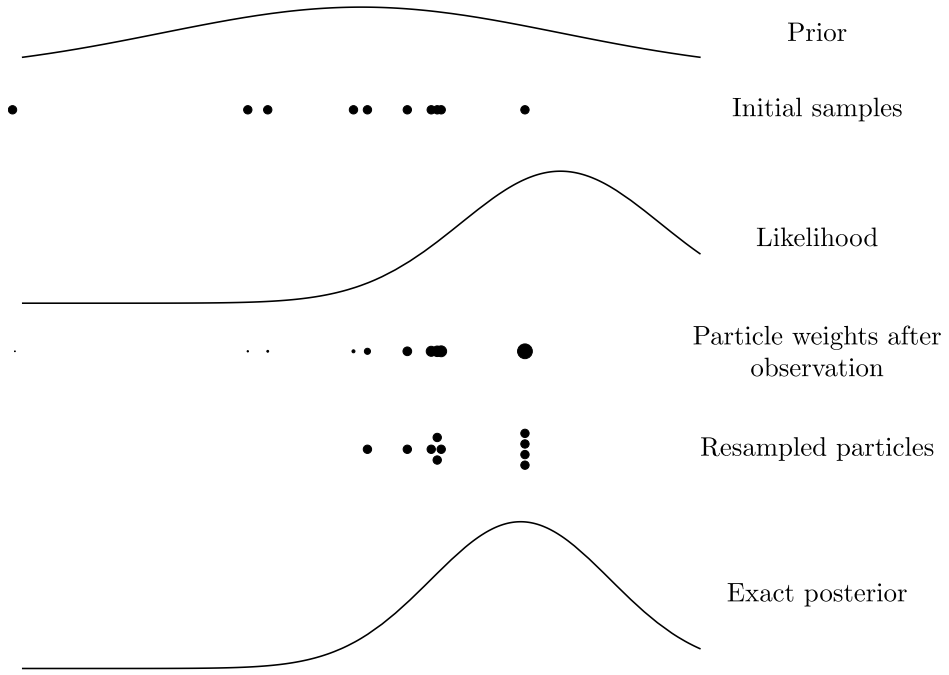

Figure 4.12: The whole process of sampling, updating weights, and resampling. First we sample particles from our prior distribution. Then we update the weights (illustrated by the particle sizes) according to the likelihood of an observation. Finally we resample such that the particles are unweighted samples from the posterior. For clarity, the resampled particles are shown stacked vertically where they are in the same location. The exact posterior, which the particle filter algorithm does not calculate, is shown below the resampled particles for comparison.

Our evolvability selection method requires that we are able to extract from our posterior distribution the probability p_i that the ith population has

greater evolvability than any other population. We calculate this by

$$p_i = \sum_{j=1}^{N_{samples}} w_j \mathbb{1}(\underset{k \in \{1..K\}}{\operatorname{argmax}} x_j^k = i) \quad . \tag{4.28}$$

If this probability p_i is greater than or equal to the certainty threshold parameter P, then we replace each population with a copy of population i. Each particle is updated after this duplication such that $\mathbf{x}_j^k \leftarrow \mathbf{x}_j^i$ for all $k \in \{1..K\}$, for all j.

4.5 Termination Heuristics

In this section I describe optional heuristics for improving the efficiency of the EGS algorithm. Recall that in the EGS algorithm, K populations are maintained. In each generation, KN fitness evaluations are spent, where N is the population size within each population. After M generations, MNK fitness evaluations have been expended in total. Within the lineage that has been successful (i.e., that has survived each evolvability selection step) only MN fitness evaluations have been expended—the same number expended during M generations when selecting for fitness alone. The remaining $MN(K-1)$ fitness evaluations are used to measure and select for evolvability, and do not contribute to increasing fitness directly. For group selection for evolvability to be effective, the eventual fitness gain due to increased evolvability must more than compensate for the fact that the algorithm has a factor of K fewer fitness evaluations to use in directly selecting for fitness.

Note that differences in evolvability lead to differences in fitness only after a delay, and that increases in evolvability in early generations have more of an impact on eventual fitness than later increases in evolvability, if the increases

in evolvability persist. These observations motivate the following *termination heuristics*, according to which—once a certain condition is satisfied—we stop selecting for evolvability and maintain only one population thereafter.

- According to *termination heuristic 1*, selection for evolvability stops when half of the total budget of fitness evaluations has been expended.

- According to *termination heuristic 2*, selection for evolvability stops when the number of fitness evaluations since we last selected for evolvability is more than one tenth of the remaining fitness evaluation budget.

4.6 Fitness Functions

In the experiments reported in this chapter, episodic group selection for evolvability is compared to selection for fitness alone on four fitness functions. This section describes those fitness functions. The fitness functions have been designed such that experiments can go on indefinitely; they are either open ended, such that fitness and evolvability can increase without limit, or they vary over time such that the fitnesses and evolvabilities of individuals (in the absence of selection) decrease over time. This is so that we can compare the final average population fitness for each algorithm. Each of the time-varying fitness functions depends upon a *fitness function rate of change* parameter δ, which determines the rate at which the fitness function changes. Fitness function changes are triggered every N fitness evaluations, where N is the number of individuals within each population.

4.6.1 Fitness function 1: simple evolvability model

The *simple evolvability model* (SEM) fitness function is simply the model described in Chapter 3, with the decay parameters $k_A = 1$, $k_B = 1$; neither fitness nor evolvability decay over time.

To recap, in this model each individual consists of two numbers, A and B. The fitness of an individual is its A value. During mutation, we add Gaussian noise to the A value with standard deviation B, and add Gaussian noise to the B value with constant standard deviation β.

If the evolvability type is ϵ_σ, then the B value of an individual corresponds exactly to its evolvability. If the evolvability type is ϵ^2_{max}, then evolvability is proportional to the B value. In either case, in these experiments the B value is not used during selection; evolvability must be estimated from the offspring fitness distributions.

This is the only one of the four fitness functions that does not change over time; because the fitness function is open ended in the sense that fitness and evolvability can both increase without limit, there is no need for the fitness function to change over time in order that there still be fitness differences between the algorithms at the end of the experiment.

4.6.2 Fitness function 2: mask matching

The *mask matching* (MM) fitness function is modified from a paper by Turney (1999). In that paper, this fitness function is used in order to demonstrate indirect selection for evolvability in an environment that changes over time.

Each individual consists of two strings of 100 bits, a *primary bit string* and a *mutation mask*. The primary bit string is compared to a *target bit string*, and the fitness of an individual is the number of bits that match the

target. This is illustrated in Figure 4.13.

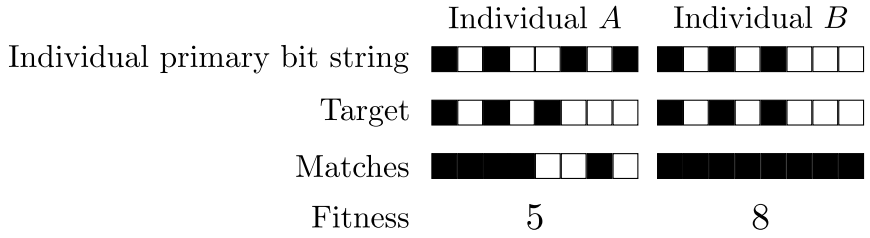

Figure 4.13: In the mask matching problem, the fitness of an individual is the number of bits in its primary bit string that match the target. A black square indicates a one, a white square a zero.

The second bit string of an individual is the mutation mask. It indicates which bits in the primary bit string are able to mutate. During mutation, each bit of the primary bit string flips with probability 0.01 unless the corresponding bit in the mutation mask is set to zero. Each bit in the mutation mask flips during mutation with probability 0.005. This is illustrated in Figure 4.14.

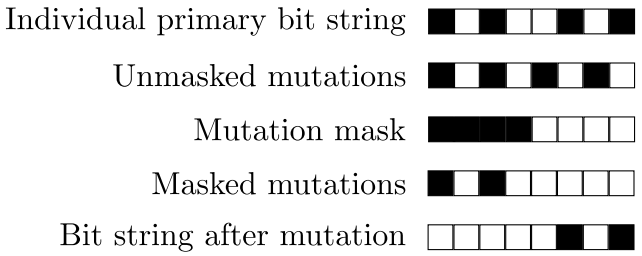

Figure 4.14: The mutation mask determines which bits of the primary bit string are able to flip during mutation. A bit may only flip if the corresponding bit in the mutation mask is set to one.

During a fitness function change event, the target changes by a small amount. In the paper by Turney (1999) each bit of the target could change, so that the most evolvable individuals, i.e., those where the way in which their descendants will vary corresponds to the ways in which the fitness function might vary, would have a mutation mask in which every bit is set to one.

Here we modify the fitness function so that there is also a *target mutation mask*, which varies over time. This modifies the way in which the target changes over time in the same way that the mutation mask of an individual modifies the mutations that it can undergo.

Each bit of the target flips with probability 0.01δ unless the corresponding bit of the target mutation mask is set to zero, where δ is the fitness function rate of change parameter. Each bit of the target mutation mask flips with probability 0.002δ.

Although its not clear exactly how the mutation mask and the target mutation mask will jointly determine the evolvability of an individual, we can see how mismatches between the two can decrease the evolvability of an individual whose primary bit string is close to the target; if the mutation mask equals zero where the target mutation mask equals one, then the target can change in ways that the individual cannot follow. On the other hand, if the mutation mask equals one in many locations where the target mutation mask equals zero, then the probability of successfully tracking changes in the target decreases due to a high probability of mutations that cannot move the bit string closer to the target.

That the target mutation mask changes over time means that, in the absence of mutation and selection, evolvability will decay.

4.6.3 Fitness function 3: symmetry matching

The *symmetry matching* (SM) fitness function is modified from a paper by Reisinger and Miikkulainen (2006). In that paper, like the fitness function described in the previous section, this fitness function is used in order to demonstrate indirect selection for evolvability in an environment that changes over time.

Each individual consists of a string of 100 bits and a single real-valued number. There is a target bit string, and the fitness of an individual is the number of bits that match the target, as in the *mask matching* fitness function.

The real-valued parameter, the *symmetry probability* p_{sym}, gives the probability that mutations affecting the bit string will be symmetric. During mutation, with probability $1 - p_{sym}$, mutations will be asymmetric; each bit of the bit string will flip with probability $p = 0.01$. Otherwise, with probability p_{sym}, mutations will be symmetric; each symmetric pair of bits will flip with probability 0.01. This is illustrated in Figure 4.15. Then, the symmetry probability parameter is mutated by adding a value drawn uniformly from $[-0.05, 0.05]$, and then capping the value of the parameter at 0 or 1 if it falls outside of the range $[0, 1]$.

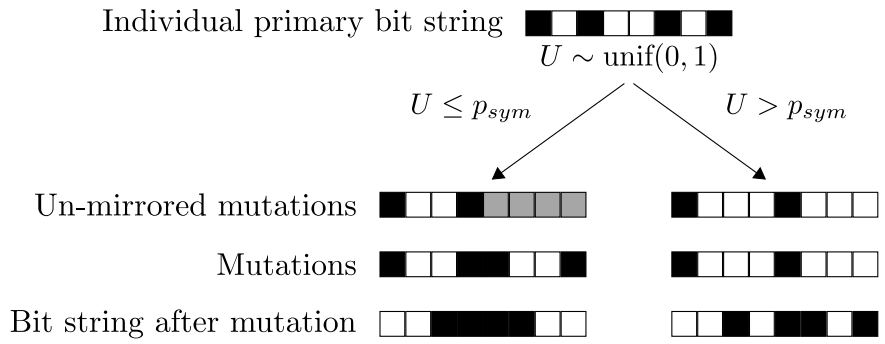

Figure 4.15: The symmetry probability parameter p_{sym} determines the probability that mutations affecting the primary bit string occur symmetrically. Each bit or symmetric pair of bits will flip with low probability.

During a fitness function change event, the target changes by a small amount. In the paper by Reisinger and Miikkulainen (2006) the changes to the target are always symmetrical, so that the most evolvable individuals would be those whose symmetry probability parameter is set to 1 (assuming the bit string of the individual is already symmetrical). Here, we modify the

fitness function so that there is also a *target symmetry probability* parameter p_{sym}^{target}, which varies over time. This modifies the way in which the target changes in the same way that the symmetry probability parameter of an individual modifies the mutations that it can undergo.

Each bit or pair of bits of the target flips with probability $p\delta = 0.01\delta$, where δ is the fitness function rate of change parameter. The parameter p_{sym}^{target} determines whether these mutations are symmetrical in the same way as illustrated in Figure 4.15. We change the target symmetry probability parameter by adding a value drawn from $[-0.05\delta, 0.05\delta]$ and capping the values in the range $[0, 1]$.

What value of p_{sym} should we expect to maximize the evolvability of an individual? Consider an individual whose bit string matches the target. With probability $1 - p_{sym}^{target}$, the changes to the target are asymmetrical, and each of the N bits flip with probability $p\delta$. We'll restrict ourselves to the case that at most one bit flips, since this is the most likely case for very small $p\delta$. An individual which doesn't undergo symmetrical mutations will have a bit string which matches the target if only that same bit flips, which occurs with probability $p(1-p)^{N-1}$. An individual which undergoes symmetrical mutation cannot have its bit string match the target in this case. If instead the changes to the target are symmetrical (which occurs with probability p_{sym}^{target}), and we restrict ourselves to the most probable case that a single pair of bits flip, then if the individual mutates asymmetrically, exactly those two bits must flip in order that the bit string matches the new target, which occurs with probability $p^2(1-p)^{N-2}$. On the other hand, if the individual mutates symmetrically, then this occurs with probability $p(1-p)^{N/2-1}$, which is much more probable. We see that if the target changes symmetrically, individuals which mutate symmetrically are much more likely

to reach the target as a result of mutations, whereas if the target changes asymmetrically, only individuals which mutate asymmetrically can reach the target as a result of mutations. We can calculate from the above that the value of p_{sym} that maximizes an individual's probability of tracking the target in a single generation is 1 if p_{sym}^{target} exceeds a certain threshold (depending on p and N) and 0 otherwise. That threshold value is in Figure 4.16 plotted against N for the case where each bit flips with probability $p = \frac{1}{N}$. We can see that for $N = 100$, the optimal value of p_{sym} is 1 if $p_{sym}^{target} > 0.38$, and 0 otherwise. Of course, this is a short-sighted analysis; an individual with $p_{sym} = 1$, i.e., that can only mutate symmetrically, has increasing probability over generations of encountering a change in the target that it can not recover from if $p_{sym}^{target} < 1$.

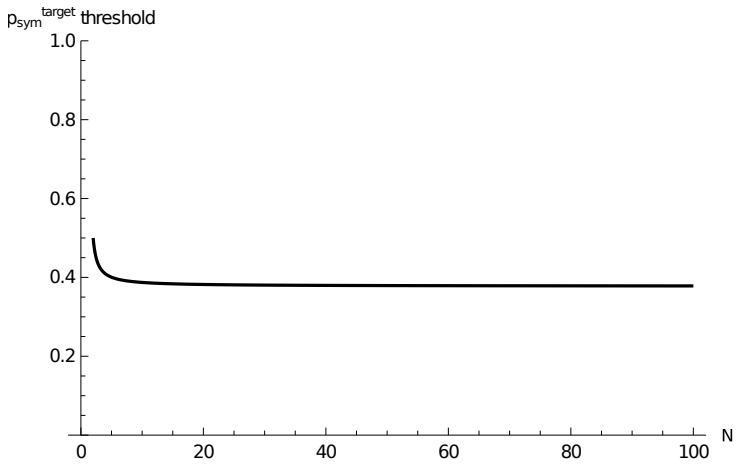

Figure 4.16: For bit strings of length N with a per bit mutation probability of $p = \frac{1}{N}$, an individual maximizes its probability of tracking a single change in the target if $p_{sym} = 1$ if p_{sym}^{target} is above the threshold value shown, 0 otherwise.

That the target parameter changes over time means that, as with the previous fitness function, evolvability will decay over time.

4.6.4 Fitness function 4: modularly-varying pattern recognition

The *modularly-varying pattern recognition* (MVPR) fitness function is taken from Clune et al. (2013). In their paper, this fitness function is used to show that a combination of a modularly-changing environment and the penalization of highly-connected neural networks leads to selection for evolvability.

Each individual consists of a neural network with eight input neurons, two hidden layers with four and two neurons respectively, and one output neuron. Each neuron uses a tanh activation function. The output from layer l of the network can be represented by the vector

$$\mathbf{a}_l = \tanh(20(\mathbf{W}^T\mathbf{a}_{l-1} + \mathbf{b})) \quad , \tag{4.29}$$

where $\mathbf{W}$ is the matrix of weights and $\mathbf{b}$ the vector of biases. The output of the final layer is taken through a step function, such that the output is 1 if the input is positive, 0 otherwise.

The network weights take values from $\{-2, -1, 0, 1, 2\}$, initialized randomly. In each generation, each network is mutated as follows. With probability 0.2, a non-zero weight is set to zero. With probability 0.2, a zero weight is set to a non-zero value in $\{-2, -1, 1, 2\}$. Each non-zero weight is modified with probability 0.01 by incrementing or decrementing with equal probability, skipping zero.

The neural networks are to perform a pattern recognition task which switches between two related tasks over time in a modular way. There are eight *left objects* and eight *right objects*, as shown in Figure 4.17. Each of these objects represents a string of four bits. Whichever task the network is performing, it must determine whether its left-most four inputs contain a *left*

object, and it must determine whether its right-most four inputs contain a *right object*. The two related tasks are as follows; during the *L-AND-R* task, the network must determine whether its inputs contains both a left object *and* a right object. During the *L-OR-R* task, the network must determine whether its inputs contains either a left object *or* a right object. These related fitness functions are illustrated in Figure 4.18, and Figure 4.19 illustrates a network giving a correct output for the *L-OR-R* task, but an incorrect output for the *L-AND-R* task.

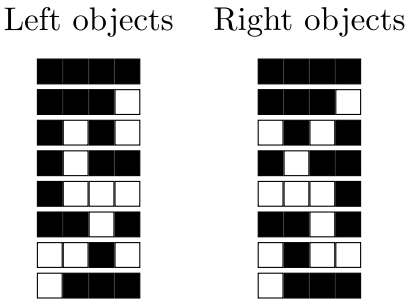

Figure 4.17: The eight left objects and eight right objects of the modularly-varying pattern recognition problem. Each represents a pattern of four bits.

The fitness of an individual is, for each of the possible 2^8 binary input patterns, the number of times the network gives the correct output for the current task.

During a fitness function change event, the task switches with probability

$$\frac{1}{1+\exp(-\delta)} - \frac{1}{2} \quad , \tag{4.30}$$

where δ is the problem rate of change parameter. This function is shown in Figure 4.20. The sigmoid function ensures that this probability is monotonically increasing with δ but cannot exceed 0.5. The probability is in the range $[0, 0.5)$. This fitness function differs from the previous two in that the

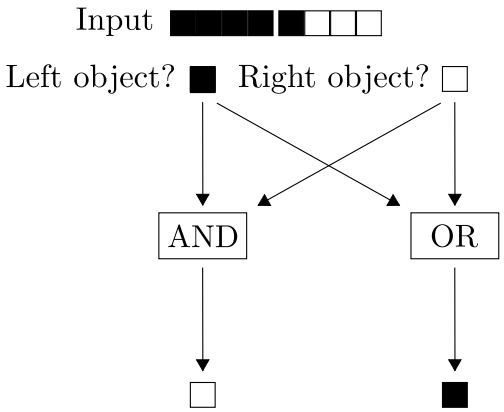

Figure 4.18: For the L-AND-R task, the network must determine whether the left-most four bits contain a left objects *and* the right-most four bits contain a right object. For the L-OR-R task, the network must determine whether its input contains a left object *or* a right object. For the specified input, the correct answer is 'no' for the former and 'yes' for the latter.

fitness function changes drastically periodically, rather than gradually.

Evolvable networks for this problem are those that are modular, such that one part of the network determines whether there is a left object, another part of the network determines whether there is a right object, and a small part of the network (in the later layers) performs the logical-AND or logical-OR operation. Such networks can adapt quickly when the task changes, as they only need to change the small part of the network dealing with the logical operation. Networks where the various functions are spread across the network cannot adapt so quickly when the task changes.

Clune et al. (2013) found that penalizing networks for their total connection length, i.e., proportionally to the number of connections with non-zero weights, leads to modular and therefore evolvable networks.

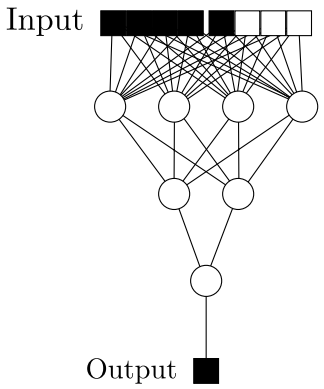

Figure 4.19: A network and its output for the specified input. The network has given the correct output for the L-OR-R task.

4.6.5 Crossover

Crossover is defined on the simple evolvability model (SEM), the mask matching problem (MM), and the symmetry matching problem (SM). For each of these problems, an individual has two traits. Depending on the problem, the traits may be two real numbers, two binary strings, or one of each. Crossover is undefined on the modularly-varying pattern recognition (MVPR) problem.

Crossover takes two individuals as input and produces two individuals as output. With probability 0.5 the first trait is crossed, otherwise the second trait is crossed. The non-crossed trait is transmitted unchanged. Real-valued traits are crossed using *simulated binary crossover*, a popular method introduced by Deb and Agrawal (1994). The probability distribution $f(\beta)$ of the parameter β from Algorithm 13 for simulated binary crossover is shown in Figure 4.21. One property of simulated binary crossover is that the mean value of the trait for the two individuals is the same before and after crossover.

Binary strings are crossed by choosing a locus in the string at random, with the string of one individual being composed of the section of the first original string before the crossover point concatenated with the section of the

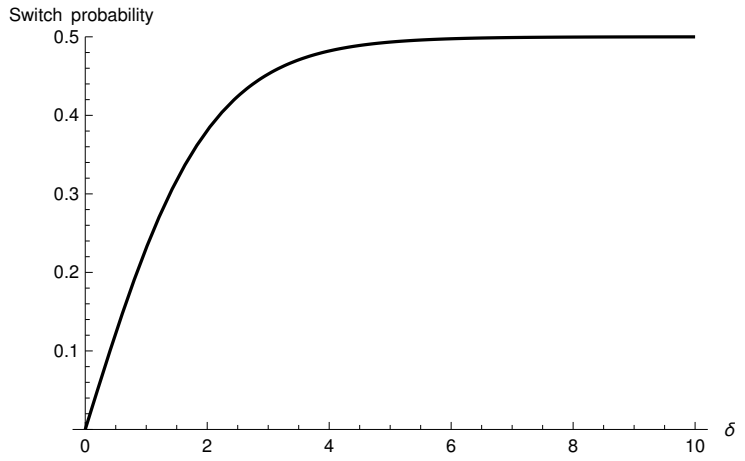

Figure 4.20: The probability that the pattern recognition task switches from L-AND-R to L-OR-R or vice versa during a fitness function change event, as a function of the problem rate of change parameter δ. The probability increases with δ and lies in the range $[0, 0.5)$.

second original string after the crossover point, and the string of the other individual being composed of the reverse.

Algorithm 12 describes the crossover procedure. Algorithm 13 describes the simulated binary crossover procedure for real-valued traits. Note that in the SEM problem, the second real-valued trait represents a standard deviation parameter, and so it must be positive, and in the SM problem, the second trait represents a probability, so it must be in the range $[0, 1]$. After performing simulated binary crossover, any trait value falling below the allowed minimum value is set to that minimum value, and any value falling above the allowed maximum value is set to that maximum value. Algorithm 14 describes the crossover procedure for binary strings.

Algorithm 12 Crossover

Input: two parents p^1, p^2, each consisting of a pair of traits
Output: two offspring o^1, o^2, each consisting of a pair of traits

$U \sim Uniform(0, 1)$
if $U < 0.5$ **then**
    $\langle o_1^1, o_1^2 \rangle \leftarrow cross(p_1^1, p_1^2)$
    $\langle o_2^1, o_2^2 \rangle \leftarrow (p_2^1, p_2^2)$
else
    $\langle o_2^1, o_2^2 \rangle \leftarrow cross(p_2^1, p_2^2)$
    $\langle o_1^1, o_1^2 \rangle \leftarrow (p_1^1, p_1^2)$

Algorithm 13 Simulated binary crossover

Input: two real-valued parent traits p_1, p_2
Output: two real-valued offspring traits o_1, o_2

$U \sim Uniform(0, 1)$
if $U \leq 0.5$ **then**
    $\beta = (2u)^{1/3}$
else
    $\beta = \frac{1}{(2(1-u))^{1/3}}$
$o_1 = \frac{p_1+p_2}{2} - \frac{\beta}{2}(i_2 - i_1)$
$o_2 = \frac{p_1+p_2}{2} + \frac{\beta}{2}(i_2 - i_1)$

Algorithm 14 Binary string crossover

Input: two binary vector parent traits, $\mathbf{p}^1$, $\mathbf{p}^2$
Output: two binary vector offspring traits, $\mathbf{o}^1$, $\mathbf{o}^2$

$U \sim Uniform\{1, \ldots, 100\}$
for each $i \in \{1, ..., 100\}$ **do**
    if $i < U$ **then**
        $o_i^1 \leftarrow p_i^1$
        $o_i^2 \leftarrow p_i^2$
    else
        $o_i^1 \leftarrow p_i^2$
        $o_i^2 \leftarrow p_i^1$

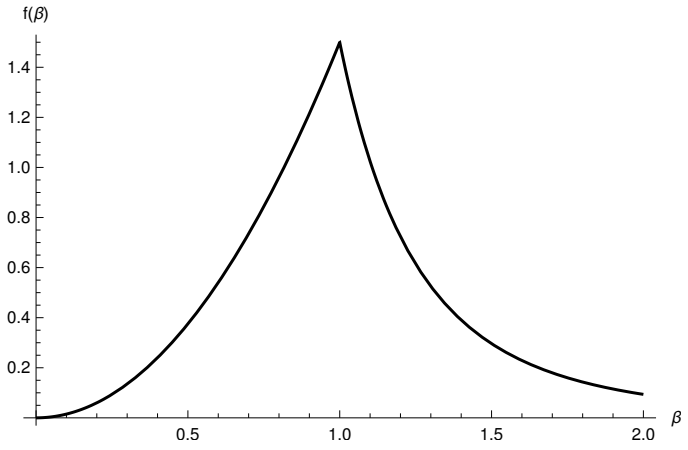

Figure 4.21: In simulated binary crossover, the trait of one individual becomes the mean value of the uncrossed trait minus $\beta/2$ times the difference of the uncrossed traits. The other becomes the mean value plus $\beta/2$ times the difference. The probability density function over β is as shown.

4.7 Experimental Design

I run separate experiments for each of the three evolvability selection methods, using the point-estimate method, or a Kalman filter or particle filter to estimate population evolvabilities, and for each of the four fitness functions. I compare each evolvability selection method to selection for fitness alone. Each algorithm, including selection for fitness alone, receives the same budget of fitness evaluations, and shares the parameters in Table 4.1. Each evolvability selection method shares the parameters in Table 4.2. The point-estimate method has the additional parameters in Table 4.3, and the Bayesian filter methods have the additional parameter in Table 4.4.

The right-hand column in each of these tables displays the probability distribution from which the parameters are sampled. I sample 10 000 times from these parameter distributions, each time running a single trial of the experiment with the sampled parameters. For each trial I record the *relative eventual fitness* and the *relative eventual evolvability*. The relative eventual

fitness is the mean fitness (averaged over all individuals) in the final generation of the EGS algorithm, minus the mean fitness achieved by selection for fitness alone.

I record eventual evolvability as follows. After the final generation of the algorithm, the fitness function is modified in such a way to ensure that individuals will be far from the optimal fitness value, without affecting their evolvabilities. For the SEM fitness function, no change occurs, since there is no global optimum. For the MM and SM fitness functions, this is achieved by performing 1000 fitness function change events as described in Sections 4.6.2 and 4.6.2, but modified such that only the primary bit string changes; the mutation mask and the symmetry probability parameter do not change. For the MVPR fitness function, this is achieved by performing a single fitness function change event with large δ, ensuring that the task to be performed changes. We then record the mean fitness before and after ten rounds of mutation and selection. The difference between the two values is the eventual evolvability. The *relative* eventual evolvability is the difference between the value achieved by the EGS algorithm and selection for fitness alone.

Table 4.1: Parameters shared by all algorithms, and the distributions they are sampled from.

Fitness evaluations E	$\sim$	Uniform($\{1000, 1500, \ldots, 10000\}$)
Population size N	$\sim$	Uniform($\{2, 4, 6, \ldots, 100\}$)
Tournament size k	$\sim$	Uniform($\{2, 4, 6, \ldots, N\}$)
Problem rate of change δ	$\sim$	Uniform($[0, 2]$)
Crossover probability p_c	$\sim$	Uniform($\{0, 0.7\}$)

My aim is to identify regions of the parameter space which cause selection for evolvability to lead to increased eventual fitness or evolvability over selection for fitness alone, and to calculate the probability that the identified

Table 4.2: Parameters shared by all evolvability selection methods, and the distributions they are sampled from.

Evolvability type ϵ	$\sim$	Uniform($\{\epsilon_\sigma, \epsilon^2_{max}\}$)
Heuristic H	$\sim$	Uniform($\{0, 1, 2\}$)
Number of populations[3] K	$\sim$	Uniform($\{2, \ldots, 5\}$)

Table 4.3: The additional parameters of the method that uses point estimates of evolvability.

Point estimate population size N'	$\sim$	Uniform($\{N, 2N, 3N, \ldots, 10N\}$)
Generations between selection M	$\sim$	Uniform($\{5, 6, 7, \ldots, 50\}$)

regions of parameter space really do lead to increased relative eventual fitness or evolvability. To do this, I split the collected data randomly into two evenly sized datasets.

I use the first dataset to train two *decision tree classifiers*. A decision tree classifier is a machine learning model to predict to which class an example data point belongs based on some number of features (Breiman et al., 1984; Quinlan, 1993). In this case, the classes in question are whether the relative eventual fitness (for the first classifier) or evolvability (for the second) is positive or negative. Figure 4.22 shows an example decision tree classifier used in this part of the data analysis. For example, the bottom two leaves on the left-hand side of the figure say that (given the decisions made above) a population size of greater than twenty-five leads to a prediction of a positive relative eventual fitness value, whereas a value less than or equal to twenty-five leads to a negative prediction.

[3]We only try K up to a value of 5 because a larger number of populations would put the EGS algorithm at an extreme disadvantage compared with selected for fitness alone due to the drastically increased number of fitness evaluations per generation.

Table 4.4: The additional parameter of the methods that use a Bayesian filter to estimate evolvability.

Certainty threshold P	$\sim$	Uniform($[0.6, 1.0]$)

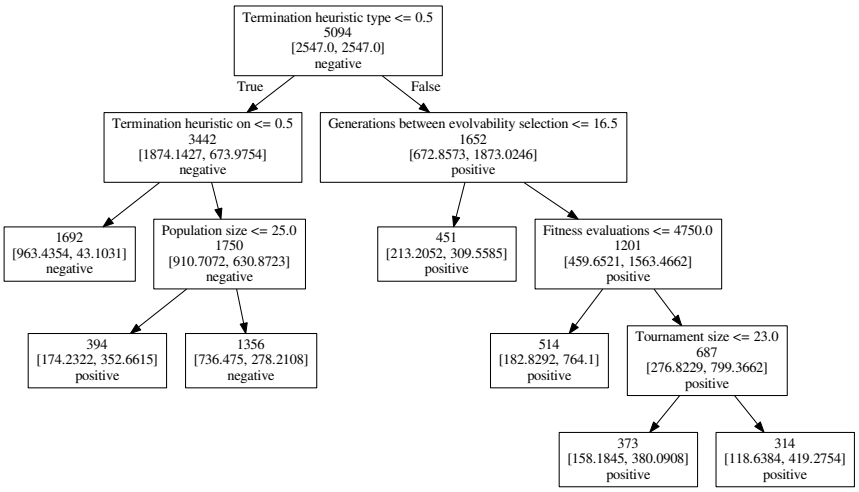

Figure 4.22: An example decision tree classifier, classifying the relative eventual fitness on the SEM problem when using a point estimate of evolvability.

Decision trees are highly interpretable. This means that we can see *why* the tree has made a particular decision. In this case, the tree allows us to see what are the important features of regions of the joint fitness function and algorithm parameter space which mean that selection for evolvability will lead to increased eventual fitness or evolvability compared to selection for fitness alone. I use the decision tree implemented in *scikit-learn*, a free machine learning library for Python (Pedregosa et al., 2011), which uses an optimized version of the CART algorithm (Breiman et al., 1984). The parameters used in the decision tree are the default values, except that the classes are weighted inversely proportionally to the class frequencies in order

to handle class imbalances, and the tree is prevented from creating leaf nodes that contain a lower proportion than 0.1 of the data points, weighted by the class weights. This last modification is to prevent overfitting.

The next step is to calculate a degree of belief that the two decision trees have identified regions of the parameter space in which relative eventual fitness or evolvability, respectively, really are positive. Once the tree has been built, the second dataset is passed through the decision trees, which aim to predict whether the relative eventual fitness or evolvability is positive or negative. For each of fitness and evolvability, this second dataset is then split into two groups, according to whether the corresponding decision tree predicts a positive or negative value. I then perform a Bayesian alternative to a t-test to calculate the probability that the two groups come from distributions with different true mean values (Kruschke, 2013).

This method involves modelling the two groups of data as coming from two t-distributions, with mean parameters μ_1 and μ_2, standard deviation parameters σ_1 and σ_2, and a shared degrees of freedom or 'normality' parameter ν. We use t-distributions rather than normal distributions because the heavy tails of the t-distribution for small ν cause the inferred value of the mean to be less seriously affected by outliers in the data.

We perform Bayesian estimation on these parameters $(\mu_1, \mu_2, \sigma_1, \sigma_2, \nu)$, so we must start with a prior distribution over each. I use the priors suggested by Kruschke. The prior distribution over each of the mean parameters, μ_1 and μ_2, is a normal distribution whose mean is the mean value of the pooled data (the data from the positive and negative groups pooled together), and whose standard deviation is 1000 times the standard deviation of the pooled data. The prior distribution over each of the standard deviation parameters, σ_1 and σ_2, is a uniform distribution over values ranging from 0.001 times to

1000 times the standard deviation of the pooled data. The prior distribution over the normality parameter ν is a shifted exponential given by $\frac{1}{29}\exp(-(\nu-1)/29)$ for $\nu \geq 1$. This distribution is designed by Kruschke to balance probability between heavy-tailed and nearly-normal t-distributions.

We then use Bayes rule to calculate posterior values over the parameters as follows.

$$p(\mu_1, \sigma_1, \mu_2, \sigma_2, \nu|D) = \frac{p(D|\mu_1, \sigma_1, \mu_2, \sigma_2, \nu)p(\mu_1, \sigma_1, \mu_2, \sigma_2, \nu)}{p(D)} \quad . \quad (4.31)$$

In words, the posterior probability of a particular setting of the t-distribution parameters equals the likelihood of the data given those parameter values multiplied by the prior probability of the parameter values divided by the evidence $p(D)$.

The evidence $p(D)$ can be thought of as the average likelihood of the results weighted by the prior, integrating over all possible values of the parameters. It can also be thought of as a normalization term, ensuring that the posterior is a probability distribution (i.e., that it integrates to 1). Since the evidence is calculated by integrating a complicated term over all possible values of the five parameters, it is not possible to compute analytically, so we use a *Markov Chain Monte Carlo* method to compute the posterior. I use a Python implementation (Straw, 2014) of the BEST software (Kruschke, 2013) to perform this computation.

From the posterior probability distribution over the t-distribution parameters, we can calculate our degree of belief that the two true mean values μ_1 and μ_2 are different. We can also calculate our degree of belief that the data categorized as having positive relative eventual fitness or evolvability really does have a mean value greater than zero—i.e., that in the region of the parameter space in question the EGS algorithm outperforms selection for

fitness alone.

The purpose of the first part of this analysis, the building of the decision trees, is to learn simple rules about how the fitness function and algorithm parameters affect whether selection for evolvability leads to increased eventual fitness or evolvability compared to selection for fitness alone. The purpose of the second part, the calculation of the probability that data classified as positive or negative by the trees come from distributions with different true means, is to determine the probability with which those rules are true. The reason a different dataset is used in the Bayesian estimation part of the analysis than that used to build the decision trees is that the rules used by the decision trees are partly due to random noise in the first dataset. Using the decision trees' classifications of the first dataset to group data, and performing Bayesian data analysis on that same data, would over-estimate the probability that these groups are sampled from distributions with different true mean values. The reasoning is the same as that leading to the use of separate training and testing datasets in machine learning (Ripley, 1996, p. 354).

4.8 Results

This section summarizes the results of the experiments described in the previous section. For each of relative eventual fitness and relative eventual evolvability, there are two tables. The first shows, for each combination of fitness function and evolvability estimation method, the probability that the mean relative eventual value of the group identified as 'positive' by the corresponding decision tree is greater than that of the group identified as 'negative', rounded to two decimal places. Recall that these probabilities are

Table 4.5: An index for the results in Tables 4.6 to 4.9, showing which table to reference for each of eventual fitness and eventual evolvability, and for each of the probability that the 'positive' group has a greater mean than the negative group and the probability that the 'positive' group has a positive mean.

	Eventual fitness	Eventual evolvability
Probability 'positive' greater than 'negative'	Table 4.6	Table 4.8
Probability 'positive' is positive	Table 4.7	Table 4.9

statements of our state of belief after performing Bayesian inference on the parameters of the distributions from which the results are drawn. When this probability is close to 1, this means we are confident that the decision tree has been able to identify a region of the parameter space that has a larger mean relative eventual value than the remainder of the parameter space.

The second table shows the probability that the group identified as 'positive' does in fact have a positive mean value. When this probability is close to 1, this means we are confident that the decision tree has been able to identify a region of the parameter space in which the EGS algorithm leads to greater eventual fitness or evolvability than selection for fitness alone. Table 4.5 provides an index for the four results tables. In each of the results tables, values greater than 0.95 are highlighted using boldface, and values less than 0.05 are highlighted using italics.

Appendix A provides more detailed statistics from these experiments. Appendix B provides, for each of relative eventual fitness and evolvability, for each of the fitness functions, and for each of the evolvability estimation methods, the decision tree that was used to separate data into 'positive' and 'negative' groups.

Figures 4.23 and 4.24 show some summary statistics of the distributions

representing our state of belief of the mean relative eventual fitness or evolvability values of the group labelled 'positive' by the corresponding decision tree. For each of the fitness functions, and for each of the evolvability estimation methods, the figures show the mode and the minimum and maximum value of the 95% highest density interval (HDI) of the posterior distribution of the mean.

As Tables 4.6 and 4.7 show, for every fitness function and for every evolvability estimation method, we believe with greater than 0.95 probability that the decision tree identifies a region of the parameter space in which the EGS algorithm leads to greater mean eventual fitness than selection for fitness alone. On the other hand, in every case, we believe with less than 0.05 probability that the region identified by the decision tree as having positive mean relative eventual fitness actually has a positive mean value. To summarize, we are able to find a region of parameter space that increases the EGS algorithm's performance, but are not able to find a region in which it outperforms selection for fitness alone.

Tables 4.8 and 4.9, however, show that the EGS algorithm is more successful in increasing mean eventual evolvability. In all but four of the twelve cases, we believe with greater than 0.95 probability that the 'positive' group has a greater mean relative eventual evolvability than the 'negative' group. In only two cases does our belief drop below 0.89. With the MVPR-particle filter pairing of fitness function and estimation method, our belief is 0.39. With the MVPR-Kalman filter pair it is 0.31. In all but two of the twelve cases, we believe that the 'positive' group actually has a positive mean value. For the MVPR-Kalman filter pair, our belief is 0.29. For the SEM-point estimate pair, our belief is 0.0.

Table 4.6: The probability that the 'positive' group has greater eventual fitness—relative to selection for fitness alone—than the 'negative' group.

	Point estimate	Kalman filter	Particle filter
SEM	**1.00**	**1.00**	**1.00**
MM	**1.00**	**1.00**	**1.00**
SM	**1.00**	**1.00**	**1.00**
MVPR	**1.00**	**1.00**	**1.00**

Table 4.7: The probability that the 'positive' group has positive eventual fitness relative to selection for fitness alone.

	Point estimate	Kalman filter	Particle filter
SEM	*0.00*	*0.00*	*0.00*
MM	*0.00*	*0.00*	*0.00*
SM	*0.00*	*0.00*	*0.00*
MVPR	*0.00*	*0.01*	*0.00*

Table 4.8: The probability that the 'positive' group has greater eventual evolvability—relative to selection for fitness alone—than the 'negative' group.

	Point estimate	Kalman filter	Particle filter
SEM	**1.00**	**1.00**	**1.00**
MM	**1.00**	**1.00**	**1.00**
SM	0.94	**1.00**	0.89
MVPR	**0.97**	0.31	0.39

Table 4.9: The probability that the 'positive' group has positive eventual evolvability relative to selection for fitness alone.

	Point estimate	Kalman filter	Particle filter
SEM	*0.00*	**1.00**	**1.00**
MM	**1.00**	**1.00**	**1.00**
SM	**0.99**	**1.00**	**1.00**
MVPR	**1.00**	0.29	**0.98**

Figure 4.23 shows that, as expected, the Kalman and particle filter outperform the point estimate method of evolvability estimation overall, in terms of increasing mean eventual fitness. The Kalman filter and particle filter outperform the point estimate method on both the SEM and MM fitness functions. The Kalman filter outperforms the other two methods on the SM fitness function. The particle filter outperforms the other two methods on the MVPR fitness function.

Figure 4.24 shows no clear winner amongst the evolvability selection methods in terms of increasing the mean eventual evolvability. Note that these plots should not be used to compare the behaviour of the algorithm across fitness functions, as the vertical axes are on different scales and are not normalized.

Looking more closely at the data in Appendix A, we can see each of the decision trees has an accuracy (proportion of data points labelled correctly as either 'positive' or 'negative') greater than 0.5, apart from the tree which aims to classify whether the mean relative eventual evolvability is positive in the case of the MVPR-Kalman filter experiment, where the accuracy is 0.49. We can also see that in five out of the twelve cases the empirical median value of the fitness data of the 'positive' group falls outside of our HDI for the mean. This suggests that the data may not be well described by a t-distribution, and warrants further investigation. The empirical median value of the evolvability data is always within our HDI of the mean, suggesting that this data may be well described by a t-distribution.

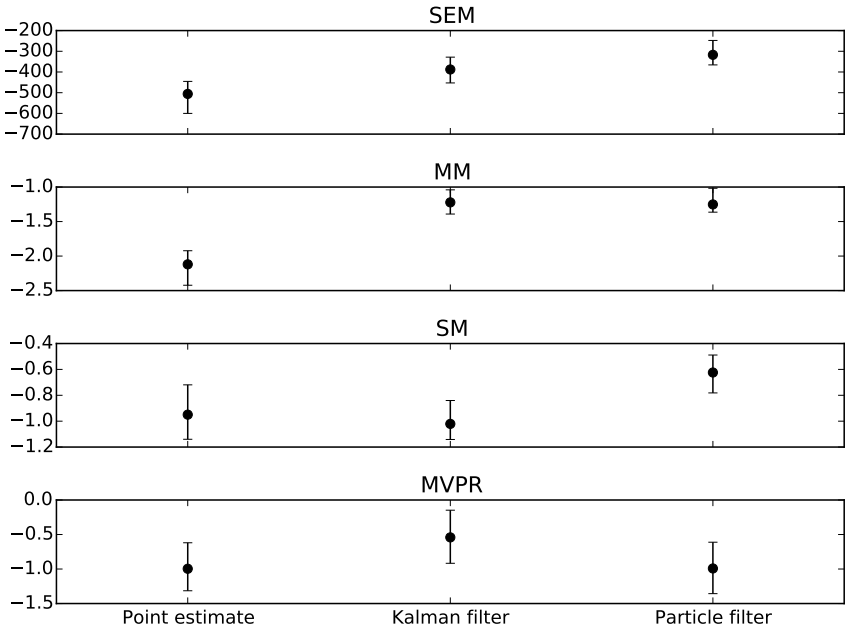

Figure 4.23: The mode and 95% highest density interval on the mean relative eventual fitness on each of the fitness functions, using each of the evolvability estimation methods.

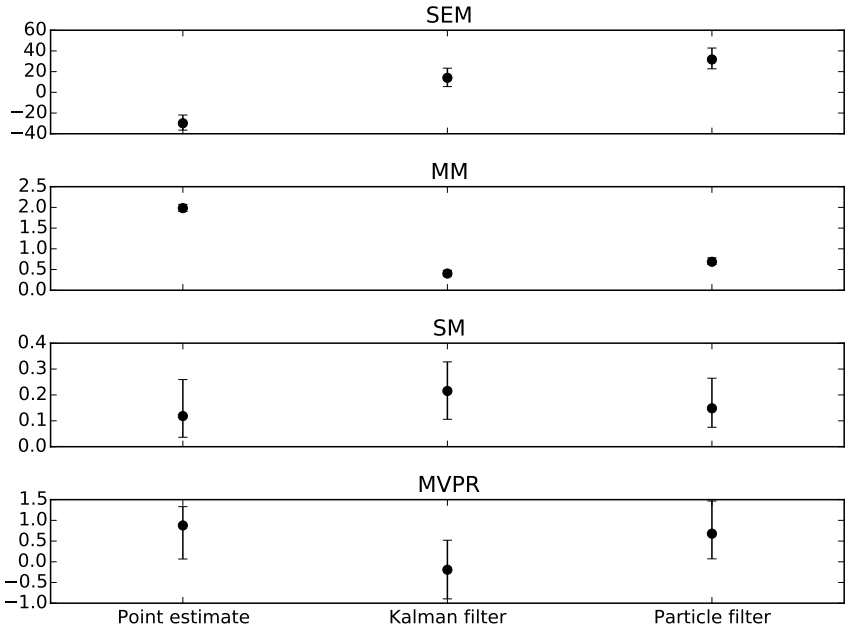

Figure 4.24: The mode and 95% highest density interval on the mean relative eventual evolvability on each of the fitness functions, using each of the evolvability estimation methods.

Looking at the decision trees in Appendix B, we find some patterns. First looking at the trees that categorize according to relative eventual fitness, nearly universally, across fitness functions and estimation methods, EGS only outperforms selection for fitness alone if a termination heuristic is used. This was expected, as the purpose of the termination heuristic is to more efficiently use fitness evaluations. This factor almost always appears as the root node of the tree. In many cases, that termination heuristic 2 is used is enough to ensure that the relative eventual fitness is classified as positive. Other important factors determining a positive relative eventual fitness are that the problem rate of change parameter δ is large, and that the population size N is small.

Turning to the decision trees which classify according to relative eventual evolvability, we find again that whether a termination heuristic is in use is the most important factor. That a termination heuristic is used is enough to classify the relative eventual evolvability as negative. This makes sense, as the termination heuristic causes the algorithm to stop selecting for evolvability before the end of the experiment. Other important factors determining a positive relative eventual evolvability are population size N and the certainty threshold P, though whether these should be small or large depends on the fitness function.

4.9 Limitations and Conclusion

In this chapter I introduced the episodic group selection (EGS) algorithm in an attempt to select for evolvability while making efficient use of a fixed fitness evaluation budget. I compared this algorithm to an algorithm that selects for fitness alone on four time-varying fitness functions. The findings

of this chapter are as follows.

1. We can identify regions of the parameter space in which the EGS algorithm outperforms selection for fitness alone in terms of the achieved eventual evolvability. However, we cannot identify regions of the parameter space in which the EGS algorithm outperforms selection for fitness alone in terms of the achieved eventual fitness.

2. The Kalman filter and particle filter evolvability estimation methods appear to outperform the point-estimate method in terms of the achieved eventual fitness.

3. The termination heuristics, which decide when to stop selecting for evolvability—and stop maintaining multiple populations—in order to use fitness evaluations more efficiently, appear to be crucial for increasing the achieved eventual fitness.

The limitations of the work described in this chapter are as follows. In place of null hypothesis significance testing, I performed a Bayesian analysis in which it is assumed that the relative eventual fitness and evolvability values of the experiments classified as 'positive', of those identified as 'negative', and of the data as a whole, follow t-distributions. However, in almost half of all cases, the empirical median of the 'positive' group lies outside of the 95% highest density interval on the mean of the t-distribution. This suggests that the data may not follow a t-distribution, and warrants further investigation.

The performance of the EGS algorithm on the MVPR fitness function differs between the case that a Kalman filter or a particle filter is used to track the population evolvabilities. Since these two sequential Bayesian filtering algorithms are supposed to be approximating the same posterior distribution,

it appears that one or both of the algorithms are approximating the posterior poorly. It may be that the Gaussian approximations of the likelihood functions used by the Kalman filter lead to a poor approximation, or that the repeated sampling of the particle filter are leading to a large sampling error. In either case, this warrants further investigation.

The purpose of the algorithm proposed in this chapter is to estimate population evolvabilities while making efficient use of fitness evaluations. We will see in the next chapter a modification of the algorithm designed to improve this efficiency.

Chapter 5

EGS With Asynchronous Reproduction

Chapter 4 describes episodic group selection (EGS) for evolvability, an algorithm in which we maintain K populations, use the fitness distributions within each population to estimate the population evolvabilities, and then periodically select the population with the highest estimated evolvability.

As we saw in Chapter 3, if we suppose that the estimation of evolvability does not require the use of extra fitness evaluations, then selection for evolvability can increase the eventual mean population fitness. However, as we saw in Section 4.1, and in the results in Section 4.8, the fact that our evolvability estimates are noisy, and that we must use fitness evaluations in order to calculate those estimates, means that in many cases we see no increase in eventual fitness as a result of selection for evolvability. This is because any increase in fitness due to increased evolvability must more than compensate for the increase by a factor of K in the number of fitness evaluations per generation in order to see any improvement overall. In the previous chapter we made an initial attempt to increase the efficiency with which the EGS algo-

rithm makes use of fitness evaluations by introducing termination heuristics, which determine a point at which we stop selecting for evolvability and stop maintaining multiple populations.

In this chapter, I describe another method for using fitness evaluations more efficiently in the EGS algorithm. Previously, each of the K populations went through the same number of generations synchronously between evolvability selection events. In the method described in this chapter, the populations can go through a different number of generations between evolvability selection events. The method is to use a recent *best arm identification* algorithm known as *pure exploration Thompson sampling* (Russo, 2016) to decide, at each step, which population should go through a generation of mutation and selection next.

5.1 Bandit Problems

A stochastic multi-armed bandit—named after the slot machines or "one-armed bandits" found in a casino—consists of a collection of arms $\mathcal{A} = \{1, \ldots, K\}$ (Berry and Fristedt, 1985). At each time t, we 'pull' one of the arms. When we pull arm k, we receive a reward drawn from some distribution ν_k with mean μ_k. At each time t, we take an action $a_t \in \mathcal{A}$ and observe a reward y_t drawn (independently of previous samples) from ν_{a_t}. Our choice of which arm to pull next depends on the sequence of actions and rewards seen so far $(a_{1:t-1}, y_{1:t-1})$. In a Bayesian setting, we maintain probability distributions representing our state of belief of the mean rewards of the K arms. After each action-reward pair, we update the distribution and use it to choose the next action.

The goal of a bandit problem is usually to minimize some measure of

regret. The immediate regret of pulling arm k is

$$R_k = \mu^* - \mu_k \quad , \tag{5.1}$$

where μ^* is the highest mean reward of the K arms. This regret is the difference between the expected reward of the best arm and the arm chosen. In typical bandit problems, the goal is to minimize the cumulative regret

$$R^T = \sum_{t=1}^{T} R_{a_t} \quad . \tag{5.2}$$

Another problem is *best arm identification*, in which there is a pure exploration phase of T steps (Audibert and Bubeck, 2010). The number of steps T may be fixed, or else the exploration phase terminates when our belief that one particular arm has the highest mean reward crosses some threshold probability. After the exploration phase, we make a single recommendation $\Omega(T) \in \mathcal{A}$, and the goal is to minimize the immediate regret of this decision, $R_{\Omega(T)}$.

5.2 Pure Exploration Thompson Sampling

Thompson sampling is a Bayesian algorithm for minimizing the cumulative regret. Thompson sampling proceeds by, at each step, choosing each arm with a probability equal to the probability that that arm has the highest mean reward. We can do this by sampling a vector $\boldsymbol{\theta}$ from our posterior distribution over the mean rewards, and then pulling the arm i where $i = \text{argmax}_k \, \theta_k$.

In its original form, this algorithm can perform poorly on best arm identification problems. Pure exploration Thompson sampling (PTS) is a modified version of Thompson Sampling where we reject our initial choice with

probability β, and repeatedly resample from the posterior until we choose a different arm (Russo, 2016). PTS is described in Algorithm 15.

Algorithm 15 Pure exploration Thompson sampling

Sample $\boldsymbol{\theta}$ from posterior.
$a_t \leftarrow \text{argmax}_k \, \theta_k$
Sample $U \sim \mathit{Uniform}([0, 1])$.
if $U < \beta$ **then**
    Take action a_t
else
    do
        Sample a new $\boldsymbol{\theta}$ from posterior.
        $a'_t \leftarrow \text{argmax}_k \, \theta_k$
    while $a'_t = a_t$
    Take action a'_t

Russo shows that, with $\beta = 1/2$, the probability of PTS ultimately choosing a sub-optimal arm decays exponentially with the number of pulls in the exploration phase, within a factor of two of the optimal exponent.

5.3 Episodic Group Selection with Asynchronous Reproduction

In the *episodic group selection with asynchronous reproduction* (EGS-AR) algorithm, we use the PTS best arm identification algorithm to choose which population will reproduce next. The true evolvabilities of the K populations are treated as the mean rewards of a K-armed bandit. Our evolvability observations are our observed rewards. Taking population k through one generation of reproduction corresponds to pulling the kth arm.

In the previous chapter, three methods were used to estimate the evolvabilities of the K populations: the point-estimate method, and two sequential Bayesian filtering algorithms. These filtering algorithms, the Kalman filter

and particle filter, maintain a joint probability distribution over the population evolvabilities. We can plug these distributions directly into the PTS best arm identification algorithm. For this reason, only the two sequential filtering estimation methods are used in this chapter.

As in the previous chapter, selection for evolvability occurs when we believe with greater than probability P that one population has greater evolvability than all others. When this occurs, we duplicate the population that we believe has the greatest evolvability to replace all other populations.

We still, optionally, use the termination heuristics described in the previous chapter. The next two sections describe how to choose the next population to reproduce in the case that we are using the Kalman filter or the particle filter to calculate the posterior distributions over the population evolvabilities.

5.3.1 Kalman filter

The Kalman filter represents our state of belief of the evolvabilities of the K populations by a multivariate Gaussian with mean vector $\mathbf{x}$ and covariance matrix $\mathbf{P}$.

We choose which population will reproduce next by sampling a vector θ from this multivariate Gaussian, and choosing population $i = argmax_k \theta_k$. With probability β we reject this choice, and repeatedly resample until we choose a different population.

5.3.2 Particle filter

The particle filter represents our state of belief of the evolvabilities of the K populations by a cloud of particles and associated weights. Each particle is a vector of length $K + 1$, with the first K elements representing possible

values of the evolvabilities of the K populations, and the $(K+1)$th element representing a possible value for the process noise variance q.

We choose which population will reproduce next by sampling each particle with probability equal to the associated weight, and choosing population $i = \text{argmax}_{k \in \{1,\ldots,K\}} \theta_k$, where $\boldsymbol{\theta}$ is the sampled particle. With probability β we reject this choice and repeat this procedure until we choose a different population.

5.4 Experimental Design

The experimental design is the same as that described in the previous chapter. For each of the evolvability estimation methods and for each of fitness functions, I sample 10 000 times from the parameter distributions, each time performing a single trial of the EGS-AR, EGS, and selection for fitness alone algorithms. I record the relative eventual fitness and evolvability of EGS-AR for each trial. I split the data into two datasets, use the first to build two decision trees to predict, based on the algorithm and fitness function parameters, whether the relative eventual fitness and evolvability will be positive or negative.

The second dataset is grouped according to the predictions of the two decision trees, and I perform Bayesian statistical tests to calculate a degree of belief that the mean values of the two groups are different, and that the mean value of the positive group is really positive. Additionally, I perform Bayesian estimation on the whole dataset to determine whether the mean eventual fitness and evolvability achieved by EGS-AR is greater than that achieved by EGS, averaged over the whole joint parameter distribution.

5.5 Results

This section summarizes the results from the experiments described in the previous section. For each of relative eventual fitness and relative eventual evolvability, there are two results tables. summarizing the performance of the EGS-AR algorithm relative to selection for fitness alone. The first shows, for each combination of fitness function and evolvability estimation method, the probability that the mean eventual value (relative to selection for fitness) of the group identified as 'positive' by the corresponding decision tree is greater than that of the group identified as 'negative', rounded to two decimal places. Recall that these probabilities are statements of our state of belief after performing Bayesian inference on the parameters of the distributions from which the results are drawn. When this probability is close to 1, this means we are confident that the decision tree has been able to identify a region of the parameter space that has a larger mean relative eventual value than the remainder of the parameter space.

The second results table shows the probability that the group identified as 'positive' does in fact have a positive mean value (relative to selection for fitness). When this probability is close to 1, this means we are confident that the decision tree has been able to identify a region of the parameter space in which the EGS-AR algorithm leads to greater eventual fitness or evolvability than selection for fitness alone.

Finally, for each of relative eventual fitness and relative eventual evolvability, there is a single results table summarizing the performance of the EGS-AR algorithm relative to the EGS algorithm. Since the EGS-AR algorithm was designed to outperform the EGS algorithm across the parameter space, these tables show the probability that the mean relative eventual value of the whole dataset is positive. When this probability is close to 1, we are

Table 5.1: An index for the results in Tables 5.2 to 5.7, showing which table to reference for each of eventual fitness and eventual evolvability, and for each of the probability that the 'positive' group has a greater mean relative to selection for fitness alone than the negative group, the probability that the 'positive' group has a positive mean relative to selection for fitness alone, and the probability that, averaging over the parameter distributions, the mean value is positive relative to the EGS algorithm.

	Eventual fitness	Eventual evolvability
Probability 'positive' greater than 'negative'	Table 5.2	Table 5.4
Probability 'positive' is positive	Table 5.3	Table 5.5
Probability EGS-AR outperforms EGS	Table 5.6	Table 5.7

confident that, averaged over the distribution from which the parameters were sampled, the eventual fitness or evolvability of the EGS-AR algorithm is greater than that of the EGS algorithm operating with the same parameters. Table 5.1 provides an index for the results tables. Values in the results tables greater than 0.95 are highlighted using boldface. Values less than 0.05 are highlighted using italics.

Appendix C provides more detailed statistics from these experiments. Appendix D provides the decision trees that were used to separate data into 'positive' and 'negative' groups.

Tables 5.2 and 5.3 tell much the same story as the results for the EGS algorithm in Section 4.8. In every case, we believe with probability greater than 0.95 that the corresponding decision tree has been able to identify a 'positive' region of parameter space where the mean relative eventual fitness is greater than in the remainder of the parameter space. On the other hand, in every case we believe with less than 0.05 probability that the 'positive' group actually has a positive mean value. In summary, while in every case

we are able to identify a region of the parameter space in which the EGS-AR algorithm performs well in terms of eventual fitness, in no case do we find a region in which the algorithm outperforms selection for fitness alone.

The results shown in Tables 5.4 and 5.5 are more encouraging than the corresponding results for the EGS algorithm. In all but three out of eight cases, we believe with probability greater than 0.95 that the 'positive' group has a greater mean value than the 'negative' group, with the lowest probability being 0.63 for the MVPR-Kalman filter experiment. More encouragingly, in every case we believe with probability greater than 0.95 that the 'positive' group has a positive mean value. In summary, for every fitness function and every evolvability estimation method, we are able to identify a region of the parameter space in which we believe that the EGS-AR algorithm leads to higher mean eventual evolvability than selection for fitness alone.

Tables 5.6 and 5.7 show the probability that, for equal parameter values, averaged over the parameter distributions given in Section 4.7, the mean eventual fitness or evolvability of the EGS-AR algorithm relative to the EGS algorithm is positive. In all but two of eight cases, we believe with probability greater than 0.95 that the mean relative eventual fitness is positive. In only one case, the MVPR-Kalman filter experiment, does this probability drop below 0.8. In only two of the eight cases do we believe with probability greater than 0.95 that the mean relative eventual evolvability is positive. However, in only one case does this probability drop below 0.7; for the MVPR-Particle filter experiment, this probability is 0.26. In summary, the EGS-AR algorithm outperforms the EGS algorithm across nearly all pairings of fitness function and evolvability estimation method.

Table 5.2: The probability that the 'positive' group has greater eventual fitness—relative to selection for fitness alone—than the 'negative' group.

	Kalman filter	Particle filter
SEM	**1.00**	**1.00**
MM	**1.00**	**1.00**
SM	**1.00**	**1.00**
MVPR	**1.00**	**1.00**

Table 5.3: The probability that the 'positive' group has positive eventual fitness relative to selection for fitness alone.

	Kalman filter	Particle filter
SEM	*0.00*	*0.00*
MM	*0.00*	*0.00*
SM	*0.00*	*0.00*
MVPR	*0.00*	*0.00*

Table 5.4: The probability that the 'positive' group has greater eventual evolvability—relative to selection for fitness alone—than the 'negative' group.

	Kalman filter	Particle filter
SEM	**1.00**	**1.00**
MM	**1.00**	**1.00**
SM	0.69	**1.00**
MVPR	0.63	0.86

Table 5.5: The probability that the 'positive' group has positive eventual evolvability relative to selection for fitness alone.

	Kalman filter	Particle filter
SEM	**1.00**	**1.00**
MM	**1.00**	**1.00**
SM	**1.00**	**1.00**
MVPR	**0.95**	**1.00**

Table 5.6: The probability that the mean eventual fitness—averaging over the joint algorithm and fitness function parameter distribution—is positive relative to the EGS algorithm.

	Kalman filter	Particle filter
SEM	0.80	**0.95**
MM	**0.99**	**0.97**
SM	**1.00**	**1.00**
MVPR	0.31	**1.00**

Table 5.7: The probability that the mean eventual evolvability—averaging over the joint algorithm and fitness function parameter distribution—is positive relative to the EGS algorithm.

	Kalman filter	Particle filter
SEM	0.88	0.92
MM	0.70	0.87
SM	0.90	**0.98**
MVPR	**0.95**	0.26

Looking more closely at the data in Appendix C, we can see that each of the decision trees used in this analysis has accuracy greater than 0.5. In two of eight cases and one of eight cases respectively, the empirical median value of the eventual fitness or evolvability relative to selecting for fitness alone falls outside of our mean HDI, suggesting that the data is not well described by a t-distribution. In two of eight cases, the same is true for the eventual fitness relative to the EGS algorithm.

Looking at the decision trees in Appendix D, specifically at the trees which classify the eventual fitness relative to selection for fitness alone, we see that, as in the previous chapter, whether or not a termination heuristic is in use is the most important factor. It almost every case, the use of a termination heuristic is necessary in order that the relative eventual fitness be classified as positive. This is expected, as the purpose of the heuristic is to use fitness evaluations more efficiently. Other factors which appear frequently are that the population size N should be small, that termination heuristic 2 should be used, and that the evolvability type should be ϵ_σ.

Turning to the decision trees which classify the eventual evolvability relative to selection for fitness alone, we see again that whether or not a termination heuristic is in use features prominently. The lack of a termination heuristic often leads to a positive classification. This is expected, since the termination heuristics stop the algorithm from selecting for evolvability before the end of the experiment. Other factors that feature prominently in the positive class are a small population size N and a low problem rate of change parameter δ. The evolvability type also features, but with the preferred type depending on the fitness function.

5.6 Limitations and Conclusion

In this chapter I introduced the episodic group selection with asynchronous reproduction (EGS-AR) algorithm, intended to make more efficient use of fitness evaluations than the EGS algorithm. I compared this algorithm to an algorithm that selects for fitness alone and to EGS on four time-varying fitness functions. The findings of this chapter are as follows.

1. We can identify regions of the parameter space in which the EGS-AR algorithm outperforms selection for fitness alone in terms of the achieved eventual evolvability. The results are stronger than those comparing the EGS algorithm to selection for fitness alone. However, we cannot identify regions of the parameter space in which the algorithm outperforms selection for fitness alone in terms of eventual fitness.

2. In most pairings of fitness function and evolvability estimation method, EGS-AR outperforms EGS in both eventual fitness and eventual evolvability, when averaged over the parameter distributions. Note that this fact depends on our choice of distributions over the parameters from which we sample when performing the experiments.

The limitations of the work described in this chapter are as follows. As in the previous chapter, in some cases the empirical median of the relative eventual fitness values lies outside of the 95% highest density interval on the mean of the t-distribution we are supposing describes the data. This suggests that the data may not follow a t-distribution, and warrants further investigation.

Again, as in the previous chapter, the performance of the EGS-AR algorithm on the MVPR fitness function depends on whether a Kalman filter

or a particle filter is tracking the population evolvabilities. Since these algorithms are supposed to be approximating the same posterior distribution, one or both of the algorithms appears to be approximating the distribution poorly. This warrants further investigation.

Chapter 3 established a kind of upper bound on our expectations of the performance of an algorithm that selects for evolvability. The EGS-AR algorithm represents a step towards such an upper bound, by outperforming the EGS algorithm when assigned an equal number of fitness evaluations. However, EGS-AR is still outperformed by selection for fitness alone in terms of eventual fitness. It would be useful to establish an upper bound of the performance of an algorithm that selects for evolvability on the four time-varying fitness functions the EGS and EGS-AR algorithms have been evaluated on. This could be done by discounting any fitness evaluations which are used to estimate and select for evolvability, thereby increasing the number of generations allocated to the algorithm, and recording the result. However, in the case of using a Kalman filter of particle filter to track the population evolvabilities, separating out fitness evaluations in terms of those used to select for fitness, and those used to estimate and select for evolvability, are not straightforward. This is because the evolvability estimates of the filters is based on a sequence of evolvability observations, and populations in which we measure a sequence of large evolvability observations will see large increases in fitness; when we select a population for having a large evolvability, we are partly selecting for fitness. We could determine an upper bound as described above by returning to the point-estimate evolvability estimation method described in the previous chapter, and using a very large poll population size N', while not counting any of the NN' fitness evaluations used in the evolvability selection step.

Chapter 6

Related Work

This chapter reviews the small number of publications that feature evolutionary algorithms in which selection for fitness is supplemented with or replaced by selection for evolvability estimates. The algorithms described differ in their definitions and measures of evolvability, but they share the same broad idea: to use offspring fitnesses or behaviours to estimate the evolvability of the parent or the evolvability of the offspring and to use this estimate during selection.

Some of the methods estimate evolvability by simple sampling. For example, if the evolvability of an individual is taken to be the probability that it produces an offspring fitter than itself, then we might estimate this by the proportion of the offspring of an individual which are fitter than itself. Other algorithms use more involved methods to calculate evolvability estimates.

With one exception, the algorithms described here work by producing a 'poll' population, as described in Chapter 4 and depicted in Figure 4.1. The poll population is used to estimate the evolvabilities of the parents and is then discarded. The next generation is then produced by selecting again from the parent population, this time for some combination of fitness and

estimated evolvability. As a short-hand, I will refer to these as *look-ahead* evolvability selection algorithms, since they work by looking ahead to see what the offspring fitness or behaviour distribution of each individual would look like before actually producing the next generation. The use of such poll populations increases the number of fitness evaluations per generation by at least a factor of two. Some of the authors are aware that, as discussed in Section 4.1, evolvability estimates will be noisy unless the sample size is large. As a result, they opt to use large poll populations, increasing the number of fitness evaluations per generation many-fold.

In the following, I describe the algorithms, the problems on which they are tested, and the findings of the authors. I also compare how the results of these algorithms are reported. Specifically, I will discuss whether these selection-for-evolvability algorithms are fairly compared to corresponding algorithms which select for fitness alone. Since the algorithms which select for evolvability can use many more fitness evaluations per generation, in order to fairly compare each algorithm should be allocated the same number of fitness evaluations rather than the same number of generations.

6.1 Estimation of Evolvability Genetic Algorithm

In what appears to be the first example of explicit selection for evolvability, Wang and Wineberg (2006) describe the *estimation of evolvability genetic algorithm* (EEGA). The focus of their research is to increase diversity within an evolutionary algorithm, and to prevent premature convergence. They select for evolvability in order to select for 'useful diversity' rather than for any kind of diversity. Of the algorithms discussed in this section, theirs is the

only one that does not use the look-ahead method described above, in which a poll offspring population is used in order to calculate the evolvabilities of the parents. In their algorithm, the next generation is constructed by selecting from the current generation three times; once for fitness, and once each for two measures of evolvability.

By their first measure of evolvability, which I'll call the *fitness difference*, the evolvability of an individual is the difference between its fitness and its parent's fitness. By their second measure, which I'll call the *genetic difference*, it is the degree to which the individual's genotype differs from its parent's.

Once an offspring population has been constructed by selecting three times for fitness, fitness difference, and genetic difference, the whole population goes through a further round of selection for fitness alone. A single iteration of this process is illustrated by Figure 6.1.

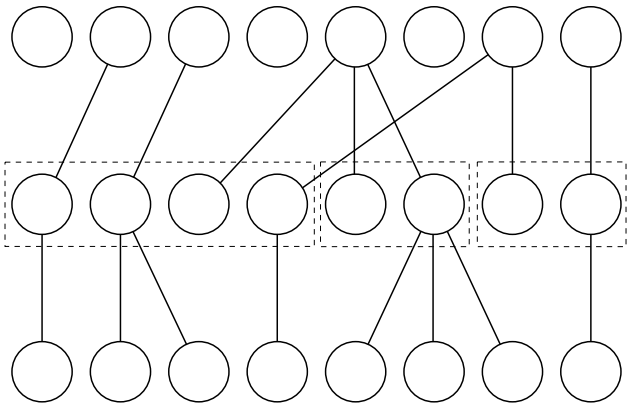

Figure 6.1: The EEGA algorithm. Lines indicate parent-offspring relationships, with offspring appearing below parents. Each of the subsets of the middle population are filled by selecting for a different criterion. The first is filled by selecting for fitness, the second by selecting for fitness difference, and the third by selecting for genetic difference. Then a third population is constructed by selecting for fitness from the whole middle population.

The purpose of selecting for genetic difference appears to be to produce

a population with a large amount of genetic variance, rather than genetic variability. The purpose of selecting for fitness differences may be to select for the (hidden) propensity of individuals to produce fitter offspring. However, their algorithm selects for an estimate of this underlying propensity using a sample size of one, and so the estimate is likely to be noisy, as we saw in Section 4.1.

They test their algorithm against a simple genetic algorithm that selects for fitness alone on a dynamic fitness function. The fitness function is a continuous function that undergoes smooth translation over time. They find that their algorithm is better able than selection for fitness alone to track the global optimum after an initial stationary period. Since their algorithm expends as many fitness evaluations per generation as selecting for fitness alone with the same population size, there is no need when comparing the algorithm with selection for fitness alone to adjust the number of generations allocated to each algorithm in order that each is allocated the same number of fitness evaluations; the comparison between EEGA and selection for fitness alone is fair. The authors conclude that the strategy is good, and note that the diversity of the populations produced by the EEGA algorithm is actually lower than that of the simple genetic algorithm, and conclude that it is not by increasing diversity alone that EEGA performs well. That EEGA outperforms the simple genetic algorithm agrees with the findings of Chapter 3, in which we see that selection for evolvability can increase eventual fitness. However, it is in disagreement with our findings in Chapters 4 and 5, where we fail to achieve higher eventual fitness by selecting for evolvability.

6.2 Recurrent Genetic Algorithm

Fakeih and Kattan (2012) use a simple method to estimate and select for evolvability by sampling, which they call a *recurrent genetic algorithm* (RGA). RGA uses a look-ahead method, using a poll offspring population to estimate the evolvability of each individual in the population.

The evolvability of an individual is defined as the expected mean fitness of its offspring. In order to estimate this, a poll population is constructed by selecting for fitness alone. The mean fitness of the offspring of each individual gives that individual's estimated evolvability. The poll population is discarded, and the next generation is constructed by selecting for estimated evolvability from the parent population.

Since the poll offspring population is the same size as the parent population, the mean number of offspring per individual is one. The result is that a very small number of samples from the offspring fitness distribution of each individual are used to estimate the mean offspring fitness; the evolvability estimates will be noisy.

They compare RGA to a simple genetic algorithm that selects for fitness alone. The algorithms are compared on several NK-landscape and hamming centres problems (unimodal and multimodal problems of tunable difficulty), and on three continuous problems. The fitness functions are not time-varying or open-ended. Broadly, the results show that RGA and the simple genetic algorithm have similar performance on the easier problems, and that RGA outperforms the simple GA on the harder problems. They find that the populations produced by RGA are more diverse than those produced by the simple GA, contradicting the results of Wang and Wineberg (2006).

In order that the comparison is fair, the authors account for the fact that RGA uses twice as many fitness evaluations per generation than selecting

for fitness alone with the same population size. They do this by halving the number of generations allocated to RGA. It is worth noting that the authors initially saw poor performance on the continuous fitness functions, and only saw improved performance after modifying the algorithm. The authors conclude that RGA has the potential to work well on real world optimization problems. That RGA outperforms the simple GA agrees with the findings of Chapter 3, in which we see that selection for evolvability can increase eventual fitness. However, it is in disagreement with our findings in Chapters 4 and 5, where we fail to achieve higher eventual fitness by selecting for evolvability.

6.3 Recurrent Bayesian Genetic Programming

In a later paper, Kattan and Ong (2015) use a Bayesian model to estimate the probability that an individual will produce offspring fitter than itself. They call the algorithm introduced in this paper *recurrent Bayesian genetic programming* (RBGP). This algorithm also uses the look-ahead method of producing a poll offspring population in order to sample from the offspring fitness distribution of the individuals in the parent population.

In RBGP, the evolvability of an individual is its probability of producing an offspring that is fitter than itself. For individual i we will call this probability p_i. Each individual produces 100 offspring, and the proportion of the offspring that are fitter than the parent is used to estimate the probabilities p_i. The poll population is then discarded and the next generation is constructed by selecting from the parent population for a combination of fitness and evolvability. The intention of the authors is to use Bayesian inference to infer the probabilities p_i from the fitness samples from the poll population,

but the inference method appears to be flawed.

In order to perform Bayesian inference of the probability p_i, we should start with some prior distribution $p(p_i)$. Our posterior distribution of p_i given the number of offspring N and the number s of those that were fitter than the parent would then be

$$p(p_i|N,s) = \frac{p(s|N,p_i)p(p_i)}{\int_0^1 p(s|N,p')p(p')\,\mathrm{d}p'} \quad , \tag{6.1}$$

where $p(s|N,p_i)$ is the likelihood that an individual whose probability of having a fitter offspring is p_i has s such offspring if it has N offspring in total. The number of offspring which are fitter, s, is a variate drawn from a binomial distribution $B(N,p_i)$, and so $p(s|N,p_i)$ is the probability density function of the binomial distribution with those parameters.

The authors use the maximum likelihood value of p_i, $\frac{s}{N}$, as their 'prior distribution'. This is a single value giving an estimate of the value of p_i rather than a distribution over the parameter which is to be inferred. Their 'likelihood' is a Gaussian distribution with parameters with values whose origin is uncertain. Their 'posterior' is also just a single number, which is an estimate of the value of p_i, rather than a distribution representing the updated state of belief of p_i.

The standard way to perform this particular inference would be to start with a beta distribution $Beta(1,1)$ prior over the probabilities p_i. Since the binomial distribution is conjugate to the beta distribution, the posterior distribution over the p_i would also be a beta distribution $Beta(1+s, 1+N-s)$.

Although the Bayesian inference in the RBGP algorithm appears to be flawed, their work seems to be the first use of Bayesian inference to estimate evolvabilities. The authors are also the first to suggest combining observa-

tions from multiple generations to produce better estimates, as is done by the EGS and EGS-AR algorithms described in Chapters 4 and 5.

The RBGP algorithm is tested against a simple genetic programming algorithm which selects for fitness alone. They are compared on fifteen problems, twelve of which are symbolic regression problems and three of which are time-series problems. The problems are not time-varying or open-ended. The RBGP algorithm achieves better median performance than the simple GP on nine of the fifteen problems. The margin between the performance of the simple GP and RBGP varies. The authors conclude that the results are promising, since their algorithm outperforms the simple GP in most cases, and loses by a small margin when it does not.

The RBGP algorithm uses a much larger sample size than the algorithms discussed so far when estimating evolvability, leading to less noisy evolvability estimates. But since each individual produces 100 offspring in the poll population, the algorithm uses 100 times as many fitness evaluations per generation. When comparing RBGP with a standard genetic programming algorithm that selects for fitness alone, the authors ensure that each method is given an equal number of fitness evaluations, but this is done by increasing the population size of the standard genetic programming algorithm by a factor of 100 rather than giving it more generations to work with, which may lead to unfair comparisons. That RBGP outperforms the simple GP agrees with the findings of Chapter 3, in which we see that selection for evolvability can increase eventual fitness. However, it is in disagreement with our findings in Chapters 4 and 5, where we fail to achieve higher eventual fitness by selecting for evolvability.

6.4 Modelling Evolvability in Genetic Programming

Fowler and Banzhaf (2016) use machine learning methods to model the evolvabilities of genetic programs that use a tree representation. I will refer to their approach here as *modelling evolvability in genetic programming* (MEGP). Initially, their method works using the look-ahead method by sampling from a poll offspring population which is then discarded.

In their algorithm, the evolvability of individual i is the probability p_i that it produces offspring whose fitness is greater than its own fitness. Initially, in order to estimate this, in each generation a poll offspring population is constructed in which each individual produces a large number of offspring, e.g., 1000 or 2000. For each individual, the proportion of these offspring which are fitter than the individual itself is used as an estimate of the probability p_i. The poll population is discarded and the next generation is constructed by selecting again from the parent population. Fowler and Banzhaf try two different selection schemes. The first is to select for evolvability only if the magnitude of the difference between the fitnesses of two individuals is below a threshold value. The second is to select for a weighted sum of fitness and evolvability, with the weight given to selecting for evolvability decaying over time. The weight given to selecting for evolvability decays for the same reason that we use the termination heuristics in the EGS and EGS-AR algorithm; early gains in evolvability have a greater impact on eventual fitness than later gains.

During the first phase of the algorithm, for each parent, several features of each parent program tree, such as the tree depth and number of dormant nodes, are recorded along with the estimated evolvability. Each of these is

used as a training example for an artificial neural network that is trained to predict the evolvability of an individual from properties of the tree.

After some number of generations, the algorithm stops producing the poll population and selects instead for the predicted evolvability produced by the artificial neural network. In the initial phase of the algorithm, the number of samples used to estimate evolvability is very large, which means the evolvability estimates are accurate, but that the number of fitness evaluations expended in each generation is increased by at least 1000-fold. After the training phase, the number of fitness evaluations per generation is the same as a standard genetic programming algorithm that selects for fitness alone with the same population size.

The algorithm is tested on some order tree problems, which are synthetic problems of tunable difficulty. The problems are neither time-varying nor open-ended. First, the authors show that if they run the algorithm entirely in its first phase, i.e., if the algorithm relies in every generation on an evolvability estimate estimated from a large sample of offspring, then the algorithm outperforms a simple genetic programming algorithm which selects for fitness alone. They then show that they achieve similar performance with the full algorithm, which at some point stops relying on samples and starts selecting for predicted evolvability; the MEGP algorithm still outperforms simple GP.

When comparing MEGP to a standard genetic programming algorithm without selection for evolvability, Fowler and Banzhaf do not account for the extra fitness evaluations per generation expended by MEGP. The fitness achieved by MEGP is compared with that achieved by standard GP at the corresponding generation. For this reason, the comparison may not be fair. That MEGP outperforms the simple GP agrees with the findings of Chap-

ter 3, in which we see that selection for evolvability can increase eventual fitness. However, it is in disagreement with our findings in Chapters 4 and 5, where we fail to achieve higher eventual fitness by selecting for evolvability.

6.5 Evolvability Search

Mengistu et al. (2016) describe the *evolvability search* (ES) algorithm. It uses the look-ahead method, producing a poll offspring population in order to estimate evolvability by sampling. Mengistu et al. claim that direct selection for evolvability estimates is an unexplored idea. As we have seen, this is not quite true.

Unlike the other algorithms described in this chapter, the evolvability measure used by the ES algorithm is not related to fitness. In the ES algorithm, the evolvability of an individual is its phenotypic variability, i.e., it's propensity to produce offspring displaying a wide range of behaviours. In their publication, the ES algorithm is tested on a map navigation problem and a bipedal robot locomotion problem, and evolvability is the propensity to produce offspring with a wide range of behaviours when solving these tasks.

In order to estimate the phenotypic variability or evolvability of an individual, a poll offspring population is produced, in which each parent individual produces 200 offspring. The measured evolvability of an individual is the total number of unique behaviours exhibited amongst that individual's offspring. The poll population is then discarded, and the next generation is produced by selecting from the parent population for evolvability alone. The motivation of Mengistu et al. is to produce evolvable phenotypes in order that they can be studied; they are less interested in their algorithm's ability to produce highly fit phenotypes.

Evolvability search is compared with an objective-based evolutionary algorithm and with novelty search (NS) on two robot control problems, in which the robots are to be controlled by artificial neural networks. The first is a maze navigation problem, in which the goal is to solve the maze, and the second is a bipedal locomotion problem, in which the goal is to travel as far as possible. The deceptive nature of the maze navigation problem prevents the objective-based algorithm from finding a solution in most cases. However, NS and ES are almost always able to find a solution. The authors note that the ES algorithm leads to a higher diversity of solutions than either objective-based search or NS. On the biped locomotion problem, ES finds solutions which outperform those found by objective-based search, but NS outperforms both. On both problems, ES produces more evolvable individuals than either of the other algorithms.

The authors conclude that, although ES is computationally expensive, direct selection for evolvability is viable. Since a large number of samples are used to estimate evolvability, the estimates should be accurate, at the cost of a 200-fold increase in the number of fitness evaluations per generation. In reporting their main results, Mengistu et al. do not account for the difference in the number of fitness evaluations per generation between the three algorithms; each algorithm is allocated the same number of generations. They note that the comparison might not be fair as a result, and subsequently report the evolvability achieved by objective-based search and NS if the algorithms are allowed to run for longer. They report that in this case the evolvability values plateau, and would likely not increase if the algorithms were allowed to run for longer still. However, due to computational constraints the algorithms are only allowed to run for 2.5 times longer than the default setting, and so the number of fitness evaluations allocated to

objective-based search and NS is still lower than that allocated to their ES algorithm, by a factor of around eighty. They also do not report the number of cases in which objective-based search or NS are able to solve the maze navigation problem if those algorithms are allowed to run for longer. That ES is able to achieve higher evolvability agrees with our findings from Chapters 3 to 5, where we find increased evolvability as a result of selecting for evolvability.

6.6 Comparison to This Work

All of the algorithms described in the preceding sections are shown to outperform a corresponding algorithm that selects for fitness alone on some fitness function. This is in agreement with the findings of Chapter 3, in which the analysis of a simple evolvability model reveals that selection for evolvability can increase eventual fitness. However, these results are not in agreement with the findings of Chapters 4 and 5, in which the EGS and EGS-AR algorithms fail to outperform selection for fitness alone.

Note though that of the related work described, only in that of Wang and Wineberg (2006) and Fakeih and Kattan (2012) is the proposed algorithm compared fairly against a benchmark algorithm. We may ask why these two algorithms were able to achieve higher eventual fitness than selection for fitness alone, while the algorithms described in Chapters 4 and 5 failed to do so. One common feature of these two algorithms is that they use very small sample sizes to estimate evolvability. Direct comparison to these two algorithms may not be possible. The EEGA algorithm of Wang and Wineberg (2006) does not use a poll offspring population in order to estimate the evolvabilities of the parent population. Instead, it uses the fitness differences between

the current generation and the previous generation in order to estimate the evolvabilities of the current generation. For this reason, their algorithm does not use more fitness evaluations per generation than the algorithm that selects for fitness alone. The RGA algorithm of Fakeih and Kattan (2012) is tested on fitness functions which are neither open ended nor time-varying, which all of the fitness functions described in Section 4.6 are. Moreover, the fitness functions on which RGA is tested are of tunable difficulty, and the improved performance of RGA over the baseline algorithm is only observed for problems of increased difficulty. The fitness functions used to test EGS and EGS-AR in Chapters 4 and 5 are not of tunable difficulty.

Mengistu et al. (2016) demonstrates increased evolvability as a result of selection for estimated evolvability, which agrees with all of the findings of this thesis.

The similarities and differences between the related algorithms described in this chapter and the EGS-AR algorithm described in Chapter 5 are summarized in Tables 6.1 to 6.3, with the EGS-AR algorithm displayed in boldface.

Table 6.1: A comparison of the algorithms described in this chapter and in Chapter 5. The columns give the evolvability estimation method, the evolvability measures used, and whether the algorithm produces a poll population in order to estimate evolvabilities (look-ahead), or not (look-back), or maintains multiple populations between evolvability selection events.

	Method	Evolvability measure	Structure
EEGA	Sampling	Fitness increase, genetic difference	Look-back
RGA	Sampling	Mean offspring fitness	Look-ahead
RBGP	Bayesian model	Probability of improvement	Look-ahead
MEGP	ML model	Probability of improvement	Look-ahead
ES	Sampling	Phenotypic variability	Look-ahead
EGS-AR	**Bayesian model**	**STD of offspring fitnesses, maximum offspring fitness**	**Multiple populations**

Table 6.2: A comparison of the algorithm described in this chapter and in Chapter 5. The columns give the sample size used to estimate evolvability, whether the algorithm combines estimates from multiple generations, and whether the fact that the evolvability selection algorithm uses more fitness evaluations is taken into account when comparing to selection for fitness alone.

	Sample size	Combines samples	Fair comparison
EEGA	1	No	Yes, evolvability selection uses same num. evaluations
RGA	Mean is 1	No	Yes, reduces num. generations
RBGP	100	No	Yes, reduces population size
MEGP	100, 1000 (varies)	No	No
ES	200	No	No
EGS-AR	**Mean is 1**	**Yes**	**Yes**

Table 6.3: A comparison of the algorithm described in this chapter and in Chapter 5. The columns give whether selection for evolvability becomes weaker over time, and whether a best-arm identification is used to efficiently decide which individuals should reproduce next.

	Decaying importance of evolvability selection	Best-arm identification
EEGA	No	No
RGA	No	No
RBGP	No	No
MEGP	Yes, decaying weight	No
ES	No	No
EGS-AR	**Yes, termination heuristics**	**Yes**

Chapter 7

Conclusion

This thesis posed two questions. The first was the extent to which we should select for evolvability if certain conditions are satisfied in order to maximize eventual fitness. One of those assumptions was that evolvability information is accurate and can be obtained at no cost in terms of fitness evaluations. The second question was, if that assumption were relaxed, whether the algorithms described in Chapters 4 and 5, which incorporate selection for evolvability, outperform selection for fitness alone in terms either of eventual evolvability or eventual fitness.

The motivation of this work was two-fold. Firstly, the algorithm proposed for the second research question might perform well on some time-varying optimization problems. Secondly, the results obtained from studying a simple model of evolvability selection, and the results obtained from the experiments comparing the proposed algorithm to selection for fitness alone, might help us to better understand the dynamics of selection for evolvability in natural evolution.

7.1 Challenges and Methods

In completing the work presented in this thesis, there was a long sequence of choices to be made. How should the populations of the EGS algorithm be structured? What algorithm should be used to estimate the population evolvabilities? On what fitness functions should the algorithms be tested? How should the EGS-AR algorithm decide which population to reproduce next?

Wherever possible, I have used existing algorithms and approaches, in order to decrease the number of novel features to be tested. The Kalman filter and particle filter algorithms, for example, which are used by the EGS and EGS-AR algorithms, are standard algorithms for sequential Bayesian filtering, and the four fitness functions on which the algorithms are tested are taken directly from work in the literature that aims to measure differences in long-term fitness due to differences in evolvability.

The most novel aspect of the work, apart from the theoretical model and the algorithms of Chapters 3 to 5 themselves, is the way in which the results are analysed. The EGS and EGS-AR algorithm, together with one of the four fitness functions, has nine or ten parameters, depending. Without any prior feeling for in which regions of the parameter space these algorithms would outperform selection for fitness alone, I sought to identify those regions. Unable to find a standard method for doing this, I devised my own. Taking a large number of samples from the specified joint distribution over the parameters, and running a single trial for each setting of the parameters, I used half of the results to train a decision tree (itself a well-known and interpretable method in machine learning), and grouped the rest of the results into a 'positive' and a 'negative' group, according to whether the decision tree predicted that the EGS or EGS-AR algorithm would outperform selec-

tion for fitness alone or not on those parameters. Having grouped the data in this way, I could then return to using a standard method from Bayesian statistics to determine a degree of belief that the mean value of the 'positive' group was greater than that of the 'negative' group, and also that the mean value of the 'positive' group was indeed positive. The end result is that the former probability tells us whether the decision tree has been able to find a region of the parameter space in which the algorithm performs well, and the latter probability tells us whether, in that region, the algorithm outperforms selection for fitness alone.

7.2 Findings

In Chapter 3, I studied a simple model of evolvability selection, and asked, within that model, the extent to which we should select for evolvability. I found that, if evolvability information is accurate and obtained without a cost in terms of fitness evaluations, then selecting for evolvability can increase eventual fitness.

In Chapter 4 I relaxed the assumption that evolvability information is both accurate and obtained for free, and examined the effect on the conclusions reached in studying the simple evolvability model. I introduced the EGS algorithm, designed to make efficient use of fitness evaluations in order to estimate and select for evolvability, and found by experiment that the EGS algorithm is capable of outperforming selection for fitness alone in terms of the eventual evolvability achieved, but not in terms of eventual fitness. The termination heuristics—an optional part of the EGS algorithm for deciding when to stop selecting for evolvability in order to make more efficient use of fitness evaluations—turned out to be crucial for increasing the eventual

fitness achieved by the algorithm.

The work of Chapter 5 is motivated by the fact that any increase in eventual fitness in the EGS algorithm due to increased evolvability appears to be more than compensated for by the increase in the number of fitness evaluations per generation. In that chapter, I introduce the EGS-AR algorithm, in which a recent best-arm identification algorithm is used to determine which algorithm will go through a generation of reproduction next. The purpose of the algorithm is to make more efficient use of fitness evaluations. I find that the EGS-AR algorithm outperforms the EGS algorithm in terms of both the eventual fitness and eventual evolvability achieved.

The research presented in this thesis does not rule out the utility of selecting for evolvability in order to increase eventual fitness. First it is demonstrated that selecting for evolvability can increase eventual fitness if evolvability information is accurate and obtained at no cost in terms of fitness evaluations. This establishes a kind of upper bound on our expectations of the performance of algorithms that select for evolvability; if eventual fitness had not been increased in this case, there would be no possibility that it would be increased once the cost of calculating evolvability estimates was taken into account. Then, the EGS algorithm was shown to outperform selection for fitness alone in terms of the eventual evolvability achieved, but not the eventual fitness. The termination heuristics are shown to be crucial to increasing the eventual fitness achieved by the EGS and EGS-AR algorithm. The EGS-AR algorithm represents a step towards the upper bound on performance established by the theoretical work, as it outperforms the EGS algorithm both in terms of eventual fitness and eventual evolvability, but still fails to achieve greater eventual fitness than selection for fitness alone.

The failure of the EGS and EGS-AR algorithms to outperform selection

for fitness alone in terms of eventual fitness disagrees with the findings of the related work described in Chapter 6. However, in only two of five cases are the proposed algorithms in the related work compared fairly to a corresponding algorithm that selects for fitness alone. One common feature of these two algorithms is the use of very small sample sizes to estimate evolvability, which is surprising given the argument presented in Section 4.1. That selection for evolvability estimates leads to increased evolvability agrees with the findings of Mengistu et al. (2016).

7.3 Limitations and Future Work

In this section I discuss the limitations and shortcomings of the work of this thesis, and give recommendations for future work intended either to remedy theses shortcomings or to take the research in new directions.

The study of the simple evolvability model in Chapter 3 is restricted to the case that the population size is $N = 2$ in order to simplify the evolutionary dynamics. The restriction eliminates any correlation between fitness and evolvability in the population, so there are no indirect selection effects. The model also assumes that evolvability information is accurate and comes at no cost in terms of fitness evaluations. It would be interesting to see an extension of the model in which either of these assumptions are relaxed.

In Chapter 3, we were restricted to considering constant values of the selection trade-off parameter γ. One possibility is to allow a selection trade-off parameter $\gamma(t)$ that is a function of the current generation number t. We expect the optimal $\gamma(t)$ to be monotonically increasing with t, selecting more strongly for evolvability towards the beginning of the simulation, and for fitness towards the end. Finding the optimal function $\gamma(t)$ may be a

difficult problem. Since the problem is that of obtaining a function which maximizes an expression containing that function within a sum, the correct approach may be related to the calculus of variations, which is concerned with finding a function to minimize a given integral.

In Chapters 4 and 5, in place of null hypothesis significance testing, I performed a Bayesian analysis in which it is assumed that the relative eventual fitness and evolvability values of the experiments classified as 'positive', of those identified as 'negative', and of the data as a whole, follow t-distributions. However, in some cases, the empirical median of the data lies outside of the 95% highest density interval on the mean of the inferred t-distribution. This suggests that the data may not be well described by a t-distribution, and warrants further investigation.

The performance of the EGS and EGS-AR algorithm on the MVPR fitness function differs between the case that a Kalman filter or a particle filter is used to track the population evolvabilities. Since these two methods should be approximating the same posterior distribution, it appears that one or both of the algorithms are approximating the distribution poorly. This warrants further investigation.

The simple evolvability model represents a kind of upper bound on the performance of an algorithm that selects for evolvability, since in that model we expend no fitness evaluations in measuring evolvability. The EGS-AR algorithm is an attempt to modify the EGS algorithm in order to step towards this upper bound. It appears to succeed, outperforming the EGS algorithm in terms of both eventual fitness and eventual evolvability. However, the EGS-AR algorithm fails to outperform selection for fitness alone in terms of eventual fitness. It would be useful to establish an upper bound on the performance of the EGS algorithm on the same four time-varying fitness

functions, by returning to the point-estimate evolvability estimation method, using a large poll population size N', and not counting towards the fixed fitness evaluation budget any of the NN' fitness evaluations used in the evolvability selection step. In doing so, we would be discounting any fitness evaluations that are used purely to estimate and select for evolvability.

Termination heuristics were introduced in Chapters 4 and 5, and the statistical analysis used indicated that whether or not a termination heuristic is in use is often the most important factor in determining the performance of the algorithm. Although this thesis has focused on making the sampling methods that are used in order to estimate evolvability more efficient in terms of the number of fitness evaluations used, we may be able to achieve improved performance by designing better termination heuristics.

7.4 Concluding Remarks

This thesis has shown that, in principle, selection for evolvability estimates may be a viable strategy for solving time-varying or open-ended problems. It has shown that, in the case that evolvability estimates are accurate and free, selection for evolvability can increase long-term fitness. Even when the cost of calculating evolvability estimates is taken into account, selection for evolvability is shown to increase long-term evolvability.

The work presented here may be of interest to any researcher interested in the evolution of evolvability in artificial systems. There are many such researchers, as we saw in Section 2.4.2. Direct selection for evolvability appears to be a novel topic; this thesis represents the sixth published work, by various authors, concerning direct selection for evolvability in evolutionary algorithms. All were published since 2006, and five were published since

2012. This work will be of direct interest to researchers interested in applying direct selection for evolvability in order to better solve time-varying or open-ended problems.

Bibliography

Aboitiz, F. (1991). Lineage selection and the capacity to evolve. *Medical hypotheses*, 36(2):155–156.

Alberch, P. (1991). From genes to phenotype: dynamical systems and evolvability. *Genetica*, 84(1):5–11.

Altenberg, L. (1994). The evolution of evolvability in genetic programming. In Kinnear, K. E., editor, *Advances in Genetic Programming*, Complex Adaptive Systems, pages 47–74, Cambridge. MIT Press.

Altenberg, L. (1995). Genome growth and the evolution of the genotype-phenotype map. In *Evolution and Biocomputation*, pages 205–259. Springer.

Audibert, J.-Y. and Bubeck, S. (2010). Best arm identification in multi-armed bandits. In *COLT-23th Conference on Learning Theory-2010*, pages 13–p.

Baugh, D. and McMullin, B. (2013). Evolution of G-P mapping in a von Neumann self-reproducer within Tierra. In *Advances in Artificial Life, ECAL*, volume 12, pages 210–217.

Bedau, M. and Packard, N. (1996). Measurement of evolutionary activity, teleology, and life.

Berry, D. A. and Fristedt, B. (1985). *Bandit problems: sequential allocation of experiments (Monographs on statistics and applied probability)*. Springer.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Ciliberti, S., Martin, O. C., and Wagner, A. (2007). Innovation and robustness in complex regulatory gene networks. *Proceedings of the National Academy of Sciences*, 104(34):13591–13596.

Clune, J., Mouret, J.-B., and Lipson, H. (2013). The evolutionary origins of modularity. *Proceedings of the Royal Society B: Biological Sciences*, 280(1755):20122863–20122863.

Conrad, M. (1979a). Bootstrapping on the adaptive landscape. *BioSystems*, 11(2):167–182.

Conrad, M. (1979b). Mutation-absorption model of the enzyme. *Bulletin of mathematical biology*, 41(3):387–405.

Conrad, M. (1990). The geometry of evolution. *BioSystems*, 24(1):61–81.

Dawkins, R. (2003). The evolution of evolvability. *On Growth, Form and Computers*, pages 239–255.

Deb, K. and Agrawal, R. B. (1994). Simulated binary crossover for continuous search space. *Complex Systems*, 9(3):1–15.

Draghi, J. and Wagner, G. (2009). The evolutionary dynamics of evolvability in a gene network model. *Journal of evolutionary biology*, 22(3):599–611.

Draghi, J. and Wagner, G. P. (2008). Evolution of evolvability in a developmental model. *Evolution*, 62(2):301–315.

Earl, D. J. and Deem, M. W. (2004). Evolvability is a selectable trait. *Proceedings of the National Academy of Sciences of the United States of America*, 101(32):11531–11536.

Ebner, M., Shackleton, M., and Shipman, R. (2002). How neutral networks influence evolvability. *Complexity*, 7(2):19–33.

Egri-Nagy, A. and Nehaniv, C. L. (2003). Evolvability of the genotype-phenotype relation in populations of self-replicating digital organisms in a Tierra-like system. In *Advances in Artificial Life*, pages 238–247. Springer.

Eiben, Á. E., Hinterding, R., and Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 3(2):124–141.

Eyre-Walker, A. and Keightley, P. D. (2007). The distribution of fitness effects of new mutations. *Nature Reviews Genetics*, 8(8):610–618.

Fakeih, A. and Kattan, A. (2012). Recurrent genetic algorithms: Sustaining evolvability. In *Evolutionary Computation in Combinatorial Optimization*, pages 230–242. Springer.

Fowler, B. and Banzhaf, W. (2016). Modelling evolvability in genetic programming. In *Genetic Programming*, pages 215–229. Springer.

Gallagher, A. (2009). *Evolvability: a formal approach*. PhD thesis, University of Oxford.

Hansen, T. F. (2006). The evolution of genetic architecture. *Annual Review of Ecology, Evolution, and Systematics*, pages 123–157.

Hansen, T. F., Pélabon, C., and Houle, D. (2011). Heritability is not evolvability. *Evolutionary Biology*, 38(3):258–277.

Hasegawa, T. and McMullin, B. (2013). Exploring the point-mutation space of a von Neumann self-reproducer within the Avida world. In *Advances in Artificial Life, ECAL*, volume 12, pages 316–323.

Holtzman, W. H. (1950). The unbiased estimate of the population variance and standard deviation. *The American journal of psychology*, 63(4):615–617.

Houle, D. (1992). Comparing evolvability and variability of quantitative traits. *Genetics*, 130(1):195–204.

Kálmán, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45.

Kashtan, N. and Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences of the United States of America*, 102(39):13773–13778.

Kashtan, N., Noor, E., and Alon, U. (2007). Varying environments can speed up evolution. *Proceedings of the National Academy of Sciences*, 104(34):13711–13716.

Kattan, A. and Ong, Y.-S. (2015). Bayesian inference to sustain evolvability in genetic programming. In *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems, Volume 1*, pages 75–87. Springer.

Kirschner, M. and Gerhart, J. (1998). Evolvability. *Proceedings of the National Academy of Sciences*, 95(15):8420–8427.

Knight, K. (2000). *Mathematical statistics*. Texts in statistical science series. Chapman & Hall/CRC Press, Boca Raton (Fla.).

Kruschke, J. K. (2013). Bayesian estimation supersedes the t test. *Journal of Experimental Psychology: General*, 142(2):573.

Labbe, R. (2016). Kalman and Bayesian filters in Python, GitHub repository, `https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python`, commit 5fcdf5af47b2f351e570473557eef6cf09f91922.

Lehman, J. and Miikkulainen, R. (2015). Extinction events can accelerate evolution. *PloS one*, 10(8):e0132886.

Levinton, J. (1988). Genetics: paleontology and macroevolution. Technical report.

Lewontin, R. C. (1970). The units of selection. *Annual Review of Ecology and Systematics*, pages 1–18.

Mahner, M. and Kary, M. (1997). What exactly are genomes, genotypes and phenotypes? And what about phenomes? *Journal of Theoretical Biology*, 186(1):55–63.

Marrow, P., Heath, M., and Re, I. I. (1999). Evolvability: Evolution, computation, biology. In *Proceedings of the 1999 Genetic and Evolutionary Computation Conference Workshop Program (GECCO-99 Workshop on Evolvability)*, pages 30–33. Citeseer.

Maynard Smith, J. (1970). Natural selection and the concept of a protein space. *Nature*, 225:563–64.

Maynard Smith, J. and Szathmáry, E. (1997). *The major transitions in evolution*. Oxford University Press.

McMullin, B. (2000). John von Neumann and the evolutionary growth of complexity: Looking backward, looking forward. *Artificial Life*, 6(4):347–361.

McMullin, B. (2012). Architectures for self-reproduction: Abstractions, realisations and a research program. In *Artificial Life*, volume 13, pages 83–90.

Mengistu, H., Lehman, J., and Clune, J. (2016). Evolvability search: Directly selecting for evolvability in order to study and produce it. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2016)*.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Nadarajah, S. and Kotz, S. (2008). Exact distribution of the max/min of two Gaussian random variables. *IEEE Transactions on very large scale integration (VLSI) systems*, 16(2):210–212.

Ofria, C. and Wilke, C. O. (2004). Avida: A software platform for research in computational evolutionary biology. *Artificial life*, 10(2):191–229.

O'Neill, M., Nicolau, M., and Brabazon, A. (2011). Dynamic environments can speed up evolution with genetic programming. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, pages 191–192. ACM.

Palmer, M. E. and Feldman, M. W. (2011). Spatial environmental variation can select for evolvability. *Evolution*, 65(8):2345–2356.

Palmer, M. E. and Feldman, M. W. (2012). Survivability is more fundamental than evolvability. *PloS one*, 7(6):e38025.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pelikan, M., Goldberg, D. E., and Lobo, F. G. (2002). A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20.

Pigliucci, M. (2008). Is evolvability evolvable? *Nature Reviews Genetics*, 9(1):75–82.

Quayle, A. P. and Bullock, S. (2006). Modelling the evolution of genetic regulatory networks. *Journal of Theoretical Biology*, 238(4):737–753.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Reisinger, J. and Miikkulainen, R. (2006). Selecting for evolvable representations. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, pages 1297–1304. ACM.

Reisinger, J., Stanley, K. O., and Miikkulainen, R. (2005). Towards an empirical measure of evolvability. In *Proceedings of the 2005 workshops on Genetic and evolutionary computation*, pages 257–264. ACM.

Ripley, B. (1996). Pattern recognition and neural networks.

Rudolph, G. (2001). Self-adaptive mutations may lead to premature convergence. *Evolutionary Computation, IEEE Transactions on*, 5(4):410–414.

Russo, D. (2016). Simple Bayesian algorithms for best arm identification. *arXiv preprint arXiv:1602.08448*.

Smith, T., Husbands, P., Layzell, P., and O'Shea, M. (2002). Fitness landscapes and evolvability. *Evolutionary Computation*, 10(1):1–34.

Sniegowski, P. D. and Murphy, H. A. (2006). Evolvability. *Current Biology*, 16(19):R831–R834.

Stanley, K. O. and Miikkulainen, R. (2003). A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130.

Straw, A. (2014). "best" software, GitHub repository, `https://github.com/strawlab/best`, commit 6964d17f3ce889a1899097bf6c53e569e8c9b245.

Toth, M. (2014). Mechanisms of non-genetic inheritance and psychiatric disorders. *Neuropsychopharmacology: official publication of the American College of Neuropsychopharmacology*.

Toussaint, M. (2002). On the evolution of phenotypic exploration distributions. In *FOGA*, pages 169–182. Citeseer.

Turney, P. (1999). Increasing evolvability considered as a large-scale trend in evolution.

Van Belle, T. and Ackley, D. H. (2002). Code factoring and the evolution of evolvability. In *GECCO*, volume 2, pages 1383–1390.

Van Belle, T. and Ackley, D. H. (2003). Adaptation and ruggedness in an evolvability landscape. In *Genetic and Evolutionary Computation—GECCO 2003*, pages 150–151. Springer.

Visser, J., Hermisson, J., Wagner, G. P., Meyers, L. A., Bagheri-Chaichian, H., Blanchard, J. L., Chao, L., Cheverud, J. M., Elena, S. F., Fontana, W., et al. (2003). Perspective: evolution and detection of genetic robustness. *Evolution*, 57(9):1959–1972.

Von Neumann, J. and Burks, A. W. (1966). *Theory of self-reproducing automata*. University of Illinois Press, Urbana.

Wagner, G. P. and Altenberg, L. (1996). Complex adaptations and the evolution of evolvability. *Evolution*, 50(3):967–976.

Wang, Y. and Wineberg, M. (2006). Estimation of evolvability genetic algorithm and dynamic environments. *Genetic Programming and Evolvable Machines*, 7(4):355–382.

Webb, A. M., Handl, J., and Knowles, J. (2015). How much should you select for evolvability? In *ECAL 13: The Thirteenth European Conference on Artificial Life*, volume 13, pages 487–494. MIT Press.

Webb, A. M. and Knowles, J. (2014). Studying the evolvability of self-encoding genotype-phenotype maps. In *ALIFE 14: The Fourteenth Conference on the Synthesis and Simulation of Living Systems*, volume 14, pages 79–86. MIT Press.

Wolfram Research, Inc. (2016). *Mathematica 10.4*.

Appendix A

EGS Data

This chapter lists the results from the experiments described in Section 4.7. The results tables are organized by fitness function and evolvability estimation method. For each, there are two tables, one for relative eventual fitness (eventual fitness for EGS minus eventual fitness when selecting for fitness alone), and one for relative eventual evolvability. The fields of the tables are as follows.

- **Proportion positive**: The proportion of experiments in which the relative eventual fitness or evolvability is positive. Note that this depends on the distributions from which the parameters were sampled from, including the EGS algorithm parameters, and so this number is not necessarily indicative of the performance of EGS compared with selection for fitness alone.

- **Accuracy**: The proportion of times the decision tree trained to classify the relative eventual fitness or evolvability as positive or negative correctly classifies the second dataset. Note that since only one trial is performed per experiment, the data is very noisy and so high accuracy

levels are not expected. Values greater than 0.6 are in boldface, and values less than 0.5 are in italics.

- **Sensitivity**: The proportion of times the decision tree correctly classifies the results in which the relative eventual fitness or evolvability was actually positive. Values greater than 0.6 are in boldface, and values less than 0.5 are in italics.

- **Specificity**: The proportion of times the decision tree correctly classifies the results in which the relative eventual fitness or evolvability was actually negative. Values greater than 0.6 are in boldface, and values less than 0.5 are in italics.

- **Negative empirical median**: The empirical median relative eventual fitness or evolvability of the results which were classified by the decision tree as having negative values. Values which lie outside of the inferred highest density interval (HDI), presumably because the data distribution is not normal, are in italics.

- **Negative mean HDI**: The highest density interval (95%) of the distribution representing our state of belief of the mean value of the group predicted to have negative relative eventual fitness or evolvability. This is the highest density interval in which we believe the mean value falls with probability 0.95.

- **Positive empirical median**: The empirical median relative eventual fitness or evolvability of the results which were classified by the decision tree as having positive values. Values which lie outside of the inferred highest density interval (HDI), presumably because the data distribution is not normal, are in italics.

- **Positive mean HDI**: The highest density interval (95%) of the distribution representing our state of belief of the mean value of the group predicted to have positive relative eventual fitness or evolvability. This is the highest density interval in which we believe the mean value falls with probability 0.95.

- **Different means probability**: The probability representing our belief that the positive group mean is greater than that the negative group. High values mean than we believe that the decision tree has, based on the algorithm and problem parameters, split the experiments into two groups with different mean result values. Probabilities greater than 0.95 are in boldface.

- **Positive mean probability**: The probability representing our belief that the positive group has a mean value of relative eventual fitness or evolvability that is actually positive. High values mean that we believe that the decision tree has identified a region of parameter space in which EGS achieves higher eventual fitness or evolvability than selection for fitness alone. Probabilities greater than 0.95 are in boldface.

Table A.1: SEM, point estimate, eventual fitness

Proportion positive	0.13
Accuracy, sensitivity, specificity	**0.69**, **0.86**, **0.66**
Negative empirical median	*-2008.08*
Negative mean HDI	[-1706.51, -1527.72]
Positive empirical median	*-720.63*
Positive mean HDI	[-600.02, -445.43]
Different means probability	**1.00**
Positive mean probability	0.00

Table A.2: SEM, point estimate, eventual evolvability

Proportion positive	0.38
Accuracy, sensitivity, specificity	0.54, **0.70**, *0.45*
Negative empirical median	-98.78
Negative mean HDI	[-113.19, -95.47]
Positive empirical median	-28.20
Positive mean HDI	[-36.49, -21.93]
Different means probability	**1.00**
Positive mean probability	0.00

Table A.3: SEM, Kalman filter, eventual fitness

Proportion positive	0.19
Accuracy, sensitivity, specificity	**0.68**, **0.77**, **0.65**
Negative empirical median	*-1389.81*
Negative mean HDI	[-1214.12, -1084.66]
Positive empirical median	-383.48
Positive mean HDI	[-452.84, -328.33]
Different means probability	**1.00**
Positive mean probability	0.00

Table A.4: SEM, Kalman filter, eventual evolvability

Proportion positive	0.45
Accuracy, sensitivity, specificity	0.56, 0.53, 0.59
Negative empirical median	-42.09
Negative mean HDI	[-54.44, -37.67]
Positive empirical median	13.39
Positive mean HDI	[5.55, 23.32]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table A.5: SEM, particle filter, eventual fitness

Proportion positive	0.20
Accuracy, sensitivity, specificity	**0.71**, **0.70**, **0.71**
Negative empirical median	*-1401.96*
Negative mean HDI	[-1276.86, -1135.96]
Positive empirical median	-256.03
Positive mean HDI	[-365.79, -247.68]
Different means probability	**1.00**
Positive mean probability	0.00

Table A.6: SEM, particle filter, eventual evolvability

Proportion positive	0.48
Accuracy, sensitivity, specificity	0.56, *0.49*, **0.64**
Negative empirical median	-31.28
Negative mean HDI	[-37.99, -25.70]
Positive empirical median	27.61
Positive mean HDI	[22.73, 42.79]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table A.7: MM, point estimate, eventual fitness

Proportion positive	0.19
Accuracy, sensitivity, specificity	**0.67**, **0.81**, **0.64**
Negative empirical median	-13.95
Negative mean HDI	[-14.42, -13.72]
Positive empirical median	*-1.90*
Positive mean HDI	[-2.42, -1.92]
Different means probability	**1.00**
Positive mean probability	0.00

Table A.8: MM, point estimate, eventual evolvability

Proportion positive	0.65
Accuracy, sensitivity, specificity	0.59, *0.43*, **0.86**
Negative empirical median	0.15
Negative mean HDI	[0.11, 0.23]
Positive empirical median	1.97
Positive mean HDI	[1.90, 2.07]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table A.9: MM, Kalman filter, eventual fitness

Proportion positive	0.27
Accuracy, sensitivity, specificity	**0.63**, **0.75**, 0.59
Negative empirical median	-6.87
Negative mean HDI	[-7.14, -6.66]
Positive empirical median	-1.20
Positive mean HDI	[-1.39, -1.04]
Different means probability	**1.00**
Positive mean probability	0.00

Table A.10: MM, Kalman filter, eventual evolvability

Proportion positive	0.55
Accuracy, sensitivity, specificity	0.54, 0.58, *0.50*
Negative empirical median	0.06
Negative mean HDI	[0.02, 0.14]
Positive empirical median	0.35
Positive mean HDI	[0.35, 0.48]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table A.11: MM, particle filter, eventual fitness

Proportion positive	0.26
Accuracy, sensitivity, specificity	**0.65**, **0.81**, 0.59
Negative empirical median	-7.48
Negative mean HDI	[-7.92, -7.45]
Positive empirical median	-1.28
Positive mean HDI	[-1.36, -1.02]
Different means probability	**1.00**
Positive mean probability	0.00

Table A.12: MM, particle filter, eventual evolvability

Proportion positive	0.56
Accuracy, sensitivity, specificity	0.54, *0.40*, **0.73**
Negative empirical median	0.07
Negative mean HDI	[-0.01, 0.10]
Positive empirical median	0.75
Positive mean HDI	[0.62, 0.78]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table A.13: SM, point estimate, eventual fitness

Proportion positive	0.25
Accuracy, sensitivity, specificity	**0.69**, **0.85**, **0.64**
Negative empirical median	*-20.95*
Negative mean HDI	[-20.09, -19.07]
Positive empirical median	*-0.55*
Positive mean HDI	[-1.14, -0.72]
Different means probability	**1.00**
Positive mean probability	0.00

Table A.14: SM, point estimate, eventual evolvability

Proportion positive	0.52
Accuracy, sensitivity, specificity	0.50, *0.32*, **0.69**
Negative empirical median	0.02
Negative mean HDI	[-0.05, 0.11]
Positive empirical median	0.08
Positive mean HDI	[0.04, 0.26]
Different means probability	0.94
Positive mean probability	**0.99**

Table A.15: SM, Kalman filter, eventual fitness

Proportion positive	0.29
Accuracy, sensitivity, specificity	**0.61**, **0.97**, *0.45*
Negative empirical median	-13.28
Negative mean HDI	[-13.67, -12.94]
Positive empirical median	*-0.75*
Positive mean HDI	[-1.14, -0.84]
Different means probability	**1.00**
Positive mean probability	0.00

Table A.16: SM, Kalman filter, eventual evolvability

Proportion positive	0.51
Accuracy, sensitivity, specificity	0.51, *0.38*, **0.65**
Negative empirical median	0.03
Negative mean HDI	[-0.08, 0.09]
Positive empirical median	0.16
Positive mean HDI	[0.11, 0.33]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table A.17: SM, particle filter, eventual fitness

Proportion positive	0.29
Accuracy, sensitivity, specificity	**0.65**, **0.88**, 0.56
Negative empirical median	-10.99
Negative mean HDI	[-11.52, -10.93]
Positive empirical median	*-0.40*
Positive mean HDI	[-0.78, -0.49]
Different means probability	**1.00**
Positive mean probability	0.00

Table A.18: SM, particle filter, eventual evolvability

Proportion positive	0.52
Accuracy, sensitivity, specificity	0.51, 0.54, *0.48*
Negative empirical median	0.09
Negative mean HDI	[-0.00, 0.16]
Positive empirical median	0.19
Positive mean HDI	[0.08, 0.26]
Different means probability	0.89
Positive mean probability	**1.00**

Table A.19: MVPR, point estimate, eventual fitness

Proportion positive	0.40
Accuracy, sensitivity, specificity	0.56, 0.53, 0.57
Negative empirical median	-3.29
Negative mean HDI	[-3.56, -2.94]
Positive empirical median	-0.86
Positive mean HDI	[-1.31, -0.62]
Different means probability	**1.00**
Positive mean probability	0.00

Table A.20: MVPR, point estimate, eventual evolvability

Proportion positive	0.51
Accuracy, sensitivity, specificity	0.52, **0.63**, *0.40*
Negative empirical median	-0.47
Negative mean HDI	[-0.90, 0.69]
Positive empirical median	0.87
Positive mean HDI	[0.07, 1.33]
Different means probability	**0.97**
Positive mean probability	**1.00**

Table A.21: MVPR, Kalman filter, eventual fitness

Proportion positive	0.42
Accuracy, sensitivity, specificity	0.55, *0.45*, **0.63**
Negative empirical median	-2.11
Negative mean HDI	[-2.34, -1.79]
Positive empirical median	-0.56
Positive mean HDI	[-0.92, -0.15]
Different means probability	**1.00**
Positive mean probability	0.01

Table A.22: MVPR, Kalman filter, eventual evolvability

Proportion positive	0.50
Accuracy, sensitivity, specificity	*0.49*, 0.52, *0.47*
Negative empirical median	0.19
Negative mean HDI	[-0.90, 0.72]
Positive empirical median	-0.14
Positive mean HDI	[-0.90, 0.52]
Different means probability	0.31
Positive mean probability	0.29

Table A.23: MVPR, particle filter, eventual fitness

Proportion positive	0.40
Accuracy, sensitivity, specificity	0.55, 0.57, 0.55
Negative empirical median	-2.90
Negative mean HDI	[-3.03, -2.44]
Positive empirical median	-0.85
Positive mean HDI	[-1.36, -0.61]
Different means probability	**1.00**
Positive mean probability	0.00

Table A.24: MVPR, particle filter, eventual evolvability

Proportion positive	0.52
Accuracy, sensitivity, specificity	*0.50*, 0.60, *0.39*
Negative empirical median	0.59
Negative mean HDI	[-0.09, 1.71]
Positive empirical median	0.40
Positive mean HDI	[0.07, 1.47]
Different means probability	0.39
Positive mean probability	**0.98**

Appendix B

EGS Decision Trees

This chapter provides the decision trees that are used as part of the data analysis pipeline for the experiments described in Section 4.7. The purpose of these trees is to identify regions of the parameter space in which the EGS algorithm leads to greater mean eventual fitness or evolvability than selection for fitness alone. The feature labels for the termination heuristic in the trees have the following meaning. 'Termination heuristic on' is a feature whose value is one if a termination heuristic is used and zero otherwise. 'Termination heuristic type' takes value zero for termination heuristic 1 and the value one for termination heuristic 2.

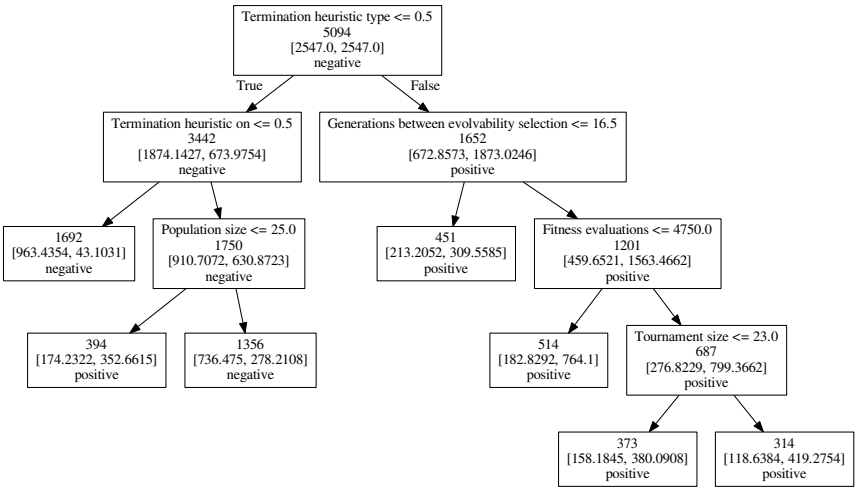

Figure B.1: SEM, point estimate, eventual fitness

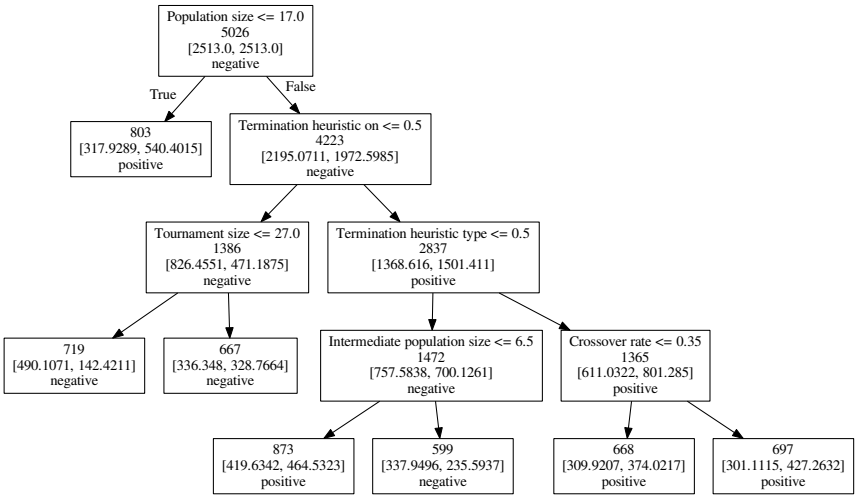

Figure B.2: SEM, point estimate, eventual evolvability

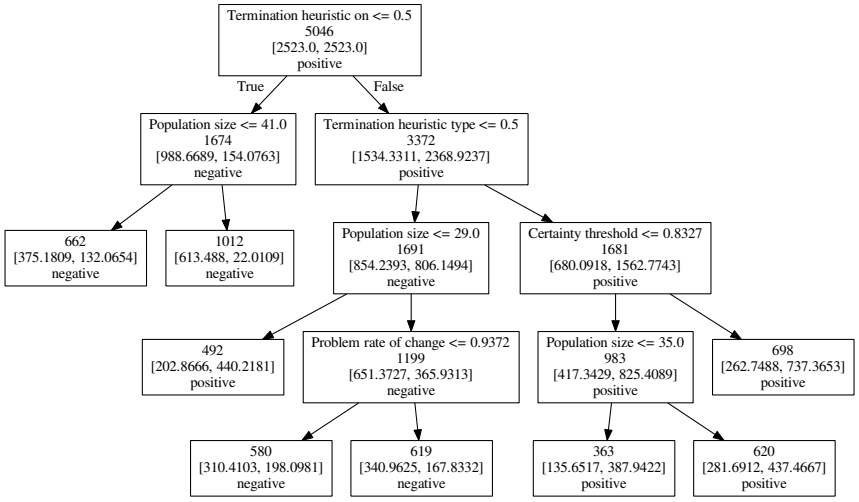

Figure B.3: SEM, Kalman filter, eventual fitness

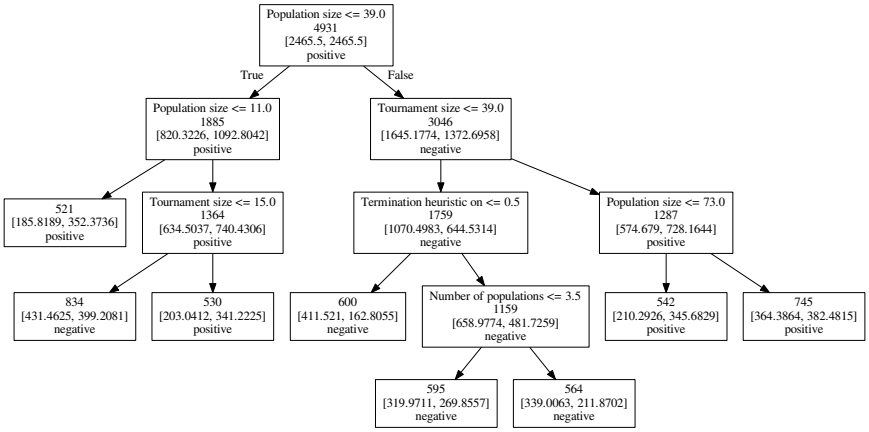

Figure B.4: SEM, Kalman filter, eventual evolvability

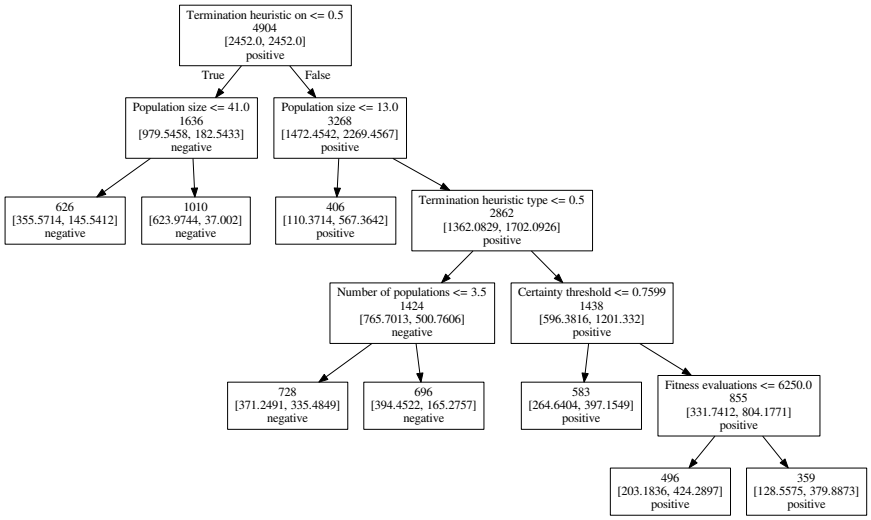

Figure B.5: SEM, particle filter, eventual fitness

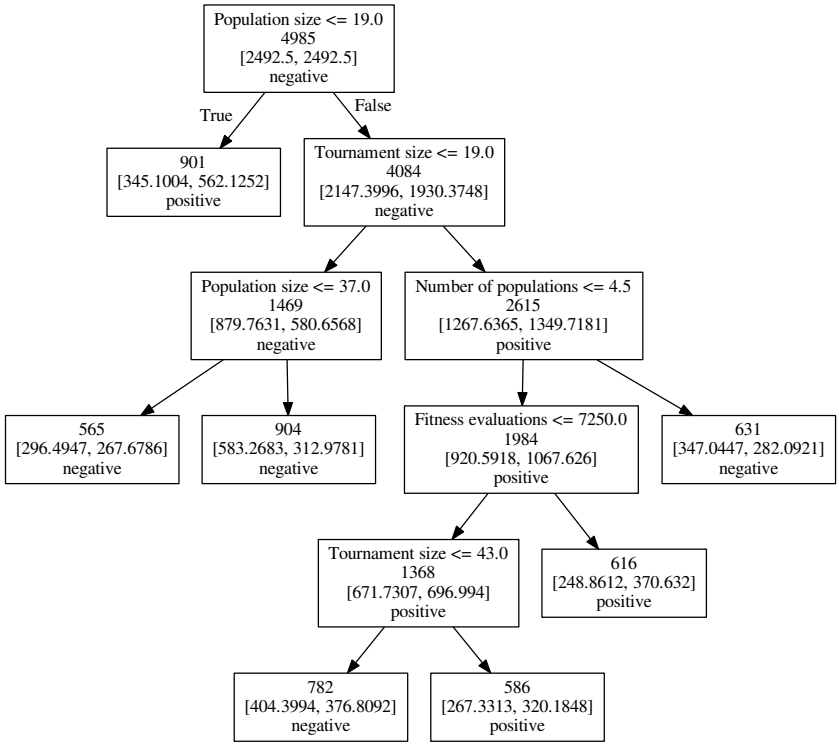

Figure B.6: SEM, particle filter, eventual evolvability

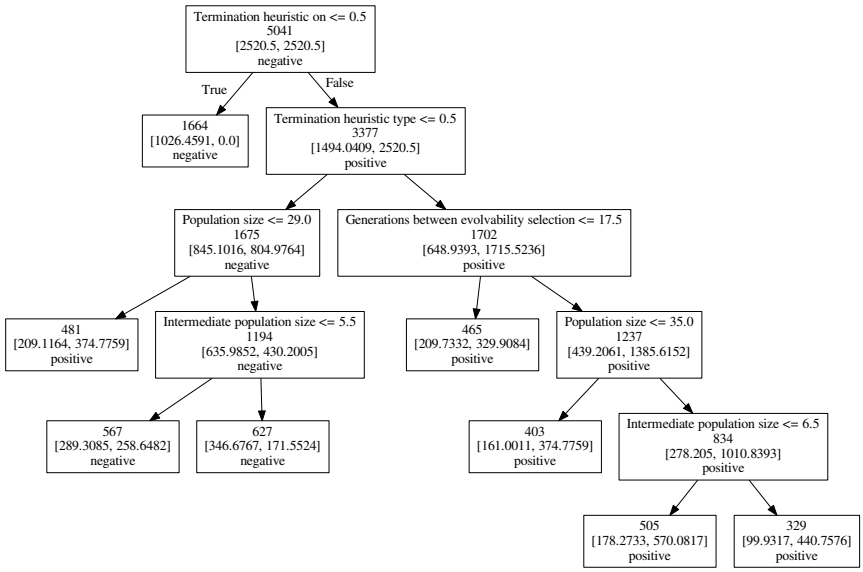

Figure B.7: MM, point estimate, eventual fitness

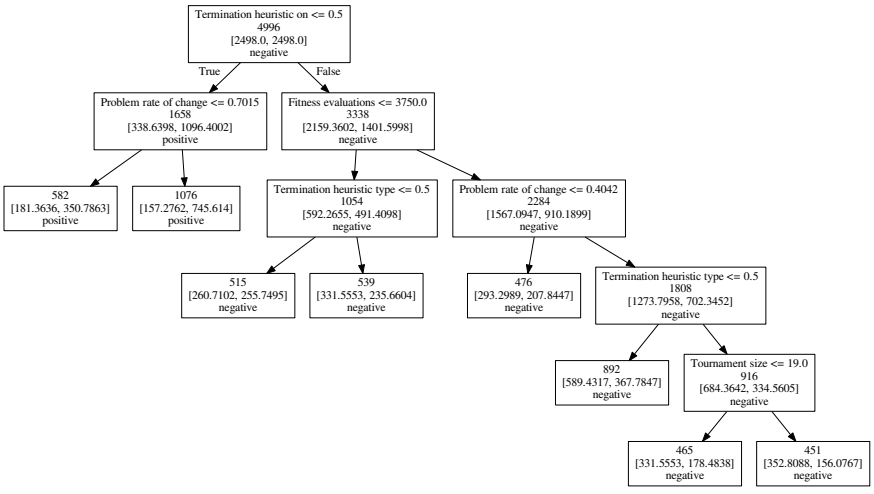

Figure B.8: MM, point estimate, eventual evolvability

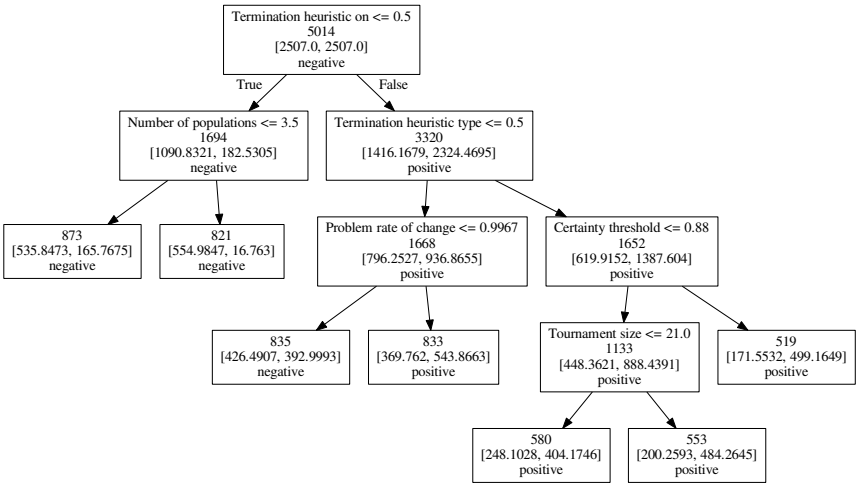

Figure B.9: MM, Kalman filter, eventual fitness

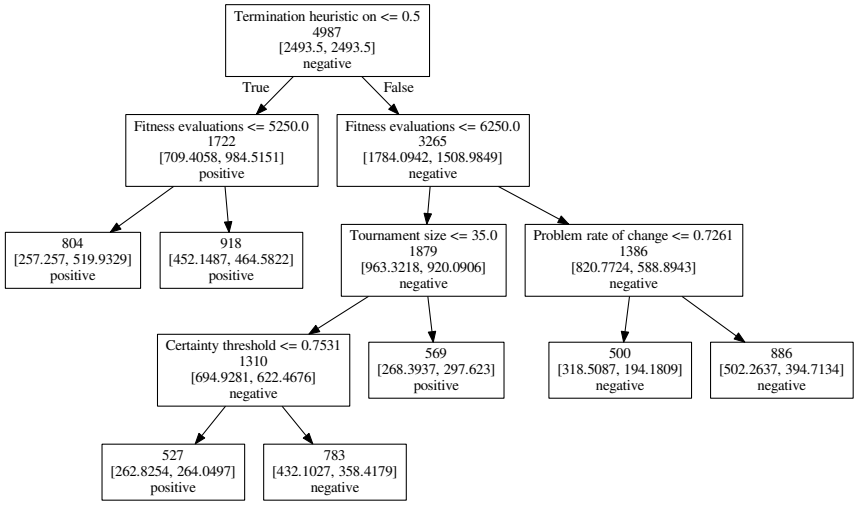

Figure B.10: MM, Kalman filter, eventual evolvability

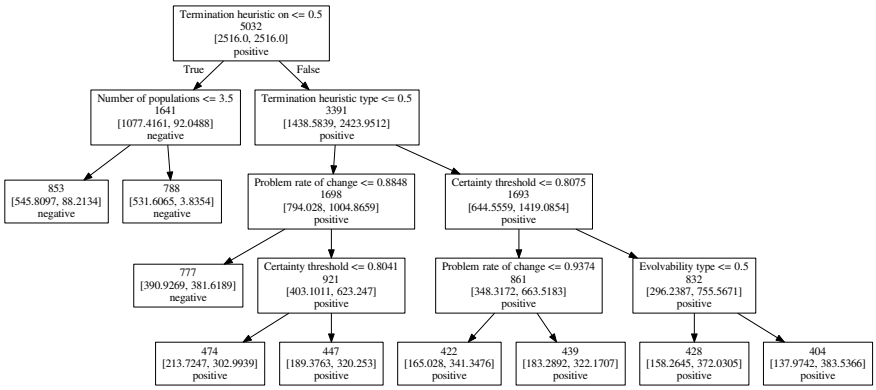

Figure B.11: MM, particle filter, eventual fitness

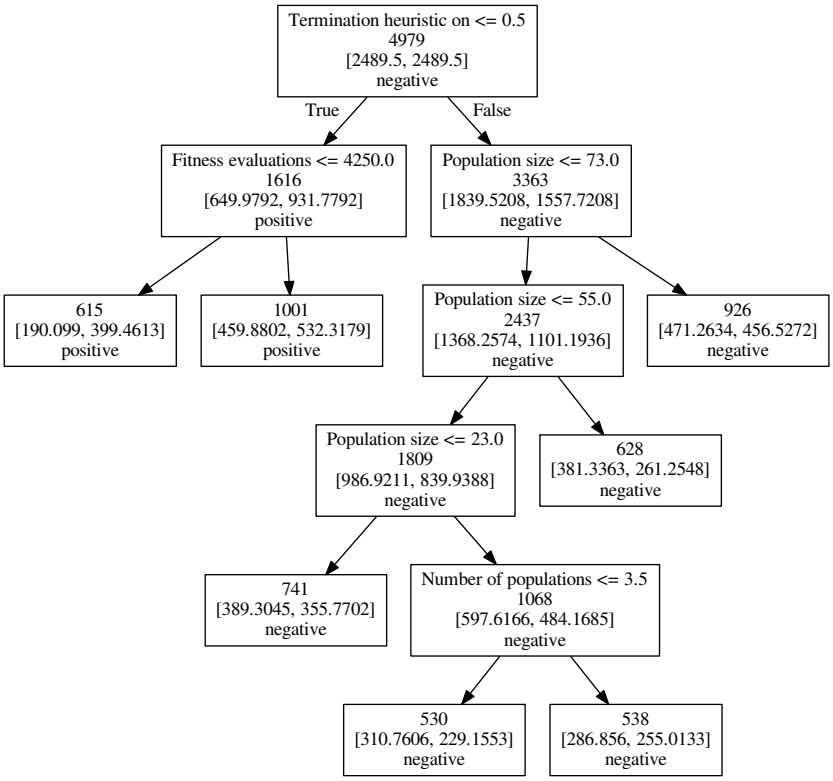

Figure B.12: MM, particle filter, eventual evolvability

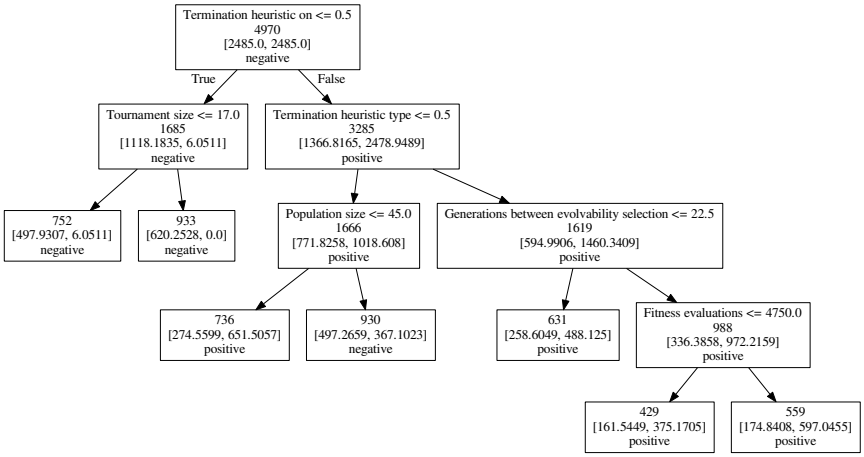

Figure B.13: SM, point estimate, eventual fitness

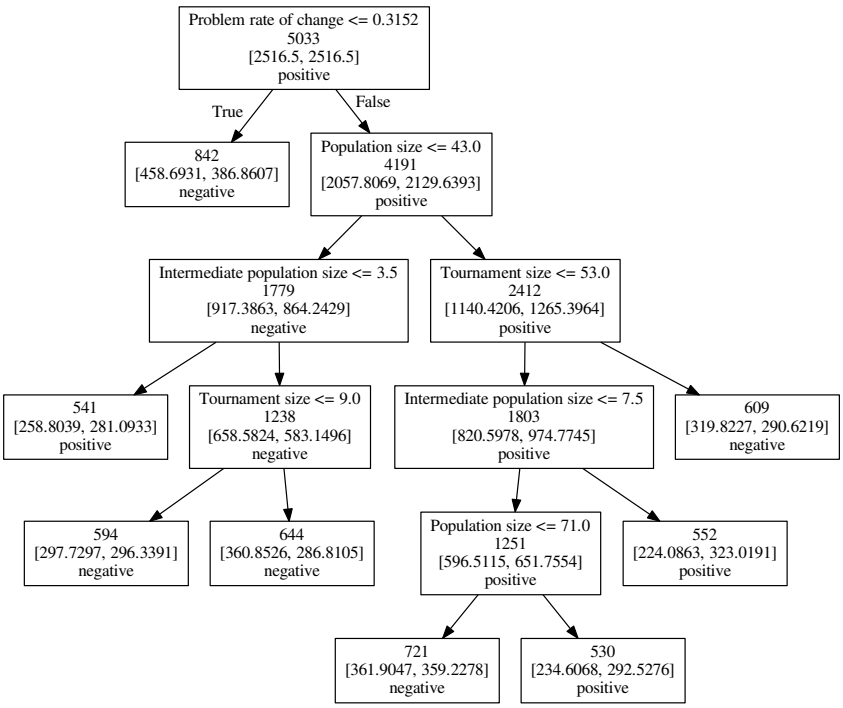

Figure B.14: SM, point estimate, eventual evolvability

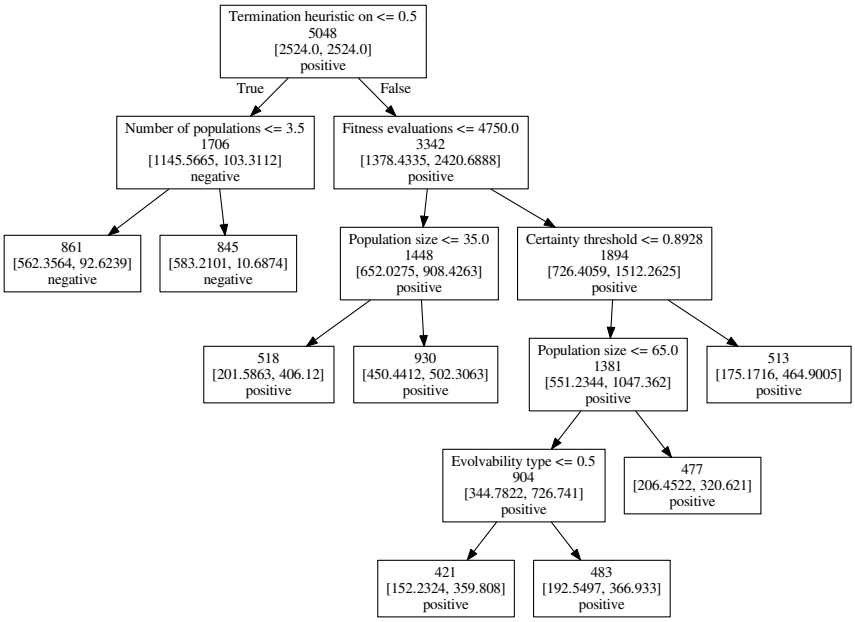

Figure B.15: SM, Kalman filter, eventual fitness

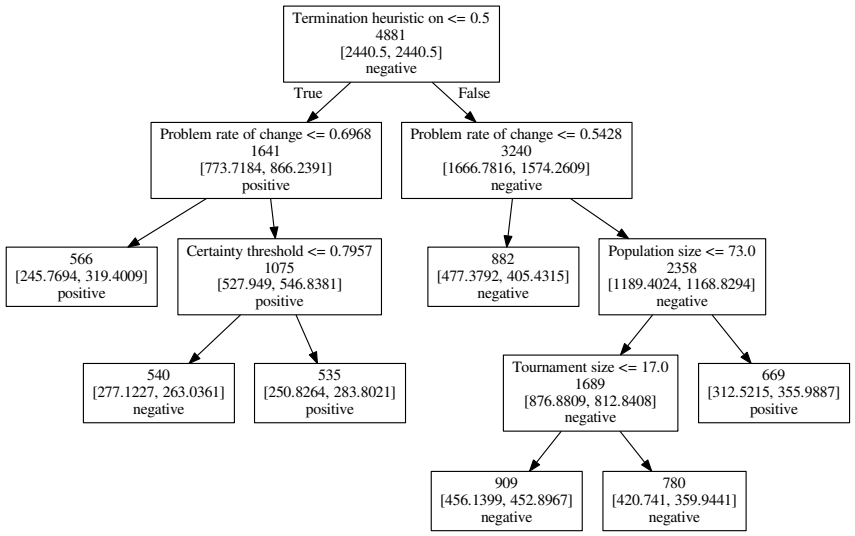

Figure B.16: SM, Kalman filter, eventual evolvability

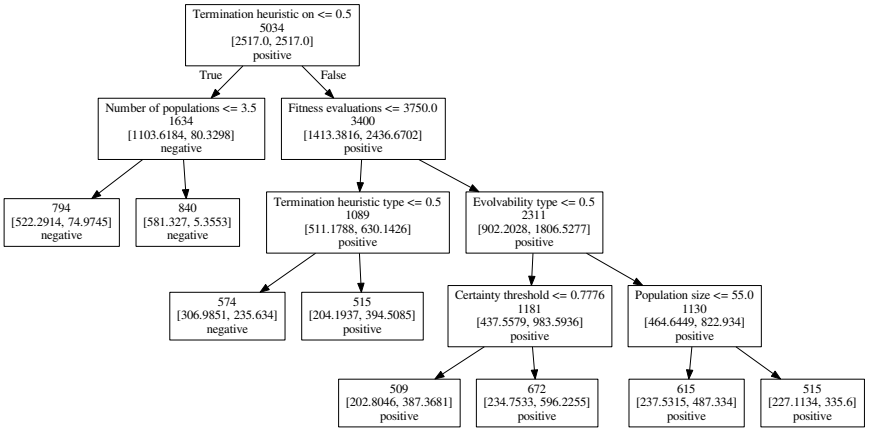

Figure B.17: SM, particle filter, eventual fitness

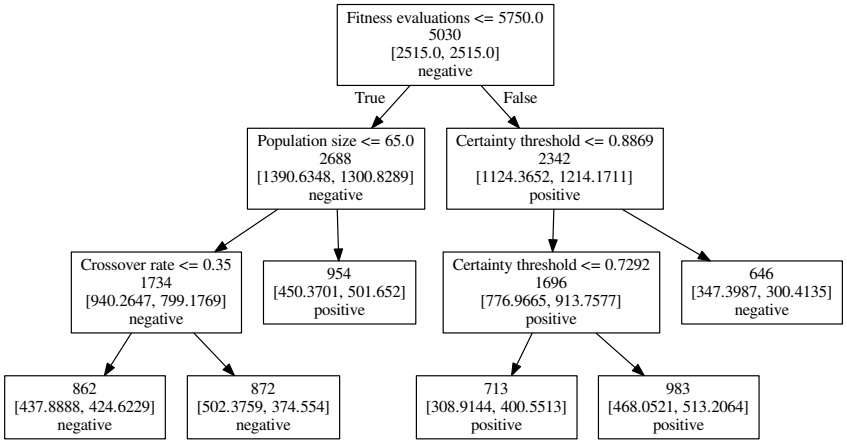

Figure B.18: SM, particle filter, eventual evolvability

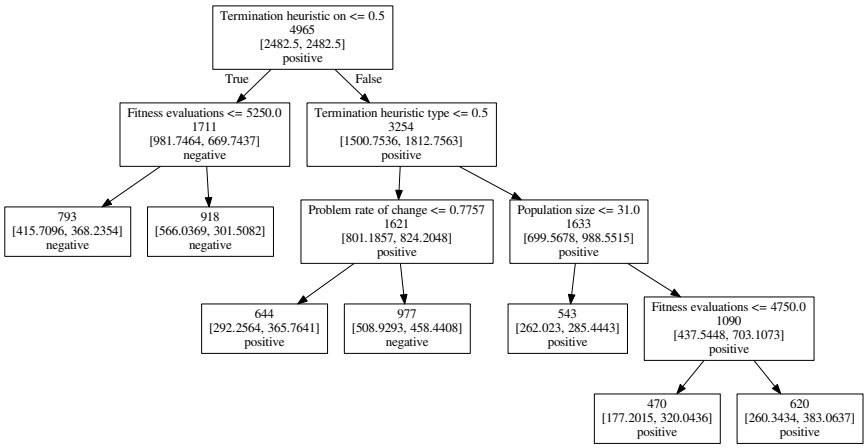

Figure B.19: MVPR, point estimate, eventual fitness

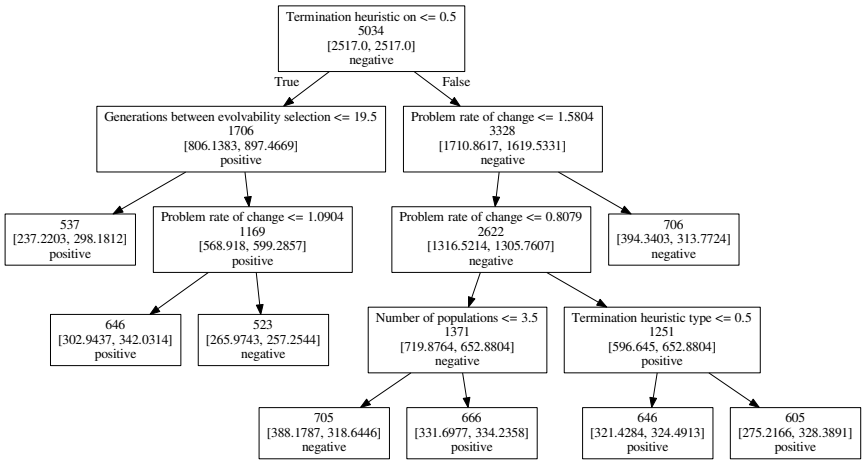

Figure B.20: MVPR, point estimate, eventual evolvability

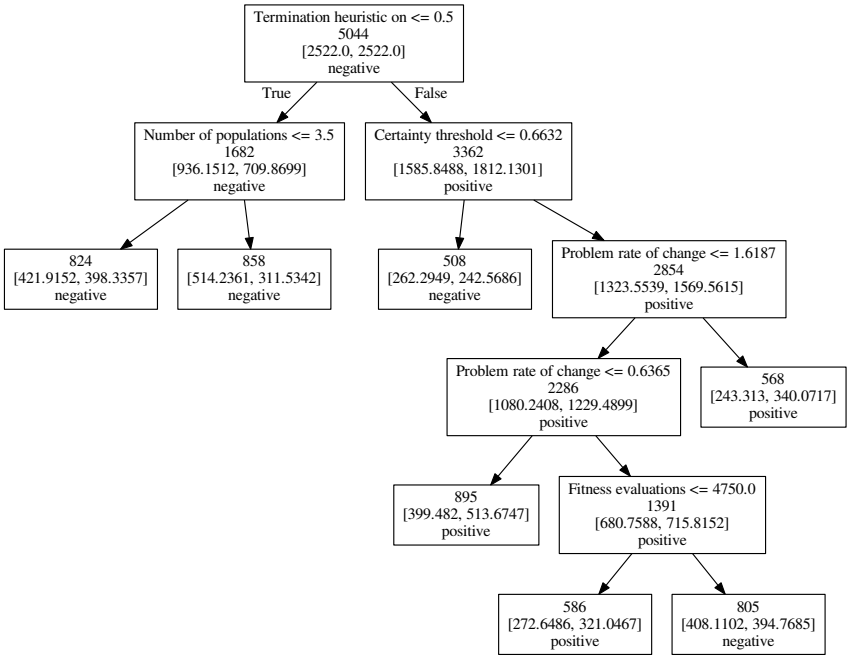

Figure B.21: MVPR, Kalman filter, eventual fitness

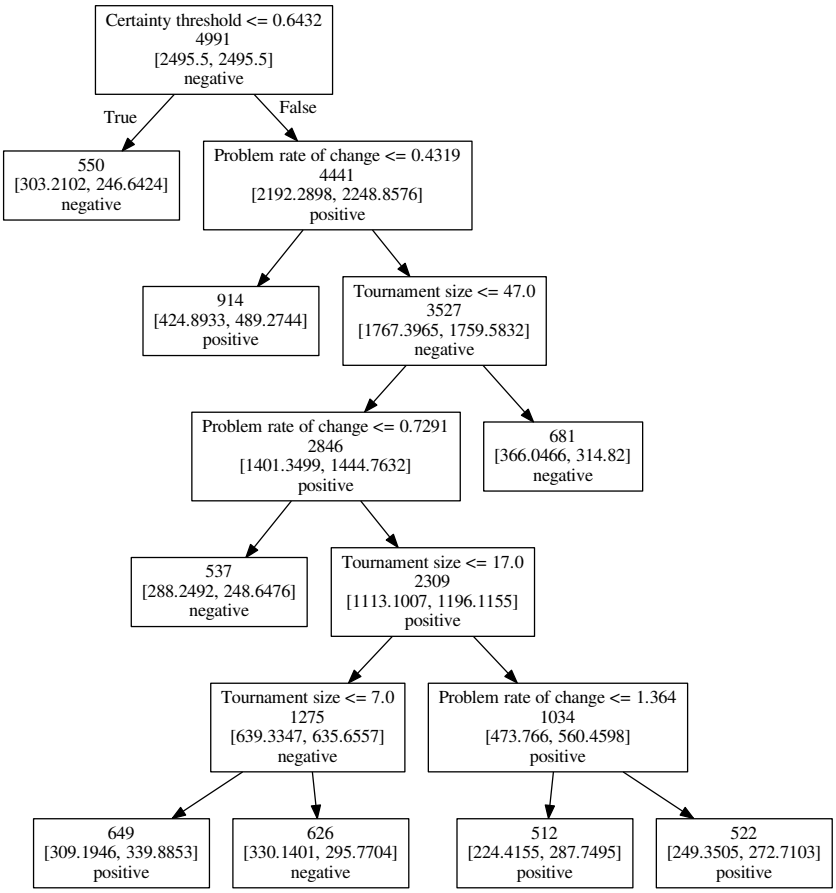

Figure B.22: MVPR, Kalman filter, eventual evolvability

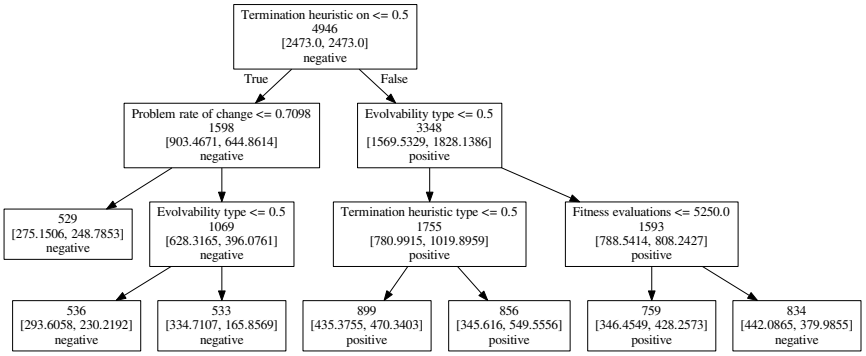

Figure B.23: MVPR, particle filter, eventual fitness

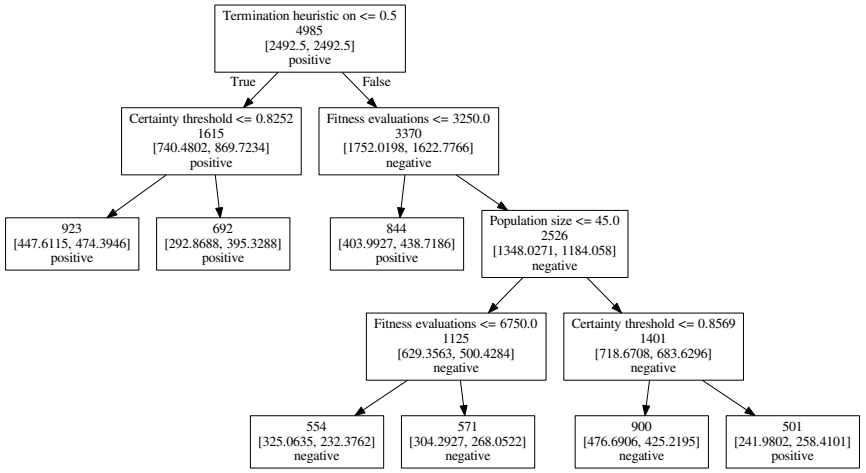

Figure B.24: MVPR, particle filter, eventual evolvability

Appendix C

EGS-AR Data

This chapter lists the results from the experiments described in Section 5.4. The results tables are organized by fitness function, evolvability estimation method, and whether the EGS-AR algorithm is being compared with selection for fitness alone or to the EGS algorithm. For each, there are two tables, one for relative eventual fitness (eventual fitness for EGS-AR minus eventual fitness when selecting for fitness alone, or for EGS), and one for relative eventual evolvability. The fields of the tables are the same as those in Appendix A, with the following additional fields in the tables comparing the EGS-AR algorithm with the EGS algorithm.

- **All data empirical median**: The empirical median relative eventual fitness or evolvability of the whole dataset. Values which lie outside of the inferred highest density interval (HDI) of the mean, presumably because the data distribution is not normal, are in italics.

- **All data mean HDI**: The highest density interval (95%) of the distribution representing our state of belief of the mean value of the whole dataset. This is the highest density interval in which we believe the

mean value falls with probability 0.95.

- **All data positive mean probability**: The probability representing our belief that the mean relative eventual fitness or evolvability is positive. High values mean that we believe that, averaging over the parameter distributions described in Section 4.7, the EGS-AR algorithm leads to a greater mean eventual fitness or evolvability than the EGS algorithm. Probabilities greater than 0.95 are in boldface.

Table C.1: SEM, Kalman filter, eventual fitness, compared to selection for fitness

Proportion positive	0.20
Accuracy, sensitivity, specificity	**0.61**, **0.82**, 0.56
Negative empirical median	*-1355.16*
Negative mean HDI	[-1150.90, -1004.82]
Positive empirical median	-457.84
Positive mean HDI	[-597.47, -453.12]
Different means probability	**1.00**
Positive mean probability	0.00

Table C.2: SEM, Kalman filter, eventual fitness, compared to EGS

Proportion positive	0.59
All data empirical median	84.05
All data mean HDI	[-1009.12, 619.87]
All data positive mean probability	0.80
Accuracy, sensitivity, specificity	0.58, 0.56, 0.60
Negative empirical median	18.32
Negative mean HDI	[-16.85, 52.77]
Positive empirical median	*147.76*
Positive mean HDI	[112.18, 147.63]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table C.3: SEM, Kalman filter, eventual evolvability, compared to selection for fitness

Proportion positive	0.47
Accuracy, sensitivity, specificity	0.58, *0.45*, **0.69**
Negative empirical median	-30.72
Negative mean HDI	[-38.84, -24.02]
Positive empirical median	36.00
Positive mean HDI	[23.41, 45.84]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table C.4: SEM, Kalman filter, eventual evolvability, compared to EGS

Proportion positive	0.53
All data empirical median	12.47
All data mean HDI	[-193.05, 757.29]
All data positive mean probability	0.88
Accuracy, sensitivity, specificity	0.51, *0.43*, 0.59
Negative empirical median	10.22
Negative mean HDI	[2.14, 16.40]
Positive empirical median	16.19
Positive mean HDI	[11.25, 25.87]
Different means probability	0.91
Positive mean probability	**1.00**

Table C.5: SEM, particle filter, eventual fitness, compared to selection for fitness

Proportion positive	0.22
Accuracy, sensitivity, specificity	**0.66**, **0.76**, **0.63**
Negative empirical median	*-1205.16*
Negative mean HDI	[-1048.84, -933.03]
Positive empirical median	-292.13
Positive mean HDI	[-415.13, -290.51]
Different means probability	**1.00**
Positive mean probability	0.00

Table C.6: SEM, particle filter, eventual fitness, compared to EGS

Proportion positive	0.61
All data empirical median	106.64
All data mean HDI	[100.80, 536.82]
All data positive mean probability	**0.95**
Accuracy, sensitivity, specificity	0.59, 0.54, **0.66**
Negative empirical median	10.76
Negative mean HDI	[-14.93, 44.49]
Positive empirical median	176.15
Positive mean HDI	[128.22, 178.44]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table C.7: SEM, particle filter, eventual evolvability, compared to selection for fitness

Proportion positive	0.50
Accuracy, sensitivity, specificity	0.56, *0.48*, **0.64**
Negative empirical median	-22.10
Negative mean HDI	[-28.57, -14.10]
Positive empirical median	41.73
Positive mean HDI	[39.19, 57.08]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table C.8: SEM, particle filter, eventual evolvability, compared to EGS

Proportion positive	0.54
All data empirical median	12.67
All data mean HDI	[-3.22, 16.90]
All data positive mean probability	0.92
Accuracy, sensitivity, specificity	0.52, *0.46*, 0.60
Negative empirical median	5.46
Negative mean HDI	[-1.40, 11.53]
Positive empirical median	20.53
Positive mean HDI	[18.41, 29.76]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table C.9: MM, Kalman filter, eventual fitness, compared to selection for fitness

Proportion positive	0.27
Accuracy, sensitivity, specificity	**0.62**, **0.83**, 0.55
Negative empirical median	-7.31
Negative mean HDI	[-7.66, -7.18]
Positive empirical median	-1.28
Positive mean HDI	[-1.39, -1.03]
Different means probability	**1.00**
Positive mean probability	0.00

Table C.10: MM, Kalman filter, eventual fitness, compared to EGS

Proportion positive	0.51
All data empirical median	0.06
All data mean HDI	[0.03, 3.18]
All data positive mean probability	**0.99**
Accuracy, sensitivity, specificity	*0.49*, *0.39*, **0.60**
Negative empirical median	0.09
Negative mean HDI	[-0.04, 0.31]
Positive empirical median	-0.03
Positive mean HDI	[-0.26, 0.12]
Different means probability	0.07
Positive mean probability	0.24

Table C.11: MM, Kalman filter, eventual evolvability, compared to selection for fitness

Proportion positive	0.56
Accuracy, sensitivity, specificity	0.55, 0.53, 0.59
Negative empirical median	0.03
Negative mean HDI	[-0.03, 0.10]
Positive empirical median	0.47
Positive mean HDI	[0.43, 0.57]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table C.12: MM, Kalman filter, eventual evolvability, compared to EGS

Proportion positive	0.50
All data empirical median	0.00
All data mean HDI	[-0.52, 0.02]
All data positive mean probability	0.70
Accuracy, sensitivity, specificity	*0.49*, 0.52, *0.46*
Negative empirical median	0.07
Negative mean HDI	[-0.03, 0.10]
Positive empirical median	-0.00
Positive mean HDI	[-0.07, 0.06]
Different means probability	0.31
Positive mean probability	0.64

Table C.13: MM, particle filter, eventual fitness, compared to selection for fitness

Proportion positive	0.30
Accuracy, sensitivity, specificity	0.57, **0.68**, 0.52
Negative empirical median	*-5.10*
Negative mean HDI	[-5.73, -5.21]
Positive empirical median	-1.57
Positive mean HDI	[-1.87, -1.48]
Different means probability	**1.00**
Positive mean probability	0.00

Table C.14: MM, particle filter, eventual fitness, compared to EGS

Proportion positive	0.54
All data empirical median	**0.49**
All data mean HDI	[0.58, 0.83]
All data positive mean probability	**0.97**
Accuracy, sensitivity, specificity	0.58, *0.25*, **0.94**
Negative empirical median	-0.31
Negative mean HDI	[-0.37, -0.09]
Positive empirical median	6.17
Positive mean HDI	[5.73, 6.64]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table C.15: MM, particle filter, eventual evolvability, compared to selection for fitness

Proportion positive	0.59
Accuracy, sensitivity, specificity	0.57, *0.42*, **0.78**
Negative empirical median	0.07
Negative mean HDI	[-0.00, 0.11]
Positive empirical median	*1.35*
Positive mean HDI	[1.50, 1.76]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table C.16: MM, particle filter, eventual evolvability, compared to EGS

Proportion positive	0.54
All data empirical median	0.19
All data mean HDI	[-2.41, 1.27]
All data positive mean probability	0.87
Accuracy, sensitivity, specificity	0.54, *0.35*, **0.76**
Negative empirical median	0.03
Negative mean HDI	[-0.03, 0.09]
Positive empirical median	*0.83*
Positive mean HDI	[0.92, 1.21]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table C.17: SM, Kalman filter, eventual fitness, compared to selection for fitness

Proportion positive	0.29
Accuracy, sensitivity, specificity	**0.65**, **0.83**, 0.57
Negative empirical median	*-9.35*
Negative mean HDI	[-10.07, -9.39]
Positive empirical median	*-0.52*
Positive mean HDI	[-1.01, -0.65]
Different means probability	**1.00**
Positive mean probability	0.00

Table C.18: SM, Kalman filter, eventual fitness, compared to EGS

Proportion positive	0.51
All data empirical median	0.10
All data mean HDI	[0.08, 0.32]
All data positive mean probability	**1.00**
Accuracy, sensitivity, specificity	0.52, 0.54, *0.50*
Negative empirical median	-0.02
Negative mean HDI	[-0.17, 0.21]
Positive empirical median	0.27
Positive mean HDI	[0.18, 0.53]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table C.19: SM, Kalman filter, eventual evolvability, compared to selection for fitness

Proportion positive	0.51
Accuracy, sensitivity, specificity	*0.50*, 0.60, *0.39*
Negative empirical median	0.14
Negative mean HDI	[-0.01, 0.20]
Positive empirical median	0.09
Positive mean HDI	[0.05, 0.25]
Different means probability	0.69
Positive mean probability	**1.00**

Table C.20: SM, Kalman filter, eventual evolvability, compared to EGS

Proportion positive	0.51
All data empirical median	0.04
All data mean HDI	[-0.01, 0.19]
All data positive mean probability	0.90
Accuracy, sensitivity, specificity	0.51, *0.45*, 0.58
Negative empirical median	-0.10
Negative mean HDI	[-0.19, -0.01]
Positive empirical median	0.02
Positive mean HDI	[-0.12, 0.07]
Different means probability	0.91
Positive mean probability	0.31

Table C.21: SM, particle filter, eventual fitness, compared to selection for fitness

Proportion positive	0.33
Accuracy, sensitivity, specificity	0.55, **0.89**, *0.38*
Negative empirical median	-9.22
Negative mean HDI	[-9.80, -8.99]
Positive empirical median	*-0.76*
Positive mean HDI	[-1.17, -0.87]
Different means probability	**1.00**
Positive mean probability	0.00

Table C.22: SM, particle filter, eventual fitness, compared to EGS

Proportion positive	0.55
All data empirical median	**0.54**
All data mean HDI	[0.66, 1.33]
All data positive mean probability	**1.00**
Accuracy, sensitivity, specificity	0.58, *0.27*, **0.96**
Negative empirical median	-0.20
Negative mean HDI	[-0.33, -0.06]
Positive empirical median	10.35
Positive mean HDI	[9.49, 10.59]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table C.23: SM, particle filter, eventual evolvability, compared to selection for fitness

Proportion positive	0.52
Accuracy, sensitivity, specificity	0.52, *0.46*, 0.58
Negative empirical median	0.06
Negative mean HDI	[-0.03, 0.15]
Positive empirical median	0.30
Positive mean HDI	[0.26, 0.47]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table C.24: SM, particle filter, eventual evolvability, compared to EGS

Proportion positive	0.51
All data empirical median	0.07
All data mean HDI	[0.02, 0.11]
All data positive mean probability	**0.98**
Accuracy, sensitivity, specificity	0.52, *0.50*, 0.55
Negative empirical median	-0.04
Negative mean HDI	[-0.12, 0.06]
Positive empirical median	0.23
Positive mean HDI	[0.12, 0.33]
Different means probability	**1.00**
Positive mean probability	**1.00**

Table C.25: MVPR, Kalman filter, eventual fitness, compared to selection for fitness

Proportion positive	0.42
Accuracy, sensitivity, specificity	0.50, **0.62**, *0.41*
Negative empirical median	-2.09
Negative mean HDI	[-2.36, -1.73]
Positive empirical median	-1.30
Positive mean HDI	[-1.47, -0.87]
Different means probability	**1.00**
Positive mean probability	0.00

Table C.26: MVPR, Kalman filter, eventual fitness, compared to EGS

Proportion positive	0.50
All data empirical median	-0.07
All data mean HDI	[-1.60, 1.82]
All data positive mean probability	0.31
Accuracy, sensitivity, specificity	0.50, *0.50*, 0.50
Negative empirical median	-0.11
Negative mean HDI	[-0.62, -0.04]
Positive empirical median	-0.16
Positive mean HDI	[-0.50, 0.22]
Different means probability	0.85
Positive mean probability	0.36

Table C.27: MVPR, Kalman filter, eventual evolvability, compared to selection for fitness

Proportion positive	0.51
Accuracy, sensitivity, specificity	*0.49*, *0.35*, **0.65**
Negative empirical median	0.45
Negative mean HDI	[-0.01, 1.53]
Positive empirical median	0.52
Positive mean HDI	[-0.04, 1.67]
Different means probability	0.63
Positive mean probability	**0.95**

Table C.28: MVPR, Kalman filter, eventual evolvability, compared to EGS

Proportion positive	0.51
All data empirical median	0.28
All data mean HDI	[-3.87, 0.96]
All data positive mean probability	0.95
Accuracy, sensitivity, specificity	*0.50*, 0.52, *0.48*
Negative empirical median	0.38
Negative mean HDI	[-0.01, 1.47]
Positive empirical median	0.28
Positive mean HDI	[0.06, 1.41]
Different means probability	0.49
Positive mean probability	**0.98**

Table C.29: MVPR, particle filter, eventual fitness, compared to selection for fitness

Proportion positive	0.42
Accuracy, sensitivity, specificity	0.54, *0.49*, 0.58
Negative empirical median	-2.16
Negative mean HDI	[-2.53, -1.91]
Positive empirical median	-0.95
Positive mean HDI	[-1.30, -0.62]
Different means probability	**1.00**
Positive mean probability	0.00

Table C.30: MVPR, particle filter, eventual fitness, compared to EGS

Proportion positive	0.51
All data empirical median	0.18
All data mean HDI	[-0.01, 0.42]
All data positive mean probability	**1.00**
Accuracy, sensitivity, specificity	0.51, 0.56, *0.46*
Negative empirical median	-0.05
Negative mean HDI	[-0.28, 0.40]
Positive empirical median	0.29
Positive mean HDI	[0.02, 0.60]
Different means probability	0.84
Positive mean probability	**0.98**

Table C.31: MVPR, particle filter, eventual evolvability, compared to selection for fitness

Proportion positive	0.51
Accuracy, sensitivity, specificity	0.51, 0.51, *0.50*
Negative empirical median	0.41
Negative mean HDI	[-0.10, 1.37]
Positive empirical median	0.86
Positive mean HDI	[0.25, 1.73]
Different means probability	0.86
Positive mean probability	**1.00**

Table C.32: MVPR, particle filter, eventual evolvability, compared to EGS

Proportion positive	0.49
All data empirical median	-0.25
All data mean HDI	[-0.75, 0.37]
All data positive mean probability	0.26
Accuracy, sensitivity, specificity	*0.50*, *0.39*, **0.60**
Negative empirical median	-0.05
Negative mean HDI	[-0.72, 0.55]
Positive empirical median	-0.40
Positive mean HDI	[-0.95, 0.62]
Different means probability	0.44
Positive mean probability	0.35

Appendix D

EGS-AR Decision Trees

This chapter provides the decision trees that are used as part of the data analysis pipeline for the experiments described in Section 5.4. The purpose of these trees is to identify regions of the parameter space in which the EGS-AR algorithm leads to greater mean eventual fitness or evolvability than selection for fitness alone or than the EGS algorithm. The feature labels for the termination heuristic in the trees have the following meaning. 'Termination heuristic on' is a feature whose value is one if a termination heuristic is used and zero otherwise. 'Termination heuristic type' takes value zero for the 'halfway' termination heuristic and the value one for the 'relative' termination heuristic.

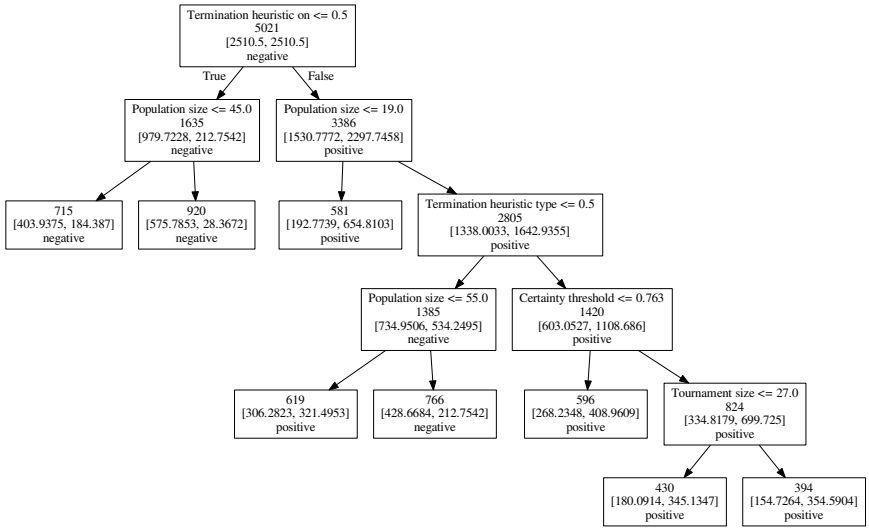

Figure D.1: SEM, Kalman filter, eventual fitness, compared to selection for fitness

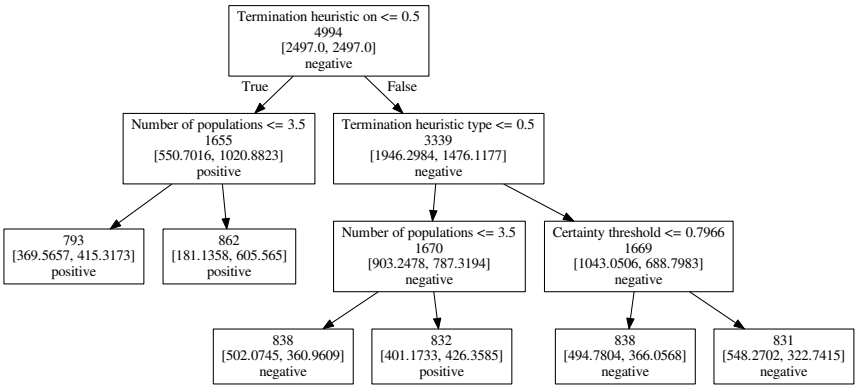

Figure D.2: SEM, Kalman filter, eventual fitness, compared to EGS

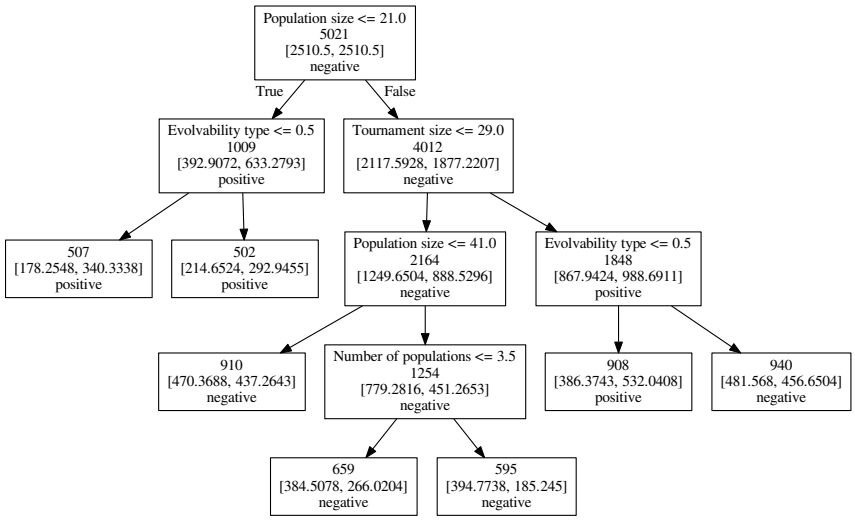

Figure D.3: SEM, Kalman filter, eventual evolvability, compared to selection for fitness

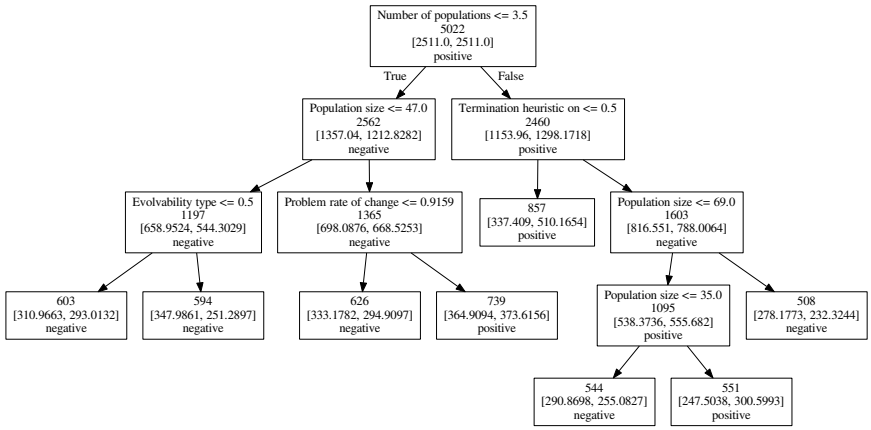

Figure D.4: SEM, Kalman filter, eventual evolvability, compared to EGS

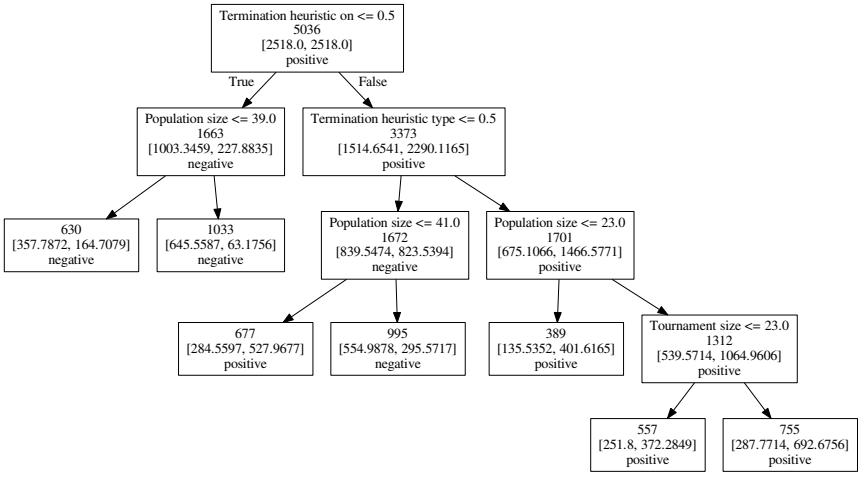

Figure D.5: SEM, particle filter, eventual fitness, compared to selection for fitness

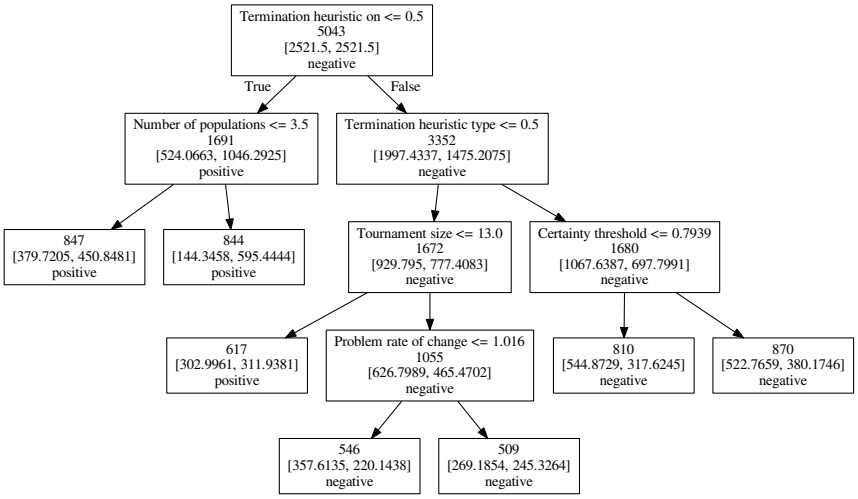

Figure D.6: SEM, particle filter, eventual fitness, compared to EGS

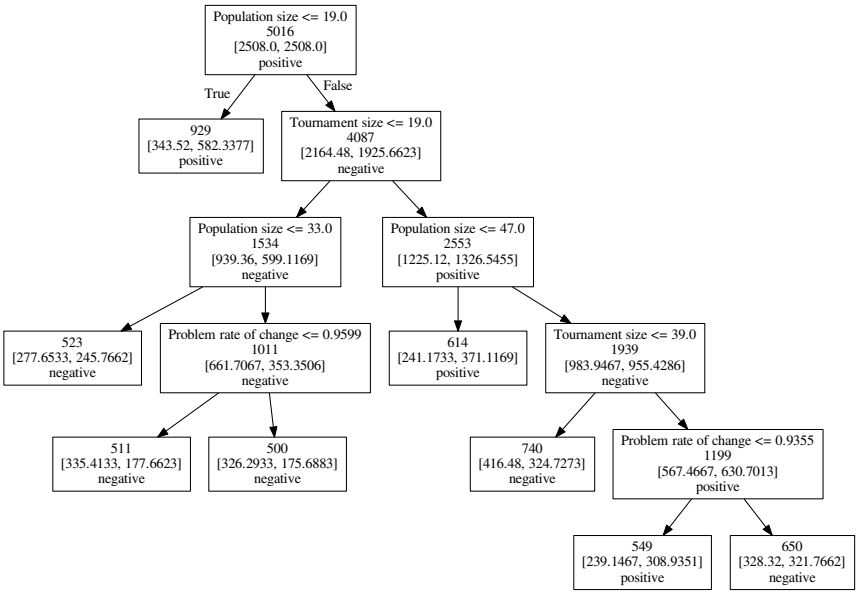

Figure D.7: SEM, particle filter, eventual evolvability, compared to selection for fitness

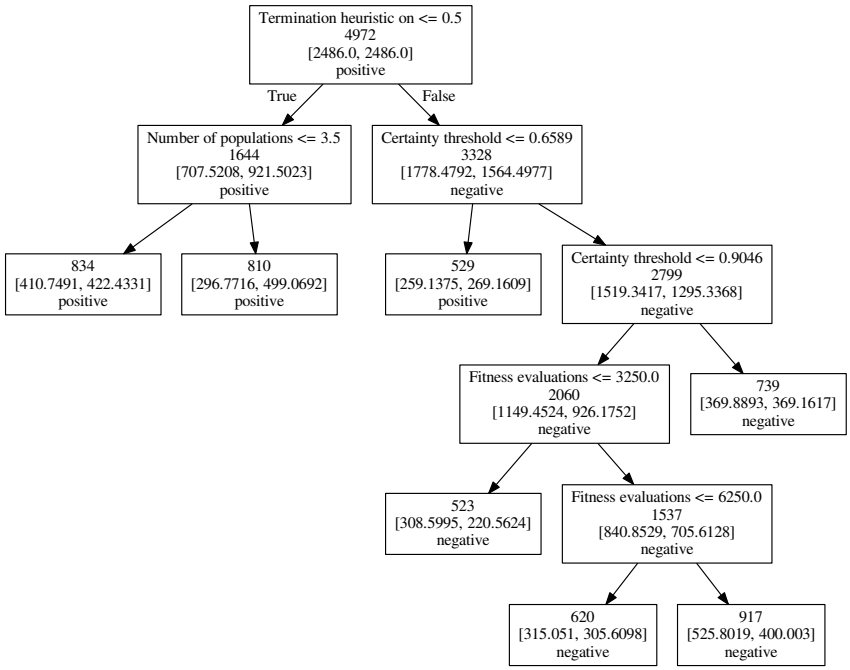

Figure D.8: SEM, particle filter, eventual evolvability, compared to EGS

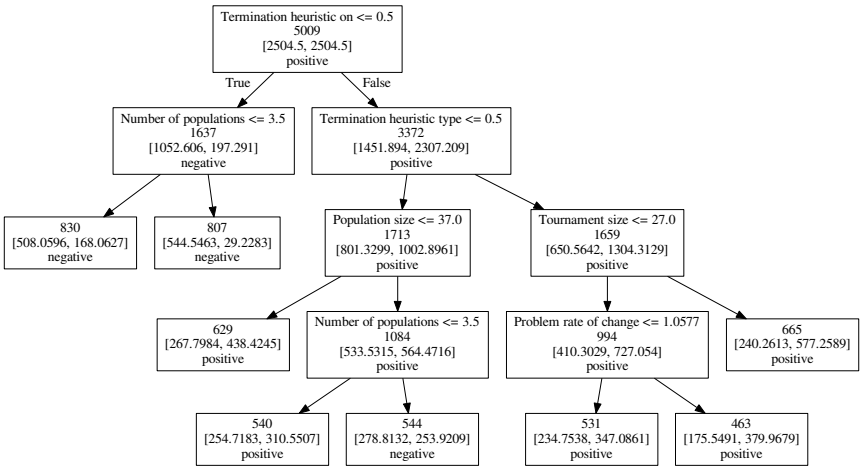

Figure D.9: MM, Kalman filter, eventual fitness, compared to selection for fitness

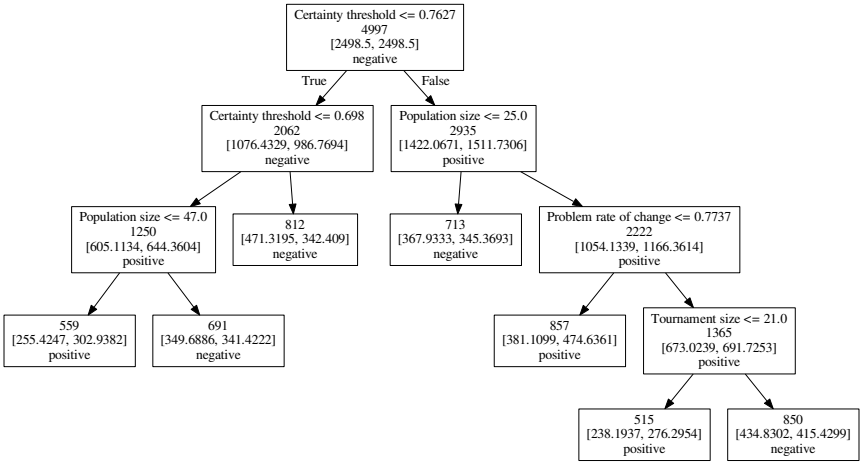

Figure D.10: MM, Kalman filter, eventual fitness, compared to EGS

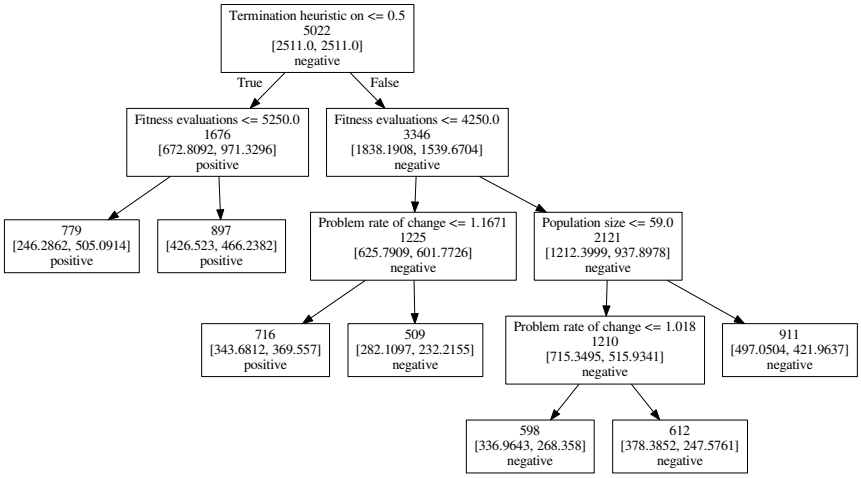

Figure D.11: MM, Kalman filter, eventual evolvability, compared to selection for fitness

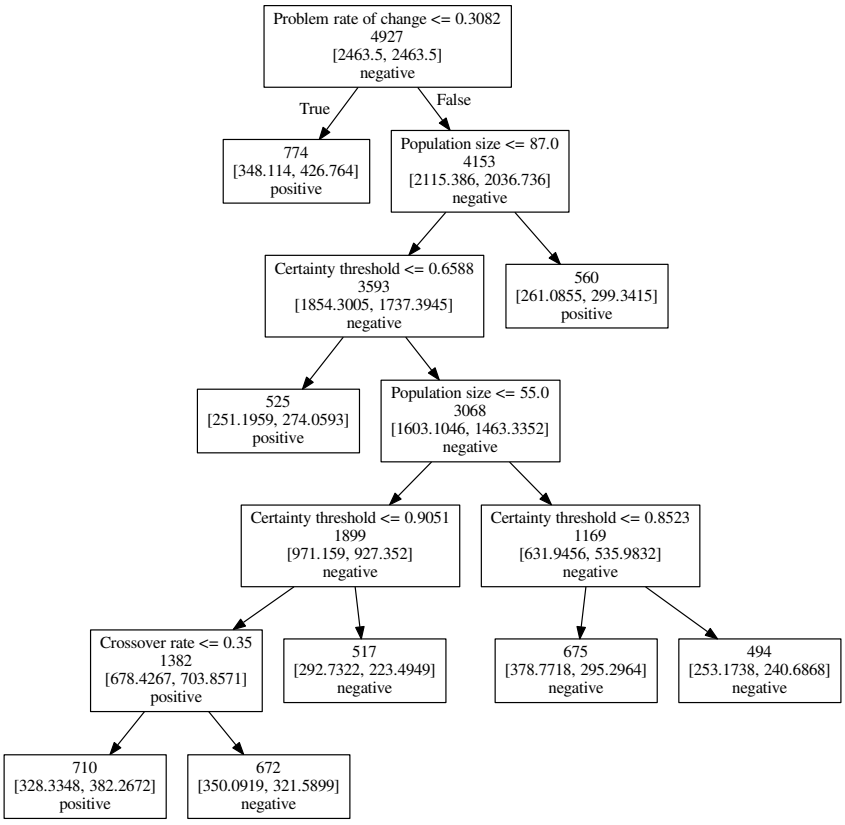

Figure D.12: MM, Kalman filter, eventual evolvability, compared to EGS

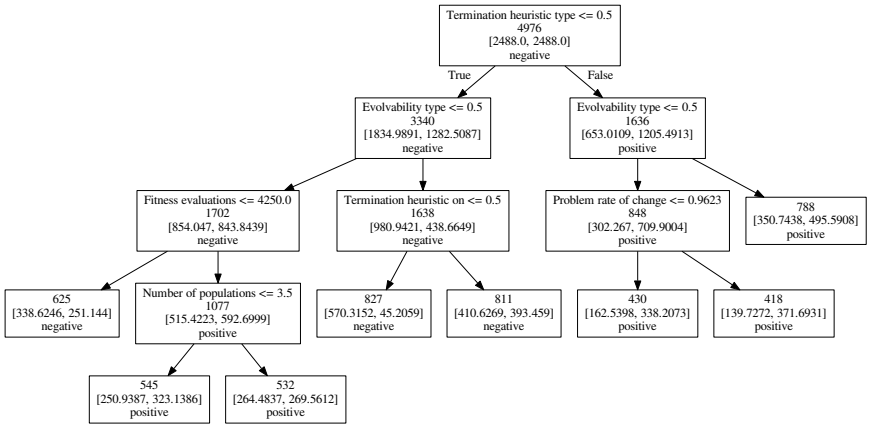

Figure D.13: MM, particle filter, eventual fitness, compared to selection for fitness

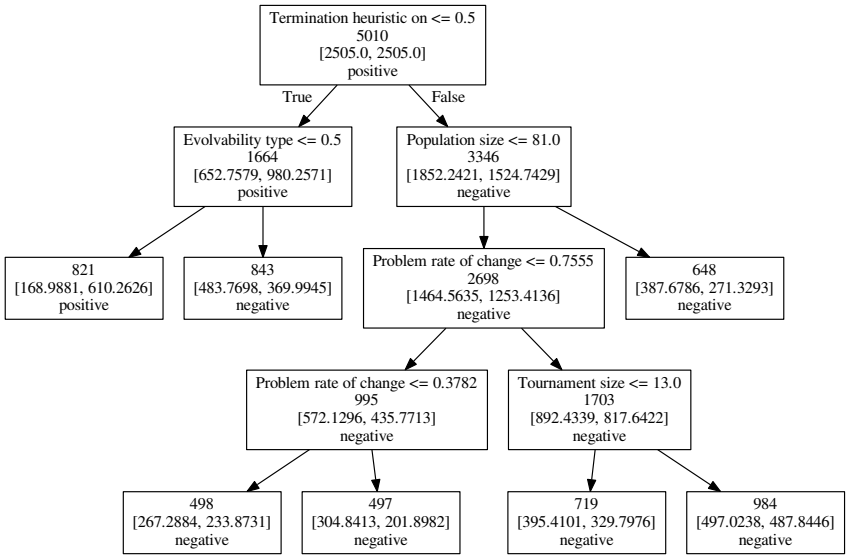

Figure D.14: MM, particle filter, eventual fitness, compared to EGS

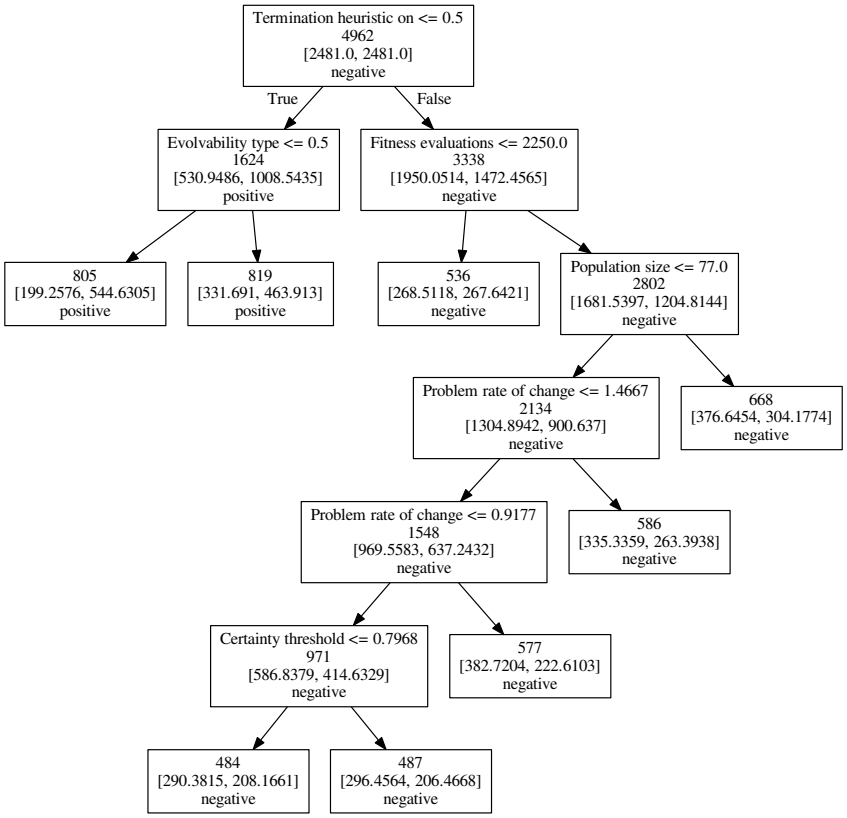

Figure D.15: MM, particle filter, eventual evolvability, compared to selection for fitness

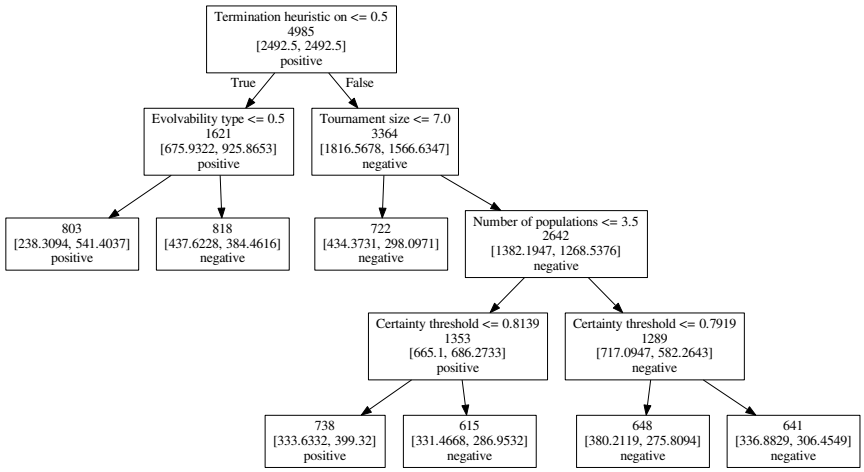

Figure D.16: MM, particle filter, eventual evolvability, compared to EGS

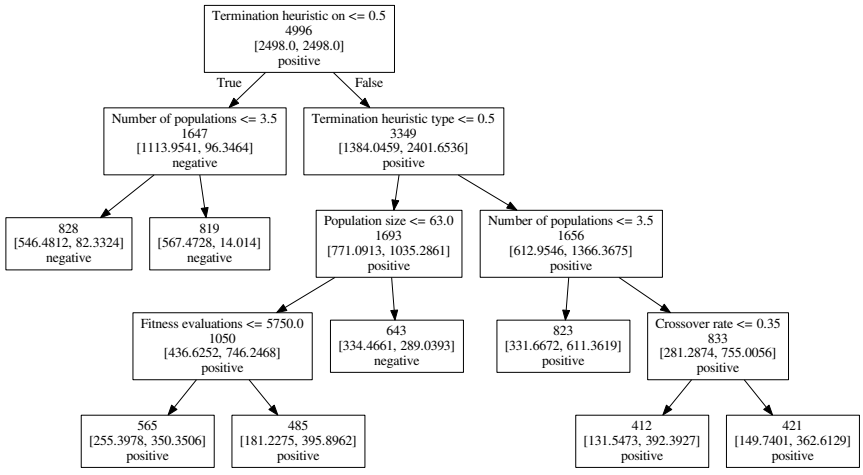

Figure D.17: SM, Kalman filter, eventual fitness, compared to selection for fitness

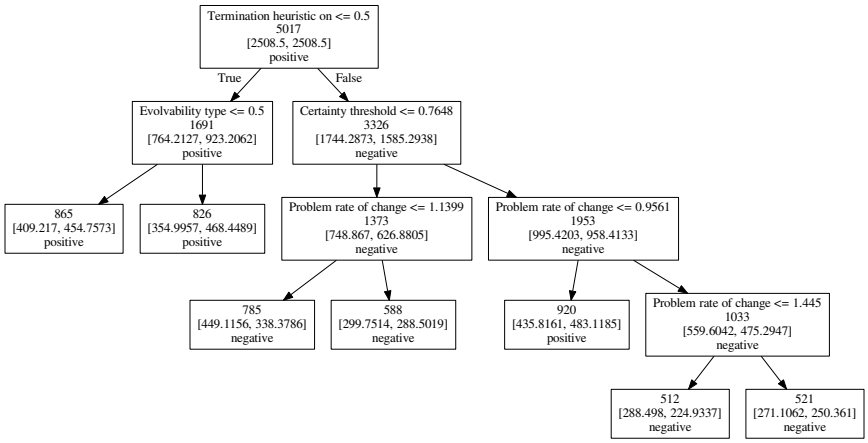

Figure D.18: SM, Kalman filter, eventual fitness, compared to EGS

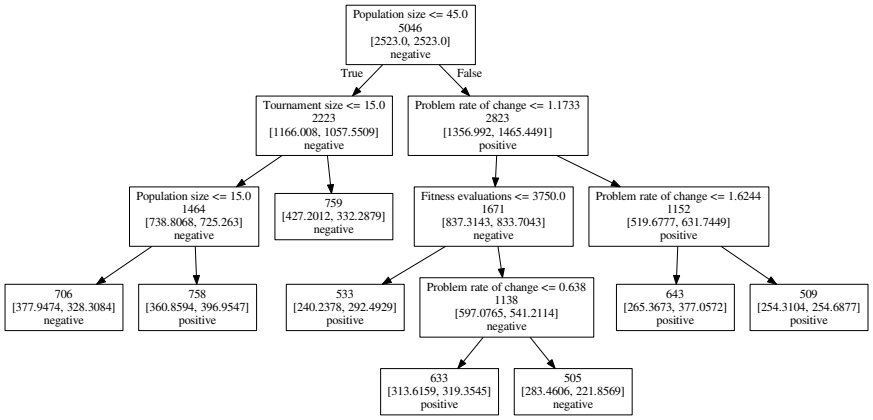

Figure D.19: SM, Kalman filter, eventual evolvability, compared to selection for fitness

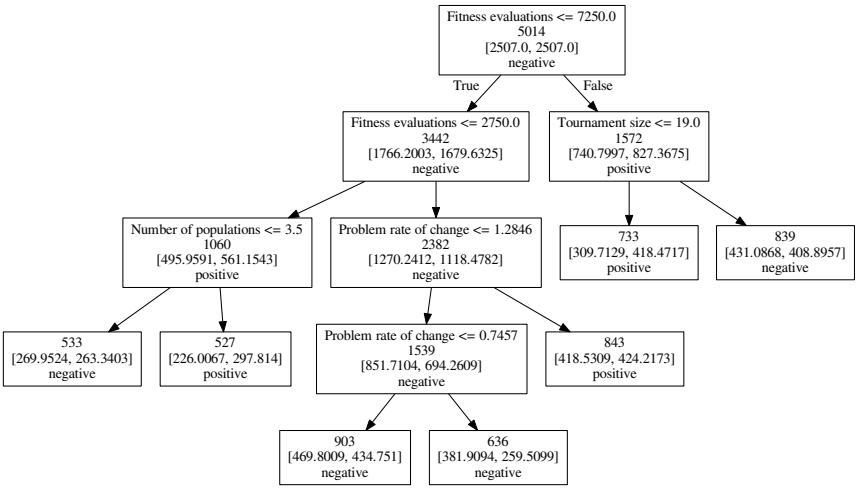

Figure D.20: SM, Kalman filter, eventual evolvability, compared to EGS

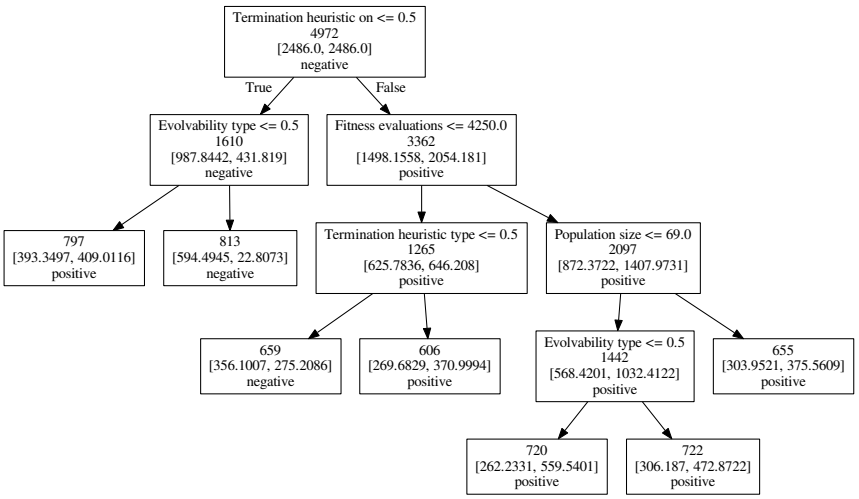

Figure D.21: SM, particle filter, eventual fitness, compared to selection for fitness

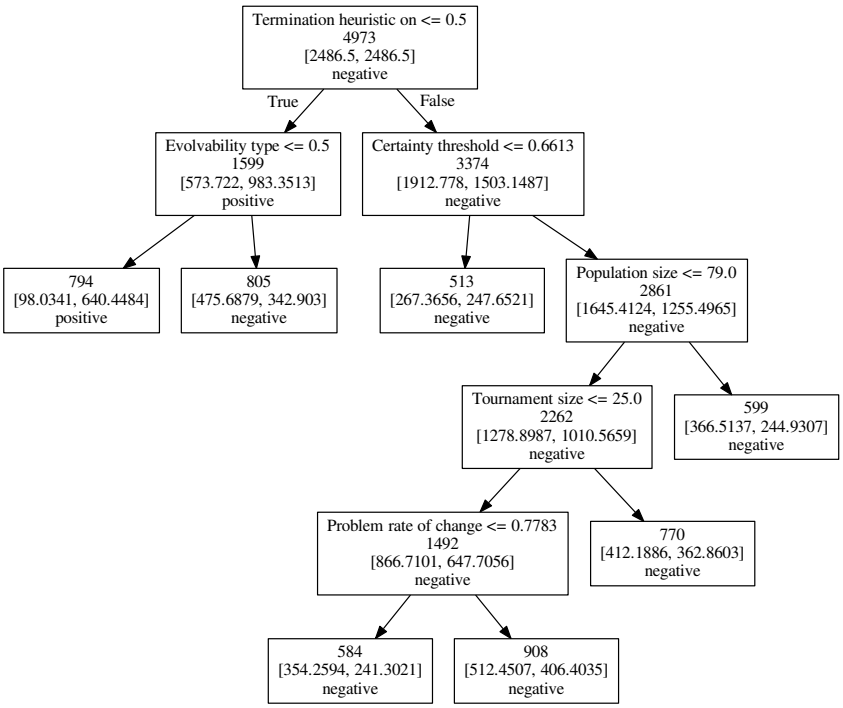

Figure D.22: SM, particle filter, eventual fitness, compared to EGS

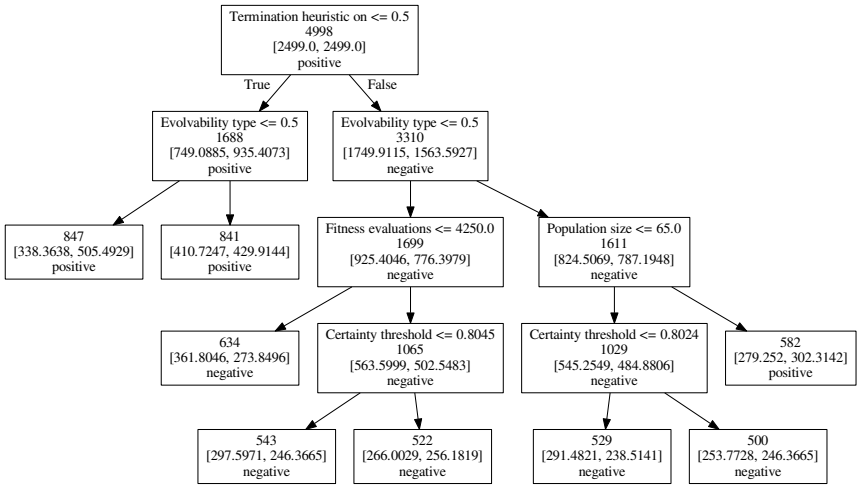

Figure D.23: SM, particle filter, eventual evolvability, compared to selection for fitness

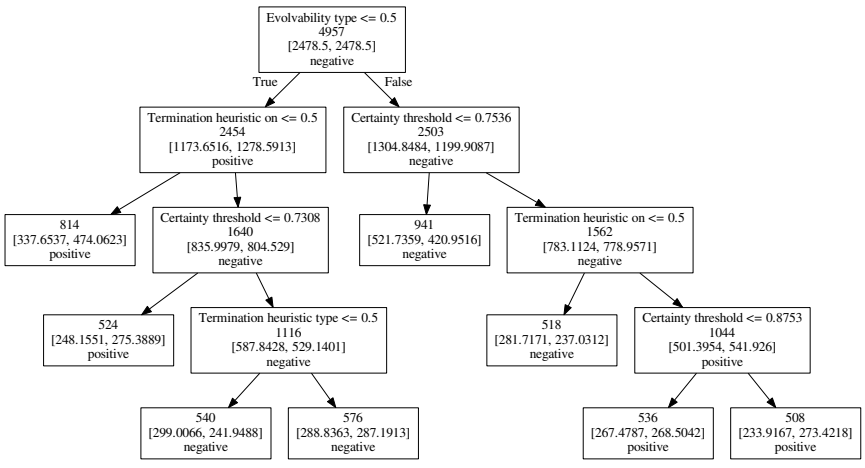

Figure D.24: SM, particle filter, eventual evolvability, compared to EGS

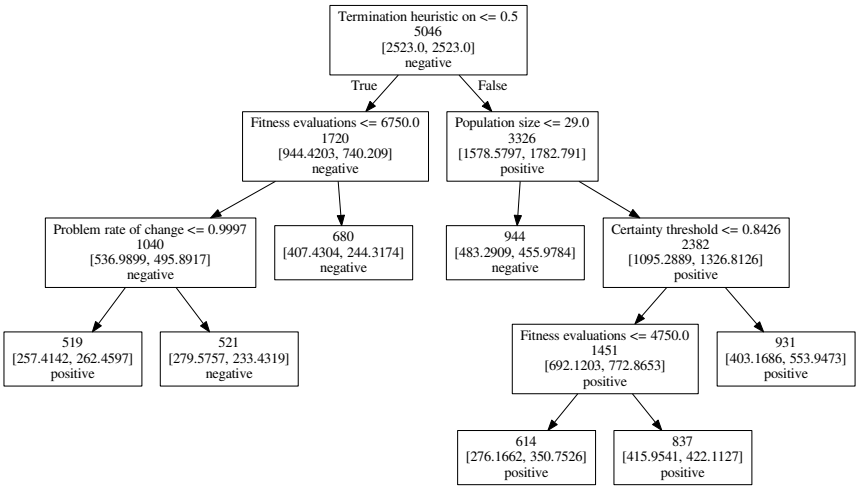

Figure D.25: MVPR, Kalman filter, eventual fitness, compared to selection for fitness

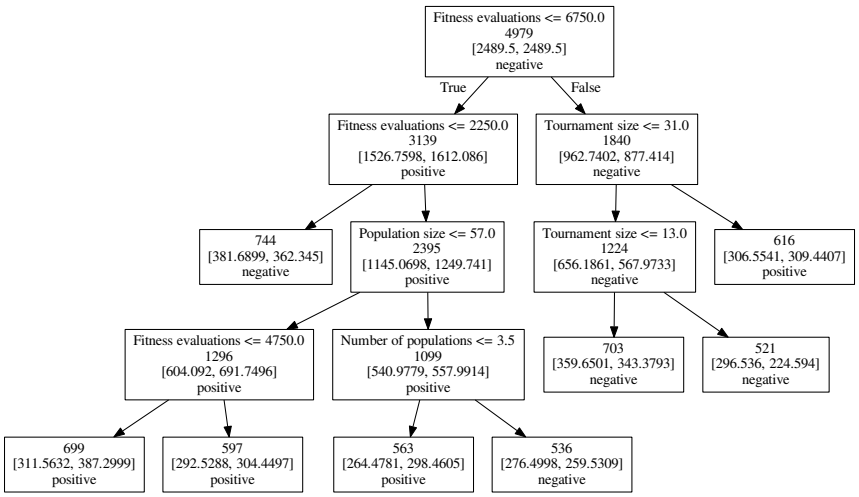

Figure D.26: MVPR, Kalman filter, eventual fitness, compared to EGS

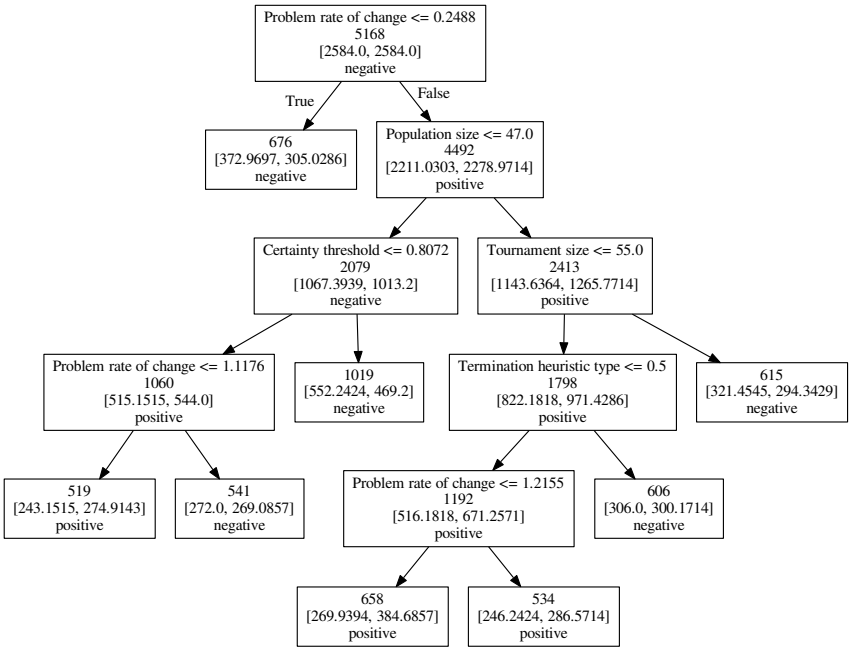

Figure D.27: MVPR, Kalman filter, eventual evolvability, compared to selection for fitness

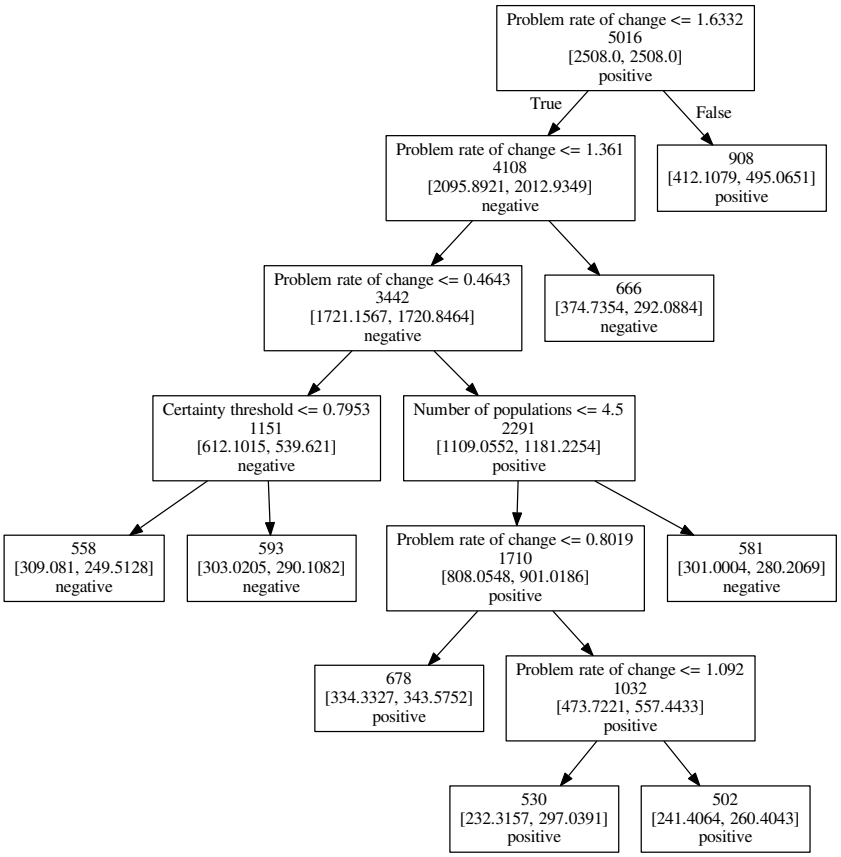

Figure D.28: MVPR, Kalman filter, eventual evolvability, compared to EGS

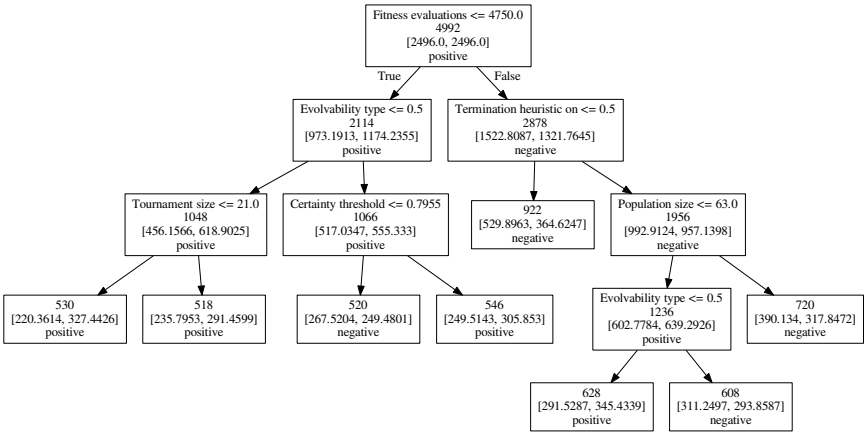

Figure D.29: MVPR, particle filter, eventual fitness, compared to selection for fitness

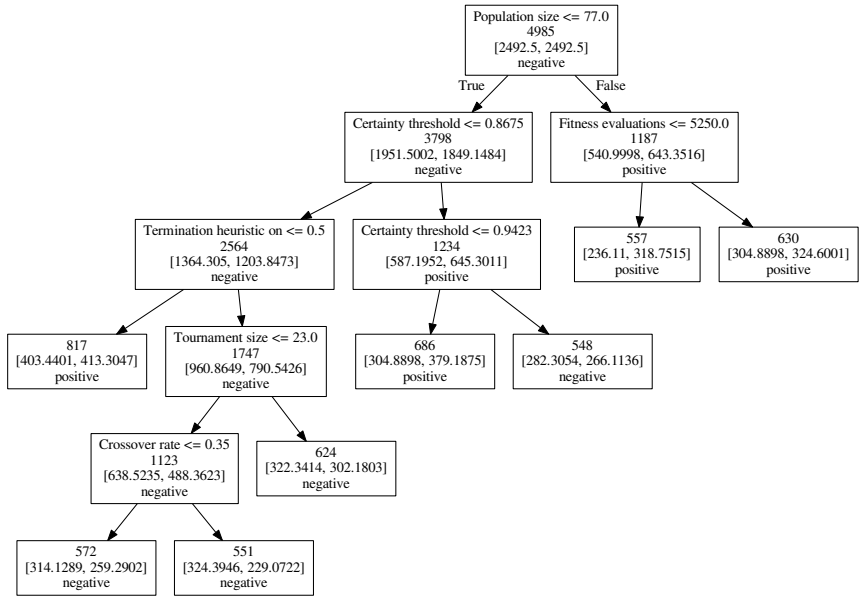

Figure D.30: MVPR, particle filter, eventual fitness, compared to EGS

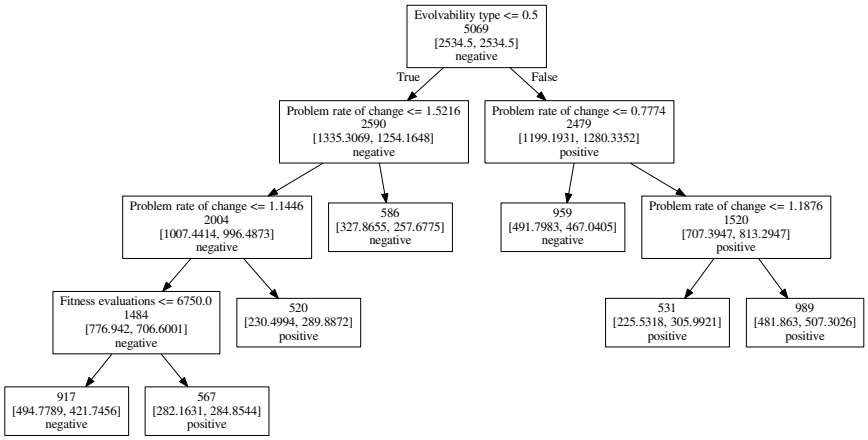

Figure D.31: MVPR, particle filter, eventual evolvability, compared to selection for fitness

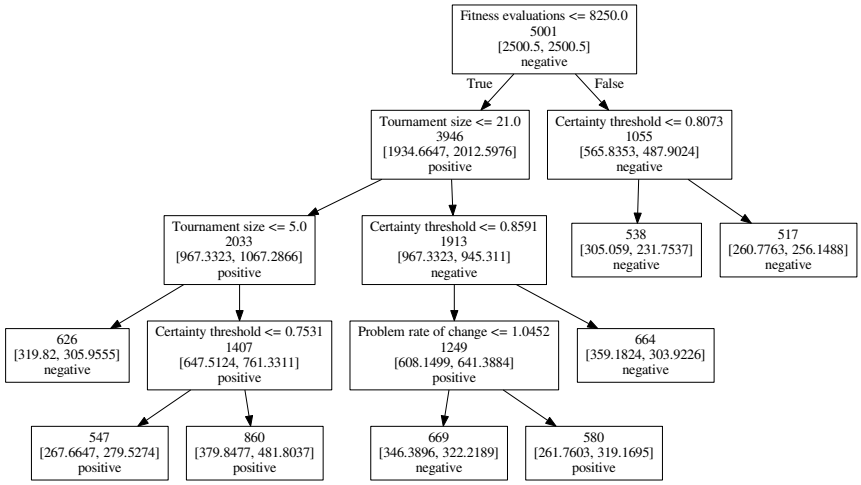

Figure D.32: MVPR, particle filter, eventual evolvability, compared to EGS