University of Sussex

Separating what is Evaluated from what is Selected in Artificial Evolution

Nicholas Tomko

Submitted for the degree of Doctor of Philosophy

University of Sussex

Oct 2013

Declaration

I hereby declare that this thesis has not been and will not be submitted in whole or in part to another University for the award of any other degree.

Signature:

Nicholas Tomko

UNIVERSITY OF SUSSEX

Nicholas Tomko, Doctor of Philosophy

Separating what is Evaluated from what is Selected in Artificial Evolution

Summary

In artificial evolution, selection and evaluation are separate and distinct steps. This distinction is rather different in natural evolution, where fitness (corresponding to evaluation) is a direct consequence of selection rather than a precursor to it. This thesis presents a new way of thinking about artificial evolution that separates evaluation and selection and consequently opens up the space of potential evolutionary algorithms beyond the limitations imposed by ignoring this distinction.

In Part I of the thesis we explore how varying the level of evaluation and selection impacts evolution. Using novel genetic algorithms (GAs) we show how group level evaluation allows evolution to find solutions to problems that require niching or a division of labour amongst component parts, something that cannot be accomplished using a standard GA. One of the inspirations for testing GAs with group-level evaluation was recent research into bacterial evolution which shows in bacterial colonies, distinguishing between the individual and group is very difficult because of the symbiotic relationship between different bacteria. We find that depending on the task it sometimes makes sense to select the individual while in other cases simply selecting groups is the best choice. Finally, we present a method for evolving the group size in these types of GAs that has the benefit of avoiding the need to know the optimal division of labour ahead of time.

In Part II we move away from studying the relationship between evaluation and selection to show how our novel view of evolution can be used to develop GAs that implement horizontal gene transfer which was again inspired by looking at bacterial evolution. By testing these GAs on a variety of different tasks we show how this promiscuous gene swapping is often beneficial to evolution because it can reduce the probability of the population getting stuck on a sub-optimal solution.

The thesis demonstrates the benefits of of looking at artificial evolution in terms of both evaluation and selection when it comes to algorithm development, and thus provides the GA community with a new context in which they can choose different algorithms appropriate to different tasks.

Acknowledgements

There are three groups of people I'd like to thank for making my PhD both enjoyable and successful.

First, I'd like to thank both Andy and Inman for their supervision. When I'm asked what I think makes a great PhD I always say it comes down to having great supervisors. I was luckily enough to have two fantastic supervisors with whom I really enjoyed working with and who provided excellent support and guidance throughout my 4 years at Sussex.

One of the great things about doing a PhD in the CCNR was working with people with such a wide variety of interests and backgrounds. So thank-you to Al, Pete, Matt, Greg, Edgar, Bruno, Renan, Lucas, James, Paul, and Jose for making the CCNR such a fun place to do a PhD. A special thanks goes out to Nathaniel for his help and collaboration.

Last, but definitely not least, I'd like to thank my family. My wife Leith deserves a million thanks-yous for all the emotional support and sacrifices she made over the last 4 years. Without her, this PhD would not have been possible. Finally, thanks to my parents for everything they have done; especially my father for all his advice and proofreading.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Genetic algorithms (GAs, Holland, 1975) are a search and optimisation technique based on Darwinian evolution. They fall into the general field of evolutionary algorithms or artificial evolution. Most GAs start off with a random population of individuals that are evaluated on a specific task. Each individual in the population is assigned a fitness based on how well it solves the task and the fitter individuals are more likely to be selected to pass on their genes to the next generation, while the less fit are more likely to be killed off. As in natural evolution, the new population of offspring is generated by recombining (mating) the fitter individuals and then applying random mutation. Over time, the population guided by fitness based selection, mutation and recombination moves around the solution space looking for better solutions. GAs have been successfully applied to a wide variety of different problems. For all of these tasks the goal of the GA is the same, which is to evolve fit individuals based on the task-specific fitness function. It is important to note that for the purposes of this thesis we define 'artificial evolution' as evolutionary experiments or algorithms where there is an explicitly defined fitness or cost function that is used to evaluate entities in the population.

Even though GAs are based on the principles of natural evolution, there are some important differences between natural and artificial evolution that need to be understood in order to develop novel GAs. One of these differences is that in artificial evolution, evaluation and selection can be easily separated, which is not the case in natural evolution where the fitness of a given organism, as viewed by an outside observer, is a direct consequence of selection rather than a precursor to it. In other words, high-fitness adaptations can only be labeled as high-fitness after they have been selected for. This is in contrast to artificial evolution where most GAs have an explicit fitness function that is pre-set by the experimenter before evolution begins. It is this fitness function that is used to evalu-

ate individuals and assign them a fitness and so determines which individuals should be selected to pass their genes on to the next generation. In natural evolution there is no explicit fitness function. The relative fitness of different biological organisms is something that can only be determined by an observer after selection has occurred.

The result of thinking about GAs too much in terms of natural evolution means that in many cases new GAs are developed by thinking about different ways of implementing selection, while forgetting about the evaluation step. One reason for this could be the on-going debate in natural evolution over whether group selection actually occurs, which focuses on the selection step of evolution. One of the main goals of this thesis is to overcome this way of thinking and show how ignoring the difference between evaluation and selection limits the types of GAs that can be developed. As we show, understanding that both what is being evaluated and what is being selected can be varied can result in novel algorithms with interesting properties.

This thesis argues for the importance of thinking about artificial evolution in terms of both what is evaluated and what is selected, in that a key consequence of this is the extension of the space of potential GAs. Using this view of evolution we come up with an evolutionary cycle that can be used to both analyse existing GAs and develop new GAs. After presenting this new view of evolution in detail the remainder of this thesis is spent exploring features of the novel GAs thus developed.

This work should be helpful to researchers interested in the theory of GAs as well as people who are looking to apply GAs to specific tasks. One of the foci of our research is group evolution, where groups of individuals are evaluated on a given task. We find that group evolution can be used to evolve populations where there is a division of labour between component parts. Therefore, our work should be applicable to people looking to solve tasks that would benefit from the evolution of cooperation or symbiosis between individuals in the population. Even though the focus of this research is artificial evolution, the insights may also be of relevance to group selection in nature.

1.1 Major Contributions

This thesis provides three major contributions to the field of artificial evolution, the first being:

1. A novel way of looking at artificial evolution that differentiates between what is being evaluated and what is being selected, the benefits of which include:

(a) It opens up the space of evolutionary algorithms by explicitly pointing out that in artificial evolution both the level of evaluation and level of selection can be varied.

(b) It provides a structured way to analyse existing algorithms and provides a toolbox for coming up with interesting ways to modify these algorithms.

The second and third major contributions are two novel families of GAs that were developed using this new way of thinking about artificial evolution. Both families of GAs were also inspired by some of the differences bacterial and eukaryotic evolution. First we explore symbiotic type GAs by varying what is being evaluated and what is being selected and then we implement bacterial-like horizontal gene transfer in GAs. Below, the benefits of each of these families of GAs are summarised.

2. Novel GAs that vary the relationship between what is being evaluated and what is being selected which are used to:

   (a) Show the benefits of symbiotic evolution where evaluation is occurring at the group level.

   (b) Demonstrate that on some tasks, such as certain artificial neural network tasks, group evaluation with individual selection makes sense but on other types of tasks, such as artificial immune system tasks, group evaluation with group selection is better.

   (c) Develop a method of evolving group size in symbiotic GAs that eliminates the need to pre-set the number of niches or know the optimal division of labour ahead of time. This method can be used on any GA that implements group evaluation.

3. Novel GAs that implement horizontal gene transfer which are used to:

   (a) Explore the effects of horizontal gene transfer on landscapes with different amounts of ruggedness, neutrality and epistasis.

   (b) Determine that on landscapes with a lot of local optima implementing massive amounts of horizontal gene transfer improves performance over a standard GA. However as neutrality is increased these advantages can disappear.

1.1.1 Claims this thesis does not make

To avoid confusion it is important to explicitly point out what this thesis does not do. We do not make the claim that any of the new GAs are the 'best' way to solve any specific task. This is related to the no free lunch theorem which says that for any search/optimization algorithm, any increased performance on one class of problems is paid for in reduced performance on another class of problems (Wolpert and Macready, 1997). Instead we have selected tasks to demonstrate interesting qualities of our GAs and show the potential that these GAs have on different types of problems. The purpose of comparing GAs on different tasks is: (1) To show that these new GAs are able to solve a variety of different tasks; (2) To try to understand why some of the GAs work better on certain tasks than others.

Testing our GAs on a real-world engineering task is outside the scope of this thesis. In the last chapter of this thesis we discuss some of the tasks that we believe are suited to being solved with our algorithms but we leave the testing to the reader. The overarching goal is to provide a recipe book with examples for creating novel algorithms based on a new way of viewing artificial evolution.

1.2 Structure of Thesis

This thesis is divided into three separate parts. The first part contains the Introduction, Literature Review and a theoretical chapter describing our new view of evolution that separates evaluation from selection. The Literature Review covers related work in artificial evolution as well as reviewing some of the group evolution work in biology. The 'Separating Evaluation from Selection in Artificial Evolution' chapter (Chapter 3) not only explains our new view of evolution in detail but also applies it to two different existing evolutionary experiments demonstrating how it can be used to analyse existing genetic algorithms.

The second part of this thesis, which is called Experiments I, presents a novel family of GAs that vary the relationship between evaluation and selection. Chapter 4 which is based on work previously published in Tomko et al. (2011) presents the Group GA which implements both group evaluation and group selection. We show how this type of GA can evolve solutions to problems where an explicit division of labour is required. Chapter 5 investigates the Binomics GA (first presented in Harvey and Tomko, 2010) which evaluates at the group level and selects at the individual level. In this chapter the Binomics and Group GA are compared on a number of different tasks to determine the effects that

varying the level of selection, while keeping the level of evaluation at the group level, has on evolution. One of the main problems with any GA where groups of individuals are evaluated is that a group size needs to be pre-set ahead of time. In Chapter 6 we expand on the work in Tomko et al. (2012) and present methods that avoid the need to pre-set the group size of these types of GAs. One of the inspirations for these symbiotic, group-level evaluation GAs was bacterial evolution and specifically recent metagenomic research which shows that in bacterial evolution, unlike in eukaryotic evolution, distinguishing individual bacteria within the overall colony is very difficult. This is discussed further in the Literature review chapter (Chapter 2).

The third part of this thesis (Experiments II) shifts the focus from the levels of evaluation and selection to show how other types of GAs can be developed using our view of evolution. Again, using bacterial evolution for inspiration we develop and test GAs that implement bacterial-like horizontal gene transfer which differs from the more conventional vertical gene transfer found in most GAs. Chapter 7 which is based on work presented in Tomko et al. (2013) presents the Unconstrained GA (UGA) which is an algorithm that implements massive amounts of horizontal gene transfer in terms of gene shuffling. We show that on tasks with a lot of sub-optimal solutions the UGA outperforms a standard GA. In Chapter 8 we analyse why this occurs by testing a set of related algorithms on a wide variety of tasks and show that implementing horizontal gene transfer improves evolutionary performance on landscapes with a lot of local optima. We also find that on some tasks, shuffling the genes of the elite members of the population has evolutionary benefits. Finally in Chapter 9 our conclusions are summarised and discussed. At the end of this thesis there are four appendices that include supplementary material. Appendix A is a discussion of the selection pressures of GAs and is included to show that the benchmark GAs we use to compare our novel GAs to are equivalent to standard GAs. Appendix B explains the reasoning behind using both orthodox and Bayesian statistical methods to analyse our results. Finally, the expanded results for all the tests and parameter sweeps done in this thesis can be found online at ntomko.wordpress.com.

Chapter 2

Literature Review

This literature review is broken into three main sections. In the first section a general overview of artificial evolution and genetic algorithms (GAs) is given. This will provide the reader with the background necessary to understand the key components of any evolutionary or genetic algorithm. At the end of this first section we describe in detail the two GAs we use as benchmarks to compare our algorithms to. The second section summarises the evolutionary algorithms related to the work done in this thesis. This summary is very high-level because many of these algorithms are analysed in more detail later in the thesis. Finally, we discuss group evolution in nature with the purpose of determining whether any of the group evolution theories from biology can be used to develop better artificial evolutionary algorithms. It should be noted that we only summarise the group selection debate, since expressing our opinions on this debate is beyond the scope of this thesis.

2.1 Artificial Evolution Overview

Evolutionary / genetic algorithms (GAs) (Holland, 1975) are a subset of the field of evolutionary computation that also includes evolutionary and genetic programming (Fogel et al., 1966; Koza, 1990) and evolutionary strategies (Schwefel, 1995; Rechenberg, 1973). GAs are a search/optimisation technique based on Darwinian evolution (see Mitchell, 1998; Goldberg, 1989, for a general overview). In a standard GA, a population of individuals corresponding to different potential solutions are genetically encoded and evaluated on a specific task. Each individual genotype is assigned a fitness based on how well it solves the task and the fitter individuals survive to pass on their genes to the next generation while the less fit die off. As in natural evolution, the new population of offspring is generated using recombination and random mutation. Over time, the population guided by fitness

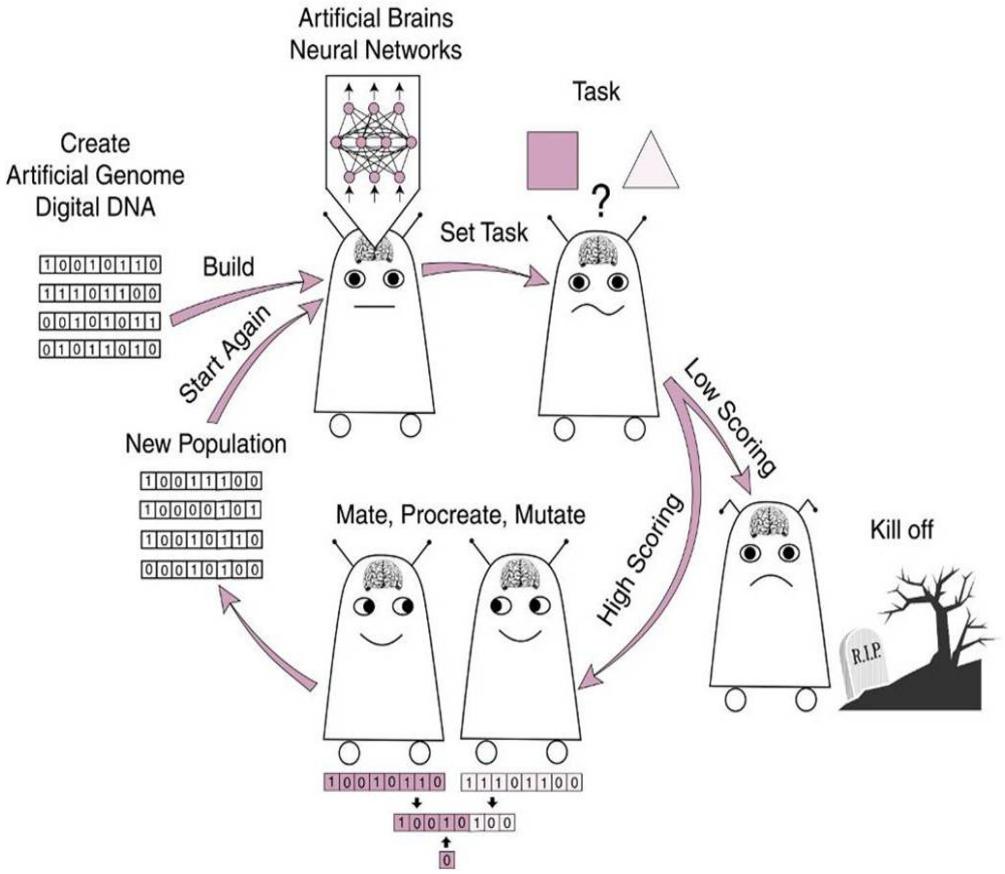

Figure 2.1: A general flow diagram showing the main steps of a genetic algorithm (thanks to Volko Straub for the image).

based selection, mutation and recombination, moves around the solution space (fitness landscape) looking for better solutions.

Figure 2.1 illustrates the main steps of a standard GA. Here we can see that the first step in any genetic algorithm is to create the artificial genotypes that make up the population and define the genotype-to-phenotype mapping. This mapping describes how a genotype in the population is evaluated on the task of interest. For example, one method to evolve robots to navigate an obstacle course is to genetically encode the robot's neural network as a string of real numbered or binary connection weights. In this case the string of connection weights is considered the genotype and the functioning robot is considered the phenotype that is evaluated on the task.

After the initial population has been generated, the evolutionary cycle can proceed. The main steps that make up a single evolutionary generation are: evaluation, selection, and recombination/mutation. The evaluation step differentiates the high scoring individuals from the low scoring individuals so that the higher scoring individuals can be selected and the low scoring individuals can be discarded from the population. Recombination

and mutation are applied to the selected individuals to add variation to the population. If evolution has been setup correctly, over time the population should become fitter and fitter as measured by the pre-defined fitness functions.

Most GAs can be classified as either generational or steady state. In this thesis we will use both types, so in the next couple of sections we will describe the differences between them.

2.1.1 Generational GAs

Traditionally, GAs were presented in generational form. In this type of GA, the entire population is evaluated, selected and modified every generation to create a new population. This roughly corresponds to a natural species that has just one breeding season, say once a year, and after breeding the parents die out without a second chance (Harvey, 2011). The main steps of a generational GA are as follows:

1. Randomly generate initial population
2. Evaluate every individual in the population
3. Select the fitter members of the population, killing off the less fit individuals
4. Recombine and mutate the selected individuals to create the new population that replaces the previous generation
5. Repeat steps 2-5 until some stopping condition is met

2.1.2 Steady State GAs

Steady state GAs (Whitley, 1989; Collins and Jefferson, 1991) are equivalent to natural species that do not have a single breeding season, but instead give birth and die asynchronously across the population. Hence the Steady State GA, which in its simplest form has as its basic event the replacement of just one individual from the population by a single new one. The main steps of a simple steady state GA are:

1. Randomly generate the initial population
2. Choose a mother from the population using some type of selection biased towards the fitter individuals
3. Choose a father from the population using the same method

4. Generate an offpsring using recombination and mutation

5. Add this offspring to the population

6. Keep the population size the same by choosing a single individual to die (this can be done at random or biased to the less fit)

7. Repeat steps 2-6 until some stopping condition is met

If the population size is 50 then fifty times around the loop is roughly equivalent to a single generation of the generational GA described in the previous section. In this thesis we compare our algorithms to either a standard generational GA or steady-state GA; we will describe these algorithms in detail later in this section.

2.1.3 Selection Methods

There are many different ways to implement selection in GAs; here we summarise a few of the more common ones. See Goldberg and Deb (1991) for a comparison of different selection schemes.

Truncation Selection In truncation selection, the choice of parents is restricted to the fittest 20% (or 50% or ...) of the population. This is a parameter that is set before evolution begins.

Fitness-Proportionate Selection Fitness-proportionate selection guides the probability of an individual being chosen as a parent in terms of how fit an individual is compared to the rest of the population. For example, consider a small population of five individuals with fitnesses of 3, 6, 8, 5, and 1. The total fitness of the population is 23 so using fitness proportionate selection the probability of choosing the different individuals as parents is 3/23, 6/23, 8/23, 5/23 and 1/23 respectively.

There are a number of limitations with fitness proportionate selection; these include: (1) How to deal with negative fitness scores (2) If early on in evolution all individuals have zero fitness except for one, then this individual will unfairly dominate selection and (3) If during evolution the fitness of all individuals varies slightly about some average then there will be very little selective pressure on the population.

Linear Rank Selection Due to the issues with fitness-proportionate selection, many GAs use linear rank selection. In linear rank selection, the individuals are sorted according

to fitness and the probability of choosing a given individual is based on their rank, not their absolute fitness score like in fitness proportionate selection. Once the individuals have been ranked their fitness is rescaled based on this relative ranking. A common choice made is to allocate (at least in principle) 2.0 reproductive units to the fittest, 1.0 units to the median, and 0.0 units to the least fit member, similarly scaling pro rata for intermediate rankings. Using this method, the probability of being a parent is proportional to these rank-derived numbers, rather than to the original fitness scores.

Tournament Selection Tournament selection is a simple way of implementing linear rank selection. This type of selection consists of randomly choosing two individuals from the population, without regard to fitness, comparing their fitness (the 'tournament') and selecting the winner as a parent. Using this method, the probability of winning a tournament is exactly the same as linear rank selection where fitness is scaled between 0.0 and 2.0.

In this thesis we have chosen to implement tournament selection in all of our algorithms. The reasons for this is that it avoids the many problems with fitness proportionate selection, it is easy to implement and it can be applied to both steady-state and generational GAs.

2.1.4 Mutation

Mutation adds random, undirected variation to the population. Mutation can be implemented in many different ways, here we discuss some of the most common methods used in this thesis.

If the population is made up of binary genotypes then it is common for the mutation rate parameter to define the per locus probability of flipping a bit, where a locus is defined in this case as any single gene within a genotype that is the target of some action, such as a mutation. It is important to note that different conventions exist for specifying a per-locus or a per-genotype mutation rate, and a failure to distinguish between these can cause confusion. For example, if the per-locus mutation rate is set to 10%, then on average, one out of ten genes will be flipped every time a genotype is mutated.

If the genotype is real numbered then a decision needs to made in terms of both how many genes are mutated and by how much they are mutated. We have chosen to use either single-locus or multi-locus mutation, where single-locus mutation means a single locus on the genotype is mutated and multi-locus mutation means that every locus on the genotype is mutated. Of course the number of mutations could be determined probabilistically as

well.

For real numbered genotypes the amount a chosen gene is changed also has to be determined. In this thesis this we have chosen to define this mutation in terms of a normally distributed random number with a certain standard deviation (that is set as a parameter) and a mean of zero. So in the real numbered case a mutation spread of 0.5 means that a normally distributed random number with mean 0.0 and standard deviation 0.5 is added to the gene.

2.1.5 Recombination

In GAs, recombination can be thought of as the method of creating a single offspring from two or more parents. In this thesis we mainly deal with two parent recombination so we will limit our overview of recombination to this case.

Two popular types of recombination implemented in GAs are crossover and uniform recombination. In one-point crossover a single locus is chosen at random and the offspring is created by combining the genes on one side of the crossover point of one parent with the genes on the other side of the crossover point of the second parent. In two-point crossover, instead of randomly choosing a single crossover point, two points are chosen. In uniform recombination, an offspring is generated by going along the genotype and at each locus the offspring inherits either parent 1's gene or parent 2's gene with an equal probability.

Both crossover and uniform recombination are types of sexual reproduction because the offspring inherits genes from two parents. Asexual reproduction in GAs can be implemented by generating an offspring which is a mutated copy of a single parent. In this thesis we have chosen to use uniform recombination whenever we generate offspring using sexual reproduction.

2.1.6 Elitism

Elitism in GAs is the process by which un-mutated copies of the fittest individual(s) of the previous generation are automatically copied into the next generation. The number of elite individuals copied is determined by a pre-set parameter that defines either an elite percentage of the population or an absolute number of elite individuals to be copied. Elitism has the benefit of preserving the genotypes of the best performing individuals throughout evolution but can also cause the population to get stuck on sub optimal solutions. In chapters 7 and 8 we investigate novel ways of implementing elitism.

2.1.7 Benchmark GAs used in this Thesis

The two GAs described below are the benchmark algorithms we use to compare some of our novel group evolution GAs to. First we describe a standard generational GA with tournament selection and then we describe the Microbial GA (Harvey, 1996, 2001, 2011) which is the benchmark steady state GA we used.

Standard Generational GA with Tournament Selection The main steps of this GA are listed below. The parameter ELITENUM is the number of elite individuals to be copied un-mutated to the next generation.

1. Evaluate the fitness of each individual genotype in the population comprising of a POPSIZE number of individuals
2. Rank each genotype according to fitness
3. The fittest ELITENUM individuals are automatically copied into the new offspring population unchanged
4. The remaining $POPSIZE - ELITENUM$ offspring are generated as follows:
   (a) Select two parents from the population using the following tournament selection method
      i. Randomly pick two individual genotypes from the population
      ii. Compare the fitness of the two individuals from the population with the fitter becoming a parent
      iii. If the fitness of the two individuals is the same then randomly pick a winner
      iv. Repeat steps i-iii to select a second parent
   (b) Produce a single offspring from these two parents by randomly choosing a single gene from either parent at each locus (this is known as uniform recombination)
   (c) Mutate this offspring
   (d) Add this offspring to the new offspring population
5. These offspring become the population for the next generation

This GA was compared against any of our novel GAs that were of the generational variety to give us an idea of how well our algorithms fared on different tasks.

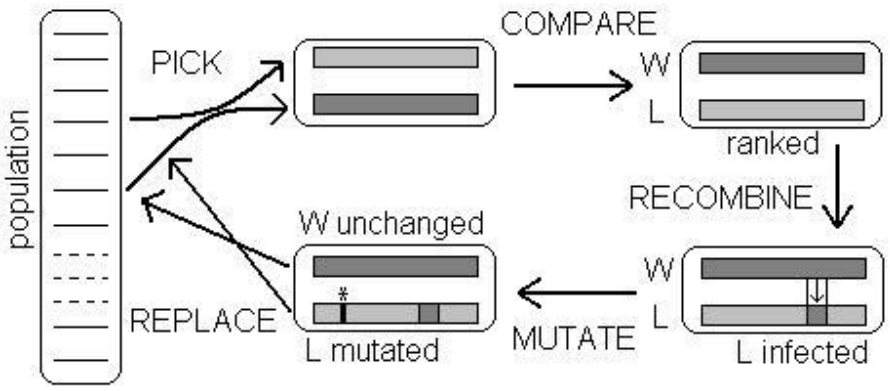

Figure 2.2: An overview of the Microbial GA. Genotypes in the population are represented as strings. One cycle of this GA consists of PICK (at random), COMPARE (their fitnesses to determine a winner = W and loser = L, RECOMBINE (some proportion of the winner's genetic material 'infects' the loser) and MUTATE (the revised version of the loser). (Harvey and Tomko, 2010)

The Microbial GA The Microbial GA (Harvey, 1996, 2001, 2011) is the benchmark steady state GA we use to compare all of our novel steady-state GAs to. This reason we chose this GA is because it implements selection, heredity and variation in a simple manner that makes programming the algorithm extremely simple. The main steps of the Microbial GA are described below and shown visually in figure 2.2.

1. Randomly choose two parents from the population without regard to fitness
2. Compare the fitness of each parent with the winner being the fitter of the two
3. Infect the loser with 50% of the winner's genes
4. Mutate the infected loser's gene(s)
5. Put both individuals back into the population
6. This cycle is repeated until some pre-defined stopping condition is met

The Microbial GA takes advantage of two non-standard tricks that make programming the algorithm simpler, however, as explained below these tricks are purely superficial, the Microbial GA is exactly the same as more familiar steady-state GAs. The first trick is that it selects who-dies (which we refer to as negative selection) rather than who is the parent (which we refer to as positive selection), as in more common GAs. As discussed in more detail in Harvey (2001, 2011) selecting fitter individuals to be parents, whilst making an unbiased choice of who is going to die off is exactly the same as making an unbiased

choice of who is going to be parent and then killing off the less fit individuals, which is what the Microbial GA does.

Secondly, instead of generating an offspring and then replacing the losing parent with this new offspring, the Microbial GA simplifies these steps and just infects the loser with 50% of the winners genes before putting them both back into the population. Again this short-cut makes no quantitative difference to the GA.

As many people are not familiar with the Microbial GA it is important to note that the selective pressure of the Microbial GA is exactly the same as any other steady-state GA which is equivalent to the generational GA described earlier in this chapter. Here we define selective pressure as a measure of how exploitive an algorithm is, in terms of the evolutionary exploitation versus exploration balance and not in terms 'how effective' a GA is in relation to a basket of optimisation problems, which is another way selective pressure is sometimes defined. Defined in this way, the selective pressure of the Microbial GA and generational GAs that use tournament or rank based selection methods can be broadly equivalent for the following two reasons: (1) the selection pressure of tournament-based and rank-based selection methods can be made equivalent Goldberg and Deb (1991) and (2) the negative selection in tournament-based methods has the same selective pressure as positive selection Harvey (1994). In Appendix A we expand on work presented in Harvey (1994) to explain this in detail. The full analysis of selection pressure in the Microbial GA can be found at: http://goo.gl/1yHa1.

2.2 Related Evolutionary Algorithms

Below we summarise the evolutionary algorithms related to this thesis. In many cases these will be compared to some of our novel algorithms later in the thesis. For this reason, only a high-level summary is provided here.

2.2.1 Niching Methods

Now we briefly describe the common genetically based niching methods. In general, niching methods attempt to maintain genetic diversity in the population to either reduce the chance that the population converges on a sub-optimal solution or to evolve a population that contains individuals that work together to solve a given task. For a more in-depth summary on niching see Dick (2005) and Mahfoud (1995).

Fitness Sharing and Clearing

Fitness sharing (Goldberg and Richardson, 1987) is a niching method that relies on some distance metric or similarity measure (either genotypic or phenotypic) between individuals. By using suitable methods to adjust the fitness of any individual according to how many other similar individuals are within some predetermined niche (similarity) radius, there is a tendency for the population to spread out over multiple peaks or niches in the fitness landscape; thus diversity is maintained. Clearing (Petrowski, 1996) is very similar to fitness sharing but, instead of degrading the fitness of individuals within the same similarity radius or subpopulation, it removes the least-fit individuals within the similarity radius from the population. In Horn et al. (1994) it is shown that in Learning Classifier System models where fitness is shared amongst cooperating individuals implicit niching can occur.

Crowding

Crowding was first introduced in De Jong (1975) as a method of removing similar individuals from a population, with the goal of trying to maintain diversity during evolution. Deterministic Crowding (Mahfoud, 1995) is a specific type of crowding that mates two in the population and then if the offspring is fitter, replaces the parent that is most similar to the offspring. It is similar to fitness sharing because it requires a similarity calculation done between individuals, but unlike fitness sharing there is no requirement to pre-specify a similarity radius.

2.2.2 Demes and Spatially Structured GAs

Alternatives to genetically based niching methods include spatially structured GAs; for good reviews see Dick (2005) and Tomassini (2005). In these, the population is structured within some local geographical distribution (demes) that constrains which members of the population are allowed to be selected or recombined with one another. This deme structure allows more genetic diversity to be maintained across sub-populations.

Demes can be added to any standard GA. One of the choices that needs to be made is whether to use one or two dimensional demes. When using one-dimensional demes the population can is viewed as being on a (virtual) ring where mating is restricted to individuals close to you in the ring. For two-dimensional demes, mating is restricted to some 2-D region around the given individual. Spector and Klein (2005) note that one-dimensional demes can be as effective as higher dimensional versions.

2.2.3 Symbiotic GAs

A focus of this thesis is to study new types of artificial group evolution. One of the benefits of group evolution is that it allows evolution to find solutions to tasks that require a division of labour between different individuals. If we define symbiotic as a mutually beneficial relationship between different entities, then symbiotic GAs are those in which a group of individuals is required to solve the task. Below we summarise symbiotic GAs related to our work.

Cooperative Coevolution

Cooperative coevolution was first introduced in Husbands and Mill (1991) and Husbands (1993) for the application of job shop planning and scheduling, and has been further studied in Potter and De Jong (1994) and McIlhagga et al. (1996). This is probably one of the first symbiotic GAs but can also be thought of as a spatially structured GA because of the population is sub-divided at the beginning of evolution. In this algorithm the population is pre-divided into different subpopulations, so it can be thought of as a type of spatially structured GA. Each subpopulation represents a subcomponent required to solve the overall task, hence there needs to be some *a priori* knowledge of the problem so that the appropriate number of subpopulations is chosen. Each subpopulation is evolved separately using a standard GA, but the fitness of the individual members of each subpopulation is based on the performance of the cooperative solutions.

SANE

SANE (Symbiotic, Adaptive Neuro-Evolution, Moriarty and Miikkulainen, 1996, 1995) is a method of symbiotically evolving artificial neural networks (ANNs). As described by the authors, the motivation of SANE is:

> SANE incorporates the idea of diversity into neuroevolution. SANE evolves a population of neurons, where the fitness of each neuron is determined by how well it cooperates with other neurons in the population. To evolve a network capable of performing a task, the neurons must optimize different aspects of the network and form a mutualistic symbiotic relationship. Neurons will evolve into several specializations that search different areas of the solution space. (Moriarty and Miikkulainen, 1995)

In an example implementation, they show a simple ANN with two layers of connection weights, from input to hidden neurons and from hidden neurons to outputs. They treat each hidden neuron, together with its incoming and outgoing connections, as a member of the evolving population. By randomly choosing hidden neurons from the population a full network can be formed. The network as a whole is evaluated on some required task, and the networks score is added to the fitness of each hidden neuron that it contains. Thereafter, the selection, replication, crossover and mutation of members of the population are carried out by conventional GA methods.

Moriarty and Miikkulainen (1999) report that this implementation of SANE works well on such simple ANNs. They also comment that it is feasible to extend this approach to different neuron encodings, and to diverse network architectures including recurrency.

Learning Classifier Systems

Learning Classifier Systems (Holland, 1976; Holland and Reitman, 1978) , especially the Michigan style LCS are very similar to symbiotic GAs. The goal of LCS is to create a cooperative set of rules that together solve the given task. Unlike a traditional optimisation scenario, the search is not for a single fittest rule but a number different types of rules which together give the appropriate behaviour. The rule-base of an LCS had been described as an evolving ecology of rules - "each individual rule evolves in the context of the external environment and the other rules in the classifier system" (Forrest and Miller, 1990).

In Michigan LCSs, each individual in the population is an individual rule or classifier. This is different from the Pittsburgh LCS where the population contains complete sets of rules or classifiers. This means that Michigan LCSs are very similar to symbiotic GAs because a group of population members (rules) needs to be combined to solve the task. One complication that arises in these types of LCS is that a decision needs to be made on how fitness is allocated to each rule bearing in mind that only the collective as a whole can be evaluated. This is relevant to our view of evolution which will be discussed in detail in the next chapter.

SEAM

The Symbiogenic Evolutionary Adaption Model (SEAM, Watson and Pollack, 2000b) is a symbiotic GA with the following key characteristics: (1) It combines partially specified individuals to produce offspring that are more genetically specified than the parents. This recombination operator was used in the Incremental Commitment GA (ICGA, Watson and

Pollack, 1999, 2000a) which is a simplified version of the Messy GA (Goldberg, 1989) which also combined partially specified individuals before evaluation. (2) It uses group evaluation to provide templates that avoid the need to evaluate partially specified individuals and (3) It maintains diversity by using Pareto coevolution which is defined as a method:

> ... which segregates competition to maintain diversity, and prevents large sub-optimal strings from replacing small optimal strings (Watson and Pollack, 2000b)

SEAM was shown to be able to solve the Hierarchical If-and-only-if (HIFF) and Shuffled HIFF landscapes (Watson et al., 1998; Watson and Pollack, 1999).

2.2.4 Group Evolution in Evolutionary Robotics

One of the first pieces of research to study the application of symbiotic GAs to evolutionary robotics was Quinn et al. (2003); Quinn (2006). They show how symbiotic GAs could be used to evolve homogeneous multi-robot systems and showed how a division of labour emerged even within a population of homogeneous controllers. This work was extended on by Floreano et al. (2008) and Waibel et al. (2009) who use symbiotic GAs to evolve both homogeneous and heterogeneous teams of robots on both cooperative and altruistic tasks. In section 3.4 we analyse their algorithms in detail to demonstration of some of the benefits of our view of evolution. To avoid repetition we will not summarise the algorithms here.

2.2.5 Artificial Ecosystem Evolution

Up to this point the symbiotic GAs described in this section have been mainly targeted towards solving optimisation problems. These types of algorithms can also be used to study natural evolution. For example Waibel et al. (2011) use GAs to test Hamilton's rule (Hamilton, 1964) for predicting altruistic behaviour. In another set of experiments, Williams and Lenton (2007) use symbiotic GAs to model artificial microbial ecosystems. In their model, flask ecosystems are evolved for different abiotic, phenotypic traits where fitness is evaluated at the ecosystem level.

2.2.6 In-Vivo Group Evolutionary Experiments

Group evolution can also be simulated in-vivo. Even though all the experiments in this thesis are in-silico, it is important to present different ways group evolution has been studied by biologists in a laboratory setting.

Goodnight and Stevens (1997) provide a good summary of different laboratory experiments that have shown a significant response to group evaluation and selection. One of the first set of experiments which demonstrated this were by Wade (1977) on *T. castaneum* (flour beetles). Forty-eight groups of beetles containing 16 adults per group were allowed to live and reproduce for 37 days. Then the number of adults in each group were counted and the groups with the highest number were used to populate the next generation. For example if the best group had 100 adults, then it was used to create 6 new groups of 16 beetles ($6 * 16 = 96$). The four remaining beetles from the best group were discarded and then individuals from the second best group were used to form as many groups as possible. This process was repeated until 48 new groups were formed.

In ecosystem selection, communities of symbiotic species are selected based on some ecosystem level trait. In Swenson et al. (2000) small, soil-based ecosystems were evolved for phenotypic traits such as pH level and amount of above-ground biomass.

2.2.7 GAs with Multi Individual Recombination

In Part II of this thesis we explore the effect of implementing massive amounts of horizontal gene transfer (HGT) in GAs. Here we review related algorithms including those that implement some sort of multi-parent recombination and those that are based on bacterial evolution.

Multi-Parent Recombination (Eiben and Schippers, 1996), Bit Based Simulated Crossover (BSC) (Syswerda, 1993), Probability Based Simulated Learning (PBIL) (Baluja and Davies, 1998) and Gene Pool Recombination (GPR) (Muhlenbein and Voigt, 1995) are examples of GAs that apply recombination to more than two individuals at a time, which is similar to implementing HGT.

In Eiben and Schippers (1996) a variety of multi-parent recombination schemes were tested using anywhere from 2 to 16 parents on different NK landscapes (which are described in detail in section 7.2) to try to understand how varying the amount of sexual recombination impacts evolutionary performance. GPR (Muhlenbein and Voigt, 1995) only uses two parents to generate a single offspring but these parents are constructed by randomly choosing genes at each specific locus from the entire gene pool after evaluation and selection have been applied to the population. BSC (Syswerda, 1993) is similar to GPR but is restricted to binary genotypes. In this algorithm, offspring are generated using population level probability distributions. These distributions are constructed by counting the number of 1's and 0's at each locus position and then weighting this distribution by

each individual's fitness. PBIL (Baluja and Davies, 1998) is almost identical to BPC but instead of having a population of individuals there is a probability vector that represents the population. This vector is used to generate new individuals which are evaluated and then used to update the vector.

The Pseudo-Bacterial GA (PBGA) (Nawa et al., 1997) and the Bacterial Evolutionary Algorithm (Nawa and Furuhashi, 1998) are two GAs that are inspired by bacterial evolution. Both include a genetic operator which they call the 'Bacterial Operator'. This operator attempts to mimic gene transduction which is one process by which bacteria can horizontally transmit parts of their genome to other bacteria. The goal of implementing gene transduction in a GA is to try to speed up the spread of high fitness genes through the population. Nawa et al. (1997); Nawa and Furuhashi (1998) apply their algorithms to the evolution of fuzzy rules.

2.3 Group Evolution in Biology

The remainder of this literature review introduces some of the key concepts from biological group evolution. Even though the focus of this thesis is to look at new ways to implement group evolution in artificial evolution and GAs, it is important that we review the current state of group evolution in biology to determine whether any of these natural evolution theories can be used to improve artificial evolution.

Group selection in nature has been a controversial and confusing subject in evolutionary theory for over fifty years. Even today the debate over whether group selection played a role in the evolution of cooperation is ongoing; a significant portion of a recent issue of Nature (August 2010, vol.466) was dedicated to this debate. Here we will only summarise some of the key points in this very complex debate; we will not take sides. For readers interested in the current state of the group selection debate see: Okasha (2010), West et al. (2010), Dawkins (2012), Wilson (2012b), and Wilson (2012a).

After summarising the history of group selection and related theories we will show that despite initial appearances, much of this debate and related skepticism over group selection in nature is not relevant to the application of group selection or evaluation in artificial evolution. Then in the next chapter we will discuss how none of these theories explicitly distinguish between evaluation and selection, and, therefore, it is difficult for them to be applied to the development of novel evolutionary algorithms. Finally, we will briefly discuss bacterial evolution and metagenomics and how they relate to this thesis.

2.3.1 Group and Multi-Level Selection Theories

West et al. (2010) believe that much of the confusion over group selection is due to the many different ways group selection has been defined and used. Therefore, they try to clarify things by dividing group selection into two main types: old group selection and new group selection.

What West et al. (2010) refer to as old group selection was first proposed by Darwin to explain the evolution of the social structure of insects, one of the first observed examples of altruistic behaviour. Altruism benefits the recipient of the act, but is costly from a fitness standpoint to the act giver. Darwin could not reconcile this type of behaviour with his evolutionary theory of individual selection and survival of the fittest and therefore hypothesised that altruistic behaviour evolved because it was the insect colony as a whole that was selected for and not just the individual insect.

Old group selection was used by V. C. Wynne Edwards to try to show that group selection was responsible for the regulation of animal population densities (Wynne-Edwards, 1963). John Maynard Smith showed that under the right conditions this type of group selection could work, but he was very skeptical that these conditions would occur naturally (Maynard Smith, 1964, 1976). His skepticism stemmed from the belief that the term 'group selection' should be reserved for evolutionary processes where there are partially (or fully) isolated groups which can reproduce and go extinct. He saw these conditions as necessary for groups to be the evolutionary units of selection; meaning they have the properties of variation, multiplication and heredity (Maynard Smith, 1976).

This 'old' group selection theory was widely accepted well into the 20th century but fell out of favour for the following reasons: (1) Darwin's version of group selection requires well defined groups, so in nature who or what is imposing these structured groups? (2) It was thought that even if there are groups being selected for, individual selection would always be a more significant evolutionary force because there are always more individuals than groups and individual turnover is higher than group turnover (Maynard Smith, 1976). The Haystack Model was a model Maynard Smith came up with that met his criteria for group selection. In this model Maynard Smith (1964) imagined a species of mouse which lived in haystacks. Each haystack is colonized by a single fertilized female whose offspring only come out of their haystack once per year to mate with members of other haystacks in order to start new colonies. Using this model Maynard Smith was able to show how group selection could result in the evolution of altruistic behaviour, but in Maynard Smith (1964) he concludes that the Haystack Model is "... too artificial to be worth pursuing further."

Maynard Smith believed that one of the mistakes people make is using group selection to explain the evolution of a group trait before determining whether the conditions for group selection are met and whether the same trait can be explained using kin or individual selection (Maynard Smith, 1964, 1976).

According to West et al. (2010), 'new group selection' theories were introduced as a result of these criticisms. These theories are commonly known as multi-level selection theories and show that selection can occur at more than one hierarchical level. Theories by Colwell (1981); Damuth (1988); Hamilton (1975); Wilson (1975, 1977) are multi-selection theories that attempt to show that cooperation can evolve when between-group selection is stronger than within-group selection.

MLS1 and MLS2 which are two of the more well known multi-level selection theories were proposed by Damuth (1988). The difference between MLS1 and MLS2 has to do with whether the individual (MLS1) or the group (MLS2) is the object of evolution, where object of evolution is determined by whether you are interested in the changing frequency of individual or group traits (Wu and Banzhaf, 2011). In MLS1, group selection refers to the impact group membership has on individual fitness, while in MLS2, group selection refers to the change in frequency of different types of groups (Damuth, 1988). According to Damuth (1988) if you are interested in the evolution of altruism, which is an individual trait, then MLS1 should be used.

2.3.2 Inclusive Fitness and Kin Selection

Group selection fell further out of favour when William Hamilton published his Kin Selection/Inclusive Fitness (KS/IF) model as an alternative theory that could explain altruistic cooperation (Hamilton, 1964). Inclusive fitness is what individuals appear to maximise when taking into account both direct and indirect fitness effects (West et al., 2010), whereas the Fisher equation (Fisher, 1930) took account of only direct fitness effects.

Since Hamilton's theory was published, much of the debate over group selection has revolved around whether there is any difference between group selection theories and the theory of KS/IF. West et al. (2010) believe that group selection can easily be explained using the KS/IF model and there is nothing any of these group selection theories can explain that KS/IF cannot. This view was recently disputed by Nowak et al. (2010). As we discuss in section 2.3.4 we take the view that this controversy is not relevant to artificial evolution and therefore for the most part can be ignored for the purposes of this thesis. Most of this ongoing debate is exclusively framed in terms of explaining altruism, but

as we will discuss below, in artificial evolution, from an engineering standpoint, we can manipulate conditions so as to promote the evolution of mutually beneficial cooperation.

2.3.3 Dawkins' Replicators and Vehicles

Richard Dawkins' replicators and vehicles (Dawkins, 1982) are his attempt to clear up the misunderstanding over the unit of selection in biological evolution and therefore deserve some attention. According to Dawkins, replicators are "any entity in the universe of which copies are made" (Dawkins, 1982), while vehicles are just the "survival machines" for the replicators. In other words, replicators replicate and vehicles of selection interact with the environment. He believes that only prebiotic molecules and genes can be replicators and therefore all other entities such as individuals, species, or groups are only vehicles for the replicating genes.

> Active, germ-line replicators, then, are the units of selection in the following sense. When we say that adaptation is "for the good" of something, what is that something? Is it the species, the group, the individual, or what? I am suggesting that the appropriate 'something', the 'unit of selection' in that sense, is the active germ-line replicator. (Dawkins, 1982)

From this quote, we can see that Dawkins believes that only genes can be the 'units of selection' because all evolutionary adaptations are for the benefit of the genes (this is part of the gene-centered view of evolution). Based on this logic, vehicles of selection are the entities that house the 'units of selection', which according to Dawkins can be entities of any hierarchical level. Therefore vehicle selection and replicator survival are the same process because if the vehicles are selected then the replicators survive (Dawkins, 1982). Dawkins thinks that the 'individual' versus 'group' selection debate is about different vehicles, not over different units of selection because in his view only genes (replicators) can be units of selection. In section 3.3 we compare Dawkins' vehicles and replicators to our view of artificial evolution.

2.3.4 How Does this Group Selection Debate Relate to Artificial Evolution?

In artificial evolution, the conditions required for group selection and evaluation can easily be enforced by the experimenter. S/he can explicitly define and enforce group structure in the simulation, can ensure that group turnover occurs either at the same or different time

scales as individual turnover and can also eliminate within-group competition. Therefore, the concerns regarding whether the conditions necessary for group selection do or do not occur in nature should not carry over to the field of artificial evolution because the experimenter is free to enforce such conditions.

Furthermore, much of the debate about group selection in biology has been about whether it can explain the phenomenon of altruism, whereby individuals do something that reduces their own fitness in order to increase the fitness of another individual. In artificial evolution we are not necessarily concerned with this phenomenon; a more usual goal is to evolve mutually beneficial cooperation, whereby individuals work together to perform a group task, increasing the fitness of each member. Thus much of the debate in biology, including the debates around kin selection and inclusive fitness, is tangential to the issues of concern to those using group selection as an engineering tool.

In natural evolution it is difficult to separate evaluation and selection because fitness can be only calculated *after* selection and reproduction have occurred. This is probably one reason why the theories described in this section solely describe evolution in terms of selection and do not explicitly talk about evaluation. In artificial evolution fitness needs to be evaluated *before* selection occurs so that entities can be selected based on differential fitnesses. This means that in natural evolution, group selection versus individual selection can only be determined (which here means discovered or deduced) *a posteriori* by understanding which hierarchical levels (e.g. the group or the individual) benefit from the evolved traits/characteristics. On the other hand, in artificial evolution the hierarchical level of both evaluation and selection need to be determined (which here means decided by the overseer) *a priori*. Therefore trying to apply the binary classification of group versus individual selection to artificial evolution is limiting because it fails to take into account the fact that evaluation and selection are different.

For these reasons attempting to apply any of the theories discussed in this section to the development of artificial evolutionary algorithms is difficult and will limit the space of potential algorithms. At the end of the next chapter we will compare these theories to our view of artificial evolution which discriminates between evaluation and selection.

2.3.5 Bacterial Evolution and Metagenomics

In this thesis, many of the novel GAs we introduce have been influenced by bacterial (microbial) evolution. Recent research in the field of metagenomics has highlighted how bacterial evolution is very different from eukaryotic evolution. Some of these differences

include the significance of horizontal gene transfer (HGT) and how in microbial communities, distinguishing between individuals and groups is very difficult because of the symbiotic relationship between different bacteria. A significant part of this thesis implements these two ideas into artificial evolution and studies how they affect the performance of GAs.

Horizontal or lateral gene transfer (HGT) occurs mainly in microbial communities such as bacterial colonies and is the process by which genes are transferred between individuals. Vertical gene transfer (VGT), on the other hand, is the process by which genes from parent(s) are passed on to one or more offspring and is the primary mechanism of eukaryote evolution. There are many different mechanisms of HGT (Thomas and Nielsen, 2005), the most common being: transduction where genes are transferred between bacteria with the aid of phages or virus and conjugation where genes are transferred through cell to cell contact using plasmids. It has been shown that HGT is responsible for a significant amount of evolutionary innovation including antibiotic resistance, virulence attributes and metabolic properties (Ochman et al., 2000).

Previously our understanding of microbes has been based on studying rather few samples. In order to perform reproducible scientific experiments, well-defined species have been used, often with great care taken to culture them in the lab in isolation to ensure their purity. It is typically assumed that the test-tube is full of a single species that is genetically well-defined. Research in the field of metagenomics has recently realized that such assumptions may not hold true in the real world. As a very recent field, most of the reporting on metagenomics comes in specialised technical research papers. Useful overviews for a more general audience include Handelsman (2004), a report by the Committee on Metagenomics (2007) and Eisen (2007).

In microbial communities there may often be large functional differences between close relatives; further, horizontal gene transmission means that many functions (chemical cycles) typically performed by one species may be also performed by very different species. Microbes such as bacteria do not undergo sexual reproduction, but reproduce by binary fission. But they have a further method for exchanging genetic material, bacterial conjugation. Chunks of DNA, plasmids, can be transferred from one bacterium to the next when they are in direct contact with each other. Whereas the genomes of different humans vary by around 0.1%, different members of what may conventionally be termed a microbial species (or phylotype) can differ by up to 30%. It now makes conceptual sense and technical developments make it possible to perform shotgun sequencing of a whole bucketful of microbes taken from the Sargasso Sea (Venter et al., 2004) and con-

sider the metagenomic sequence of the whole community, together with the functions that such a community collectively performs. Shotgun analysis involves breaking up the DNA randomly into small segments that are individually sequenced; then using computational methods, by seeking overlaps in these fragments, they are built up again into a complete sequence.

An example of the importance of the symbiotic relationship between bacteria is in the human body. Recent analysis has sown that there are 10 times as many microbial cells in a human body than there are human cells; the human metagenome contains perhaps a hundred times more genes than the human genome (Qin et al., 2010). Many such bacteria are essential for our human well-being, and in turn they rely on us to provide them with an appropriate environment.

2.4 Summary

In this literature review we have:

1. Provided a general summary of the main steps of most genetic algorithms (Section 2.1).

2. Described the benchmark GAs we use to compare our GAs to (Section 2.1.7).

3. Reviewed existing GAs and evolutionary algorithms most related to the work done in this thesis (Section 2.2).

4. Highlighted the key points in the on-going debate over whether group selection occurs in nature (Section 2.3)

We believe this gives the reader the necessary background and context to properly understand how our new view of evolution fits in to the field artificial evolution as well as understanding how our new algorithms compare to some of the existing methods of group evolution. In the next chapter we describe our new view of evolution in detail, specifically highlighting the benefits of separating evaluation from selection.

Chapter 3

Separating Evaluation from Selection in Artificial Evolution

3.1 Introduction

One major difference between natural and artificial evolution is that in artificial evolution, evaluation and selection are separate and distinct steps. This is in contrast to natural evolution, where there is no explicit fitness function and natural selection is used to describe both evaluation and selection, because in nature it can be argued that these two processes cannot be viewed separately. For the purposes of this thesis, evaluation is defined as the process of testing some entity on a task and assigning it a fitness score based on an explicit fitness function and selection is defined as determining which entities should survive based on differential fitnesses between entities. As we discuss in section 3.3, this difference is one of the main reasons why attempting to apply natural evolutionary theories such as group, multi-level or kin selection, that we summarised in the previous chapter, to artificial evolution is difficult and limiting. In this chapter we present our novel way of viewing artificial evolution that is based on the conceptual discrimination between evaluation and selection and discuss how using this view of artificial evolution opens up the potential space of interesting and novel algorithms.

It is important to note that for the purposes of this thesis we define 'artificial evolution' as evolutionary experiments or algorithms where there is an explicitly defined fitness or cost function that is used to evaluate entities in the population. We realise that attempting to apply our view to open-ended artificial evolutionary simulations, such as Avida (Adami and Brown, 1994), where there is no explicit fitness function will be more challenging and therefore for the purpose of the thesis we limit the application of our view of artificial

evolution to experiments where there is an explicit fitness function. Whenever a fitness function is used, the method of evaluation and selection both need to be defined *a priori*. In the artificial evolution literature this distinction is rarely discussed (one exception being Watson and Pollack (2000b)) and the standard convention is to describe different evolutionary algorithms and experiments solely in terms of 'selection', in which this is used to refer to what we call 'evaluation' or 'selection', without drawing a distinction between the two.

Due to the influence of natural evolution it is common to describe evolutionary experiments in terms of 'individual' or 'group' selection, without specifying whether it is the evaluation or selection step in artificial evolution that is being referred to. Viewing artificial evolution in the same way as natural evolution can cause confusion because in many cases experiments labeled as 'group selection' are actually individual selection with group-level evaluation. For example, among the most famous examples of 'group selection' in artificial evolution are the experiments by Craig and Muir (1996) which showed that when battery hens were evolved basing selection on the number of eggs produced by a cage of hens, rather than the number produced by a single hen, egg production significantly increased. In factory farms for eggs, multiple hens are housed in cages rather than open pens because this is a more cost efficient way of maintaining a large number of hens. When evaluation and selection is done at the individual hen level (by picking the hen that lays the most eggs), the hens compete against others in the same cage, become aggressive and end up pecking and harming their cage-mates. However, by choosing to evaluate cages of hens on the egg production of the whole cage rather than of individual hens, aggressive hens that reduced overall egg production were bred out. By changing the level of evaluation from the hen to the cage after six generations Craig and Muir (1996) found that hen survival increased from 160 to 348 days and eggs per hen went from 91 to 237 (averages). This paper is called "Group Selection for adaptation to multiple-hen cages..." even though the significant change was varying the level of evaluation not the level of selection. This is an example of why in artificial evolution it is useful to look at both the selection and evaluation processes separately.

Drawing a distinction between evaluation and selection in artificial evolution has two major advantages. Firstly, it makes things clearer and easier to understand when trying to compare different evolutionary experiments, especially when considering those related to group evolution in any form; and secondly, it extends the space of potential evolutionary algorithms beyond the standard case where they both act on the individual, allowing novel

algorithms to be developed. Our view of artificial evolution is based on identifying the Units of Evaluation (UoE) and Units of Selection (UoS) and the interactions between them. Not only does this view differentiate between what is being evaluated and what is being selected, it also eliminates the need to apply an absolute 'group' or 'individual' label because evolution can be described in terms of the hierarchical relationship between the UoE and UoS. The problem with trying to understand evolution in terms of 'group' or 'individual' evaluation/selection is that because evolution can act on multiple hierarchical levels, what is a 'group' and what is an 'individual' is largely a matter of perception.

The first section of this chapter presents our new view of evolution in detail. Then we spend some time comparing this view of evolution to some of the biological theories presented in the previous chapter. To demonstrate how this view of evolution can be used to analyse existing GAs, we use it to analyse a set of group based evolutionary robotics experiments. Finally we describe how this view will be used to develop new GAs in the remainder of this thesis.

3.2 The UoE / UoS View of Artificial Evolution

Here we describe our view of evolution in detail and apply it to the evolution of battery hens experiments (Craig and Muir, 1996) discussed earlier as an example of how it can be used. This view is based on the conceptual discrimination between what is being evaluated, which we call the Units of Evaluation (UoE), and what is being selected, which we call the Units of Selection (UoS) and the interaction between them. Using this view, the five main evolutionary steps are as follows.

1. Evaluate the UoE according to the pre-defined fitness function
2. Assign fitness credits from the UoE to UoS
3. Select the UoS on the basis of the assigned credits
4. Use the selected UoS to produce the next generation of offspring
5. Use the offspring to generate new UoEs

This evolutionary cycle (figure 3.1) can be used to describe any evolutionary algorithm or experiment as long as there is some pre-defined, external fitness function. In this figure the UoE, UoS and offspring are represented with different shadings (UoE is white, UoS is light gray and offspring are dark gray) to highlight the fact that in artificial evolution there

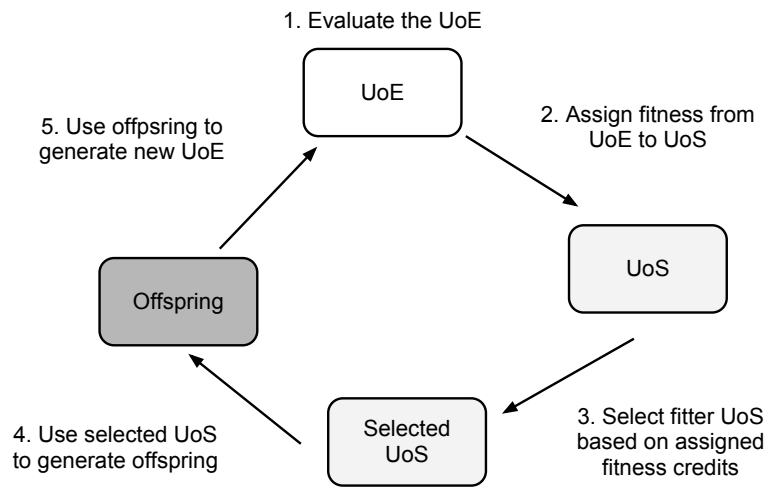

Figure 3.1: A flow diagram showing how the units of evaluation (UoE) and units of selection (UoS) interact in artificial evolution. In this figure the different evolutionary entities (UoE, UoS and offspring) are represented by different shadings (white, light gray and dark gray respectively).

is no reason the UoE and UoS have to be part of the same hierarchical level, even though in most standard evolutionary algorithms this is the case. We can categorise different algorithms according to whether the UoE is (1) the same as the UoS (UoS = UoE), (2) part of a higher hierarchical level than the UoS (UoE $\supset$ UoS), or (3) part of a lower hierarchical level (UoE $\subset$ UoS). It would also be technically possible, although difficult to think of practical reasons, to make the UoE and UoS different entities on the same hierarchical level.

To illustrate how changing the UoE and UoS can impact evolution we will apply our view of evolution to the battery hen experiments. The revolutionary outcome from the Craig and Muir (1996) experiment was that they showed how egg production could be significantly increased by changing both the UoE and UoS from the individual hen to the cage of hens. As we noted earlier, the name of the Craig and Muir (1996) paper is "Group Selection for adaption to multiple-hen cages...", even though it could be argued that changing the UoE was as important, if not more important, as changing the UoS. Below we describe four different ways battery hens could be evolved with different UoE and UoS [1]. These examples are illustrated in figure 3.2. In this figure the UoE, UoS and offspring are represented using the same colours as in figure 3.1 and the individual hens, cages of hens and offspring are shown using circles, rectangles and triangles respectively.

[1]Note that these examples are purely illustrative and may not actually be effective methods of evolving egg-producing hens.

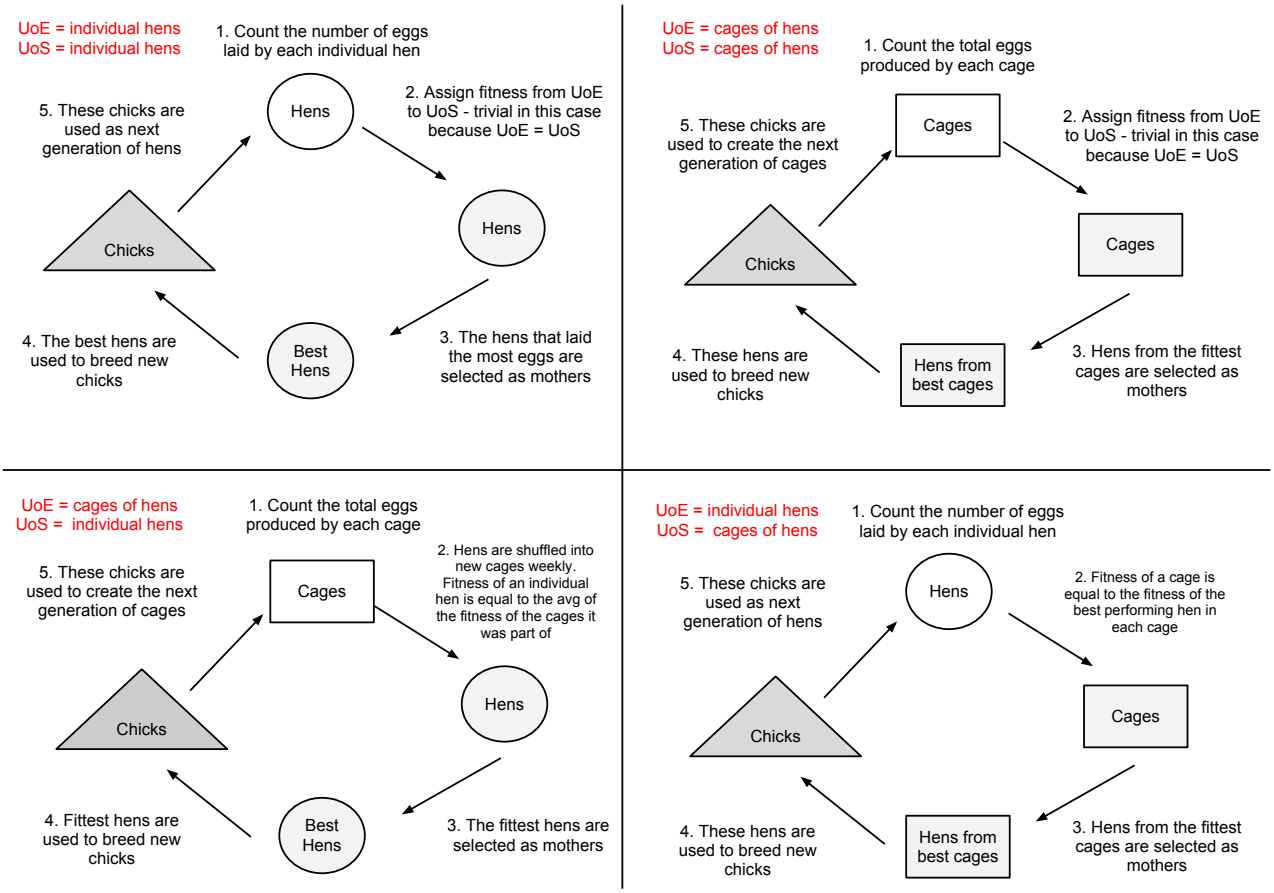

Figure 3.2: Here we use the hen example to illustrate four possible combinations of the units of evaluation (UoE) and units of selection (UoS) that could be used for evolving egg producing hens. In each quadrant, the evolutionary cycle for different combinations of UoE and UoS is shown (see text for details). Like in figure 3.1 the UoE are white, the UoS are shown in light gray and the offspring are shown in dark gray. The individual hens, cages of hens and chicks are shown as circles, rectangles and triangles respectively.

For example, a light gray circle means that the individual hens are the UoS.

1. One possibility is that hens are evaluated individually, and the hens that lay the most eggs (fittest hens) are chosen to be parents of the next generation. In this case the UoE and UoS are both individual hens (UoS = UoE) .

2. Another possibility is that the number of eggs laid by individual hens is ignored, instead only the total number of eggs laid in each cage are counted. The best cages are selected and used to produce the next generation of hens. In this case the UoE and UoS are both cages of hens (UoS = UoE).

3. Again only the cage totals are counted, but this time the farmer decides to randomly re-shuffle the cage members every week and calculate the fitness of an individual hen

as the average of the fitness of the cages it was part of. When breeding time comes around the fittest hens are selected. In this case the UoE is the cage of hens but the UoS is the individual hen so the UoE $\supset$ UoS.

4. The hens are evaluated individually but all hens in the cage that contains the fittest hen are sent off for breeding. In this case the UoE is the individual hen but the UoS is the cage of hens so the UoE $\subset$ UoS.

These are just four illustrative examples of how battery hens could be evolved with the intention of maximising egg production. It is important to understand that these four methods of evolving battery hens can only be differentiated because we have separated evaluation from selection.

When the individual population members are able to solve the task on their own, like in the battery hen example, identifying the UoE can be a little ambiguous. This is because the fitness of a cage of hens will in some way be based on the number of eggs the individual hens lay, so it could be tempting to say that the UoE is always the individual hen. We believe that in these cases the easiest way to determine the UoE is to ask whether the method of determining fitness can be calculated without any knowledge of the individual entities that make up the group. If the answer is yes then the UoE is the group, otherwise the UoE is likely the individual. To illustrate this more clearly below are three examples of how hen fitness could be calculated. In each case we explain what the UoE is assuming the UoS is a cage of hens.

Case 1 - Fitness as the Total Number of Eggs per Cage In this case fitness is calculated as the total number of eggs laid by the cage so the UoE is the cage. This is because a visiting alien observing fitness being calculated in this way would probably think that it was the cage laying the eggs rather than the individual hens, because no knowledge of the individual hens is required to calculate fitness in this way. Using this logic, anytime the fitness of a group is calculated as the sum of individual fitnesses the UoE will likely be the group. In this case the UoE and UoS are the same, so assigning fitness from the UoE to UoS is trivial.

Case 2 - Fitness is Number of Eggs Laid by a Random Hen In Case 2, fitness is calculated by randomly choosing a single hen from each cage and determining how many eggs it laid and then assigning this fitness to the cage. Here the UoE is an individual hen because the same alien visiting earth would realise that it is not the cage laying eggs but

rather the individual hens. Unlike Case 1, knowledge of the individual hens is required to calculate fitness. Assigning fitness from the individual hen (the UoE) to the cage (the UoS) is also trivial in this case because cage fitness is just equal to the number of eggs the randomly chosen hen laid.

Case 3 - Fitness is the Best Hen in the Cage Here fitness is equal to the number of eggs the best hen in each cage lays (as in the example in the bottom right of figure 3.2). As in the previous case, the UoE is the individual hen, since knowledge of the eggs laid by the individual hens is required to calculate fitness in this way. Fitness is assigned from the individual hen (the UoE) to the cage (the UoS) by taking the maximum number of eggs laid by any one hen.

It is important to note that this ambiguity of determining the UoE disappears if individual fitness is meaningless. An example of this could include a job-shop scheduling task where the fitness of an overall job-shop schedule can only be calculated by taking into account how all the machines in the shop work together. The fitness of individual machines in the shop is meaningless without looking at the overall schedule. Next we describe each of the steps of our evolutionary cycle in general so they can be applied to the analysis and development of GAs.

3.2.1 Step 1: Evaluating the UoE

The first step in this evolutionary cycle is to evaluate the UoE based on some explicitly defined fitness or cost function that is specified *a priori* by the experimenter before evolution begins. In a standard generational GA the entire population is evaluated once per generation, while in a steady-state GA the entire population is not evaluated as a whole but instead individuals from the population are evaluated whenever a new offspring is to be produced.

In standard GAs (both generational and steady-state) it is assumed that each individual in the population encodes a full UoE which can be evaluated on its own, but as we will show in this thesis this does not have to be the case. In some cases, like symbiotic GAs, the individuals in the population only encode partial solutions so some method of combining them into full UoE needs to be specified. This will be discussed in step 5 below.

3.2.2 Step 2: Assigning Fitness from the UoE to the UoS

Fitness of the UoE is determined using some pre-defined fitness function (step 1). The fitness of the UoS is derived from this evaluation and this assignment procedure can be

done in different ways depending on the hierarchical relationship between the UoE and UoS. This choice is related to the credit assignment problem (Jong, 1987; Grefenstette, 1988; Holland and Reitman, 1978), which normally refers to how fitness is sub-divided among the components of a cooperative solution.

If the UoS = UoE then the fitness of the UoS is the fitness of the UoE because they are one and the same. When the UoS $\neq$ UoE the choice of how fitness is assigned to the UoS based on the UoE becomes potentially more complicated [2]. Going back to the hen example, if the UoE is the cage of hens, but the UoS is the individual hen, then a choice has to be made on how the fitness as evaluated for each cage (UoE) relates to the fitness as assigned to an individual hen (UoS). The simplest way to do this would be to assign every hen in a given cage the fitness score of the cage but there are many ways fitness can be assigned from the UoE to the UoS; with the optimal method likely being task dependent. Michigan style learning classifier systems (LCS) (Holland, 1976; Holland and Reitman, 1978) and the niching algorithms which we discussed in the Literature Review chapter each have their own methods of passing fitness between the UoE and UoS. In Chapter 5 we will show how changing the method impacts evolution.

3.2.3 Step 3: Selecting Fitter UoS based on Assigned Fitness Credits

After the UoS have been assigned a fitness score based on the UoE, the fitter UoS can be selected as parents for the next generation of offspring. Different methods of selection (fitness proportionate, rank based, truncation, and tournament) were discussed in the Literature Review chapter so we will not spend any more time describing them here. For the purposes of this thesis we have chosen to run all of our algorithms using tournament selection and therefore comparing different types of selection is not part of the scope. For an analysis of different selective schemes please see Goldberg and Deb (1991).

When analysing this selection step, one must be careful not to confuse fitness driven selection processes with non-fitness driven choosing or picking steps, because this can lead to the mis-identification of the UoS. For example, in our second illustrative example of how to evolve battery hens above, both the UoE and UoS are cages of hens because fitness is calculated as the total of number of eggs laid per cage and the cages that lay the most eggs are selected to produce the next generation of hens. There are many different ways that these UoS (cages of hens) can be used to produce the next generation of hens (step 4), for example: the farmer can send all the hens from the fittest n cages off to breed,

[2] It may be possible to relate the case where UoS $\neq$ UoE to the Price equation (Price, 1970) with a non-zero transmission term

or can randomly pick m hens from the fittest n cages to breed, or even can choose to breed only the hens from the single fittest cage. In these three examples, the method of producing the next generation of hens is different, but in all cases the UoS is still a cage of hens because it is cage(s) of hens that are selected based on differential fitnesses. Even in the example where the farmer randomly picks individual hens from the fittest cages, the UoS is a cage of hens because the random picking of hens is not directly driven by fitness of the UoE. Part of this confusion arises from the many different ways the term 'selection' is used. In this paper we try to use 'selection' to refer to fitness based selection and use 'choose' or 'pick' to refer to processes of choosing a subset of entities from a group that is random and not driven by fitness. In terms of our evolutionary cycle, the decision of whether to generate offspring from a sub-set of the UoS falls into step 4 which we describe next.

To complicate matters further, there is also the possibility of having evolutionary experiments or algorithms where there are multiple UoE and UoS which are involved in a multi-step selection process. For example, a farmer could keep track of both the number of eggs produced by each individual hen and the total number of eggs produced by each cage and then select the individual hens that lay the most eggs from the fittest cage. In this example individual hens and cages of hens are both UoE and UoS. These examples show how complex artificial evolution can be. We hope that identifying the UoS and UoE will provide a consistent method of analysing and comparing different evolutionary experiments and algorithms.

3.2.4 Step 4: Generate Offspring with Selected UoS

After the fitter UoS have been selected based on the fitness of the UoE, the next generation of offspring can be produced. To generate offspring in any artificial evolutionary experiment, the type of recombination and mutation needs to be specified (see the Literature Review Chapter for a description of different types). Other factors that need to be considered include deciding whether some sort of group formation method such as propagule or migrant methods (Wu and Banzhaf, 2009) are going to be used. If the UoS is made up of a group of entities then there is the option of whether to use all the entities to generate the offspring or to just use a randomly chosen sub-set of the UoS as we discussed in the previous section.

3.2.5 Step 5: Use the Offspring to Generate New UoE

The final step of the evolutionary cycle is to generate new UoE from the offspring. In standard evolutionary algorithms this is often trivial because each individual offspring encodes a full UoE, i.e. each individual can solve the task on its own, but as we will show this does not need to be the case. In symbiotic GAs each individual in the population is only a partial solution to the task so does not encode a full UoE and, therefore, UoEs need to be constructed using two or more population members. For example, when evolving artificial neural networks (ANNs) it is common for the population to contain n genotypes that encode n ANNs. In this case the UoEs do not need to be assembled each generation because each population member fully specifies a UoE. An alternate method of evolving ANNs is to have a population of entities that encodes the genotype of individual neurons, rather than full networks. In this case, each UoE would have to be assembled by choosing the appropriate number of individual neurons from the population and using them to construct a fully specific network. Both SANE (Moriarty and Miikkulainen, 1996, 1995) discussed in the Literature Review Chapter, the Binomics GA (Harvey and Tomko, 2010) we present in Chapter 5 and the Group GA Tomko et al. (2011) set of algorithms we present in Chapter 4 require that the UoE are assembled from multiple population members.

Even if each individual in the population is a fully specified UoE there are cases where the newly generated offspring are not equivalent to the new population of UoE. One example of this is the Unconstrained GA (UGA) which we present in Chapter 7. In this algorithm, which is related to the multi-parent recombination algorithms described in the Literature Review, the genes of the offspring are shuffled before the new population is created. This can be thought of as implementing horizontal gene transfer on the offspring population.

3.3 The UoE / UoS View Compared to Natural Evolutionary Theories

In the section 2.3 we summarised a number of theories of natural group evolution including Multi-Level Selection Theory (MLS) and Dawkins' vehicles and replicators. Here we compare them to the UoE/UoS view of evolution and show that because they do not explicitly separate evaluation from selection they are limiting in terms of their application to analysing and developing artificial evolutionary experiments and algorithms.

MLS1 and MLS2 are two of the more well known multi-level selection theories that

try to show that natural selection can occur on more than one hierarchical level. Like most natural evolutionary theories, MLS does not explicitly draw a distinction between evaluation and selection but it does hypothesise that fitness can be associated with either the individual or group. For example, according to Damuth (1988), in MLS1 fitnesses are properties of individuals, while in MLS2 fitnesses are properties of groups. Okasha (2005) expands on this definition and states that in MLS1 "A collective's fitness is defined as the average fitness of the particles within the collective." and in MLS2 a collective's fitness is "...the expected number of *collectives* contributed to the next generation.". In the opinion of Okasha (2005), MLS1 is very similar to group selection models that want to explain the evolution of individual traits in a population that is divided into groups.

If we try to think about MLS in terms of UoE and UoS, it could be tempting to say that in MLS1 the UoE is the group and the UoS is the individual, while in MLS2 the UoE and UoS are both the group. The problem with this is that as we said before, in natural evolution, it is difficult to separate evaluation from selection like we do in artificial evolution. Even if we do apply MLS to the development of evolutionary algorithms it is still limiting because the relationship between the UoE and UoS is limited to either UoE = UoS or UoE $\supset$ UoS. Our view of evolution does not restrict the hierarchical relationship between the UoE and UoS and it also examines not only the relationship between the UoE and UoS but how they interact in the overall evolutionary cycle.

There are also some similarities between UoE and UoS and Dawkins' vehicles and replicators (Dawkins, 1982). Even though vehicles and replicators were originally used to study natural evolution, when applied to artificial evolution they can be used to highlight the difference between evaluation and selection. Vehicles can be thought of as the entities being evaluated in the environment (like our UoE) and the replicators can be thought of as the entities that are selected to pass on genetic material based on the success of the vehicles that house them (like our UoS). One of the problems with applying vehicles and replicators to artificial evolution is that Dawkins believes that replicators, which he defines as the unit of selection of evolution, can only be genes. This means that the vehicles always have to be part of a higher hierarchical level than the replicators. As discussed earlier, in artificial evolution the UoS do not have to be part of the lowest hierarchical level and in many cases are actually part of the same level as the UoE. Also, as we will show in section 5, when we use our framework to analyse evolutionary experiments, in many cases the UoS are not actually the 'genes' of the simulation. This means that calling the unit of evaluation a 'vehicle' for the genes would be incorrect in many cases. Limiting the UoS to

genes also eliminates the possibility of group selection based evolutionary algorithms. For these reasons we see our framework as extending and improving on the concept of vehicles and replicators in the artificial evolution world by understanding that the gene-centric view of evolution does not work well when analysing artificial evolution because of the flexibility the experimenter has in varying the UoE and UoS.

3.4 Opening up the Space of Potential Algorithms

One of the benefits of analysing GAs in terms of UoE and UoS is that it can open up the space of potential algorithms that can be applied to a task. To demonstrate this we apply UoE and UoS to two related papers by Floreano et al. (2008) and Waibel et al. (2009) that test four evolutionary algorithms which they claim differ in terms of the type of team used to solve the task and the level of selection used.

The four different types of evolution Floreano et al. (2008) and Waibel et al. (2009) test are (using their definitions): (1) Homogeneous teams evolved with individual-level selection (2) Heterogeneous teams evolved with individual-level selection (3) Homogeneous teams evolved with team-level selection and (4) Heterogeneous teams evolved with team-level selection. These four evolutionary algorithms were tested on robot tasks that can be solved individually, tasks that require mutually beneficial cooperation and also tasks that are best solved with altruistic cooperation. Floreano et al. (2008) describe these four evolutionary methods as follows:

> The level of selection is varied by either measuring team performance and selecting teams (team-level selection) or measuring individual performance and selecting individuals independently of their team affiliation (individual-level selection)

This quotation follows the common practice of failing to distinguish between evaluation and selection, the distinction we wish to highlight. Using our view of evolution, we will show that because evaluation and selection are not thought of separately the level of evaluation and the level of selection are the same for each of the algorithms.

In table 3.1 their four different methods of evolving robots are classified using UoE and UoS. This shows that according to our view of evolution, the algorithms Floreano et al. (2008) and Waibel et al. (2009) use are limited to having the UoE the same as the UoS.

For example, when they evolve heterogeneous teams using team-level selection, the fitness of each team is calculated as the sum of the fitnesses of the individual robots,

Table 3.1: Classifying the four GAs in Floreano et al. (2008) and Waibel et al. (2009) using units of evaluation (UoE) and units of selection (UoS). Using this view of evolution the UoS and the UoE are the same in all the algorithms.

	Homogeneous Ind Selection	Homogeneous Team Selection	Heterogeneous Ind Selection	Heterogeneous Team Selection
UoE	Individual	Team	Individual	Team
UoS	Individual	Team	Individual	Team

and then randomly chosen robots from the fittest teams are picked to be parents for the next generation (Waibel et al., 2009). For the reasons explained earlier (see battery hen examples) both the UoE and UoS are a team of robots because the team is being evaluated and then selection is being driven by the fitness differential of the teams. Distinguishing between evaluation and selection would have allowed algorithms where the UoE is the group and the UoS is the individual to be tested, hence opening up the space of potential algorithms. An example of this type of GA is the Binomics GA (Harvey and Tomko, 2010), which we discuss in chapter 5.

Floreano et al. (2008) found that team-level selection of homogeneous teams outperformed the other algorithms on the tasks that were setup to best be solved altruistically. They believe the reason it outperformed the heterogeneous team-level selection method was because in the latter, random individuals from the best performing teams, were mated with individuals from other high performing teams at every generation. In other words, this algorithm limited the evolution of cooperative solutions by reorganising the teams every generation. As we discussed earlier in this chapter, there are many ways to select the UoS based on the assigned fitness credits (step 3). Floreano et al. (2008) have chosen to pick random individuals from the best performing teams rather than to select the fittest teams to reproduce as a unit like the Group GA algorithm (Tomko et al., 2011, 2012) that we will present in chapter 4. One of the benefits of using our view of evolution is that it makes identifying these subtle differences between algorithms easier which potentially could lead to improved algorithms.

Floreano et al. (2008) also believe the evolution of homogeneous teams using team-level selection benefits from the fact that it does not require...

> ...the need for separately computing the individual performance of each individual in a team. This is particularly useful in robotic tasks where only the resulting work of the team is known, but not what each robot in the team did

and how.

According to our view of evolution, the UoE of both the homogeneous team-selection and the heterogeneous team-selection algorithms are teams of robots and therefore the only reason to calculate the individual fitness scores is if they are required to calculate the total team score (for an in depth discussion of the pros and cons of different evolutionary strategies for evolving homogeneous teams of robots see Chapter 3 of Quinn, 2006). In our opinion the need to calculate individual fitness scores should not be related to the type of team (homogeneous or heterogeneous), but instead depends on how the fitness of the UoE is determined.

3.5 Summary

The key message from this chapter is that to get the full picture of artificial evolution, evaluation and selection need be viewed separately. As we have just shown, doing this allows existing evolutionary algorithms to be consistently analysed which could help experimenters come up with new ways of applying artificial evolution. The remainder of this thesis presents a number of novel algorithms that were developed using this view of evolution and uses these algorithms to study the effects of varying the relationship between evaluation and selection.

In summary, any evolutionary algorithm or experiment can be described using UoE and UoS in the five following steps.

1. Evaluate each of the UoE according to the pre-defined fitness function
2. Assign fitness credits from the UoE to UoS
3. Select the UoS on the basis of the assigned credits
4. Use the selected UoS to produce the next generation of offspring
5. Use the offspring to generate new UoE

In standard GAs the UoE = UoS = offspring which means that steps 2 and 5 are ignored. It is also common for both the UoE and UoS to be individual entities in the population. By drawing a distinction between evaluation and selection not only is the experimenter forced to think about the different hierarchical levels in their model but also the interaction between evaluation and selection (steps 2 and 5).

The new GAs we present in the remainder of this thesis will focus on steps 1, 2 and 5 of the above evolutionary cycle because we believe it is these steps that are for the most part ignored when evolution is not viewed in terms of both evaluation and selection. In Experiments I of this thesis we focus on how varying the relationship between the UoE and UoS impacts evolution. In Chapter 4 we present the Group GA which increases the level of both the UoE and UoS from the individual to group, and show how this change allows evolution to solve problems that require a division of labour between component parts. Chapter 5 explores the effects of keeping the UoE at the group level but reducing the UoS to the individual level. One of the issues of setting the UoE to the group level is that the group size parameter needs to be optimised in order for evolution to work. In Chapter 6 we present a method of evolving the group which eliminates the need to pre-set this parameter ahead of time.

In Experiments II (Chapters 7 and 8) of this thesis focuses on the latter half of the evolutionary cycle, investigating how changing the method of constructing new UoE can improve evolution on certain tasks. We find that generating new UoE using massive amounts of horizontal gene transfer can lead to improved performance on tasks where there is a high amount of genetic epistasis.

Experiments I: Varying the Levels of Evaluation and Selection

In Experiments I of this thesis we investigate how varying UoE and UoS affects evolution. Chapter 4 explores the benefits of increasing both the UoE and UoS to the group level and is based on work done in Tomko et al. (2011). In Chapter 5 the Binomics GA Harvey and Tomko (2010) effects of varying the UoS between the individual and group while keeping the UoE at the group level are investigated. Finally, in chapter 6 a method to evolve the optimal group size in symbiotic GAs, presented first in Tomko et al. (2012), is expanded on. Expanded tables of results for the experiments done in this part of the thesis are included online at ntomko.wordpress.com.

Chapter 4

Changing the UoE: The Group GA

4.1 Introduction

Standard GAs, where both UoE and UoS are individual members of the population, work well when the task is such that a single individual can solve it on its own. But they may run into problems on multi-role tasks where different individuals are required to cooperate in different roles. In this chapter we will show that changing the UoE and UoS from an individual to a group allows evolution to solve tasks that require the population to niche, in which different individuals are performing different jobs. This is done using the Group GA (Tomko et al., 2011, 2012). In the Group GA both the UoE and UoS are groups of population members, which means that groups are selected based on differential group fitness. In this GA the relationship between the UoE and UoS is the same as a standard GA, i.e. UoE = UoS, but the actual UoE and UoS are groups, instead of individuals.

We show that when we make both the UoE and UoS groups of population members evolution is able to cause speciation and niching in the population. Here we use these terms to broadly described the evolutionary process by which a single type of organism differentiates into multiple 'specialised' types, that for instance, take advantage of different resources available in a given environment. An example of this is the immune system, where groups of different types of antibodies have different roles and responsibilities all of which contribute to the goal of keeping the body healthy and free from diseases.

For a GA to be able to find symbiotic solutions to problems, where the individual cannot solve the task on its own, it must have the following characteristics: (1) It must be able to maintain diversity within the population so that niches can form and (2) it must

allow for fitness to be evaluated at the group level. Evolutionary niching methods such as those summarised in Dick (2005) and Mahfoud (1995) solve problem (1) by enforcing diversity in standard GAs so that a single population can be split up into n different niches. One of the issues with some of the more common niching methods is that they require prior knowledge about the specific fitness landscape in order to work; for example, whether n is 2, 5 or some other number. One of the benefits that the Group GA has over some of these existing methods is that it can accomplish niching with minimal *a priori* knowledge of the fitness landscape, and without knowing how the different jobs should be shared out.

There are two main goals of this chapter. The first is to use the Group GA to illustrate the effects of changing the UoE and UoS from the individual to the group level and the second is to highlight the advantages it has over similar algorithms. We demonstrate the emergent niching ability of the Group GA on an artificial immune system matching task that has been previously been used in Forrest et al. (1993) and Potter and De Jong (2000). The goal of this task is to evolve a population of antibodies (protecting agents) to match a set of antigens (harmful invaders). To solve this task the population of antibodies needs to niche so that it contains different individuals that match different antigens. One reason this task was chosen is because the number of peaks in the fitness landscape can be changed by changing the number of antigens that the population of antibodies needs to match. The other reason for choosing this task is that it makes it very easy to determine when niching has occurred. It should be noted that the main purpose of this chapter is to show the general feasibility of GAs with group-level evaluation. In chapter 5 we compare the Group GA to other algorithms as well as investigate its sensitivity to different parameter settings on different tasks.

The remainder of this chapter is structured as follows. In the next section we describe the Group GA in detail and compare it to existing symbiotic GAs. Next we introduce our version of the immune system task and describe how the Group GA can be used to solve it. Finally we present the results of the Group GA on the immune system task and show how it can handle the situation where the number of antibodies change during evolution. Expanded results for all these tests and experiments can be found online at ntomko.wordpress.com.

4.2 Tasks Used in this Chapter

After the introduction sections of each of the remaining chapters there will be a section entitled 'Tasks Used in this Chapter' where the tasks are described. If the task(s) have been introduced in a previous chapter the appropriate sections will be referenced, if not they will be described in this section. We have chosen to structure each chapter in this way so that the reader can easily find where all of the tasks are explained.

4.2.1 The Immune System Task

In this chapter we test the Group GA on an artificial immune system matching task to demonstrate the emergent niching abilities of the Group GA that is presented below. This task, which has previously been used in Forrest et al. (1993) and Potter and De Jong (2000) was chosen because it can be solved cooperatively and clearly illustrates how the Group GA can lead to emergent niching and how it can adapt to a changing fitness landscape, both of which are difficult with a conventional GA. In Forrest et al. (1993) this task was used to study adaptation in the immune system and in Potter and De Jong (2000) different variations of this task were solved using cooperative co-evolution. We will compare the results of these two papers to the Group GA later in the chapter.

The goal of this task is to evolve a population of antibodies to protect the body from a set of antigens. Simply speaking, antigens can be thought of as bacteria, viruses or other pathogens that are dangerous to the body and the antibodies can be thought of as the guardians which target these antigens for removal. This is a group task since defense against only a subset of the antigens is insufficient; the entire set of attacking antigens needs to be targeted. Antibodies in natural immune systems need to be adaptive in order to combat new and different antigens that may enter the body. Therefore this task tries to mimic this challenge of natural immune systems on a very basic level by attempting to evolve a population of artificial antibodies to match a set of antigens that changes over time.

In this task both the antibodies and antigens are modeled as bit strings. How well an antibody combats a specific antigen is calculated as the number of bit matches between antibody and antigen. For example a [1 0 1 1] antibody matches a [0 0 1 0] antigen at location two and three and therefore the antibody's fitness is equal to two when matched to this antigen; the higher the match (fitness) score the better.

Assuming that the length of the antibodies and antigens is the same, when there is more than one antigen in the antigen set the task can be thought of as symbiotic, because

it is impossible for a single antibody to match an entire set of antigens on its own. In this case, the population of antibodies needs to evolve to contain specialists to combat each different antigen. Obviously the more antigens there are, the more difficult the task becomes, because the evolving antibody population requires a larger number of specialists.

When we apply the Group GA (and the Binomics GA in a later chapter) to the immune system task, the fitness of a group of antibodies is calculated as the average of the best match scores achieved against all the antigens in the set. In other words, to evaluate a group of antibodies, all the antibodies in the group are matched against every antigen in the set and the average of the highest match scores against each antigen is the group fitness. This means that to get a perfect fitness score there has to be at least one antibody that matches each antigen perfectly in the group. For example, to calculate the fitness of a group of antibodies represented by binary strings of [0 1 0 0], [0 0 0 0] and [1 0 1 0] against a set of antigens represented by binary strings [1 1 1 1] and [0 0 0 0] the following is done:

1. Calculate the match score of every antibody against each antigen in the set. In this case the match scores of the antibodies against the [1 1 1 1] and [0 0 0 0] antigens are 1, 0, 2 and 3, 4, 2 respectively.

2. Take the average of the best match scores for each antigen. Here the best match score for the [1 1 1 1] antigen is 2 and for the [0 0 0 0] antigen the best match score is 4. Taking the average of 2 and 4 gives us a group fitness of 3.

4.3 The Group GA

The Group GA is a steady-state GA that uses tournament selection and is based on the Microbial GA (Harvey, 1996, 2001, 2011) described in section 2.1.7 A single cycle (tournament) of the Group GA can be broken up into the following steps (see figure 4.1):

1. Randomly choose two groups of individuals. Each group is constructed by randomly choosing n different individuals from the population with replacement. This means that the same individual can be in both groups but each individual within a single group is distinct.

2. Calculate and assign a fitness score to each group of population members based on the group's performance on a given task.

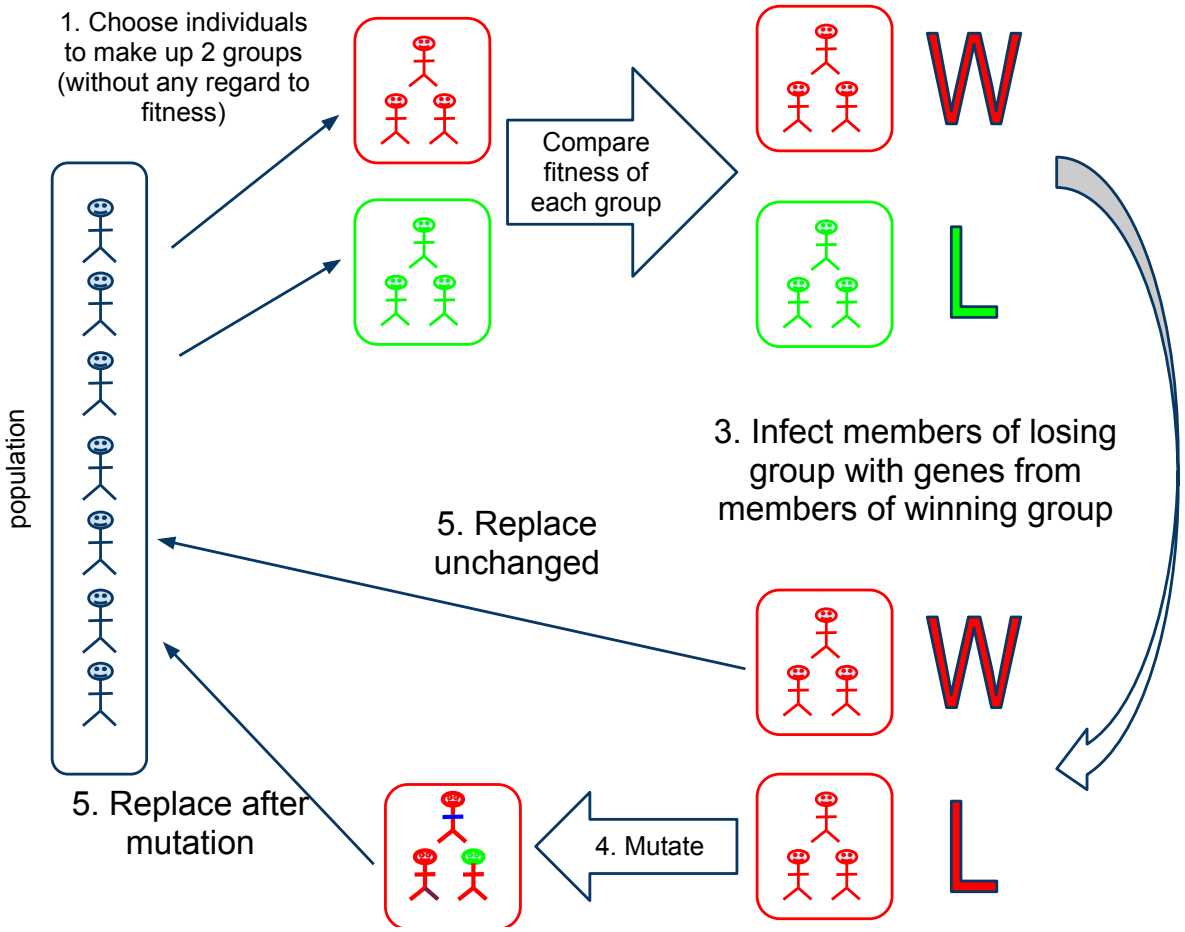

Figure 4.1: A sketch of the main steps of the Group GA.

3. The members of the less fit group are infected with genes from the individuals of the fitter group by randomly pairing individuals from each group and then overwriting some proportion of the genes from individual in the losing group with the genes from the individual in the winning group. The proportion of genes transferred is set as a parameter that can be varied between 0.0 and 1.0. In this chapter we set this parameter to 1.0 which means that all of the genes of the individual from the winning group replace the genes of the individual from the losing group. With this parameter setting 'infection' is equivalent to standard replication plus replacement which is similar to many other steady-state GAs. In later chapters we explore the effects of varying this infection parameter.

4. The genes of the individuals of the less fit group are mutated. This was done using a mutation rate of 0.1 per genotype, meaning that at each locus there was a probability of 1/640 of flipping that bit when using bit strings of length 64.

5. This process is repeated until some pre-defined stopping condition is met.

What differentiates the Group GA from more conventional GAs is that groups of population members (of some group size that is a parameter of the GA) rather than individual population members (as in conventional GAs) are evaluated and then selected based on the overall fitness of the group. Hence the driver of fitness based selection is the relative fitness of an entire group of population members that work together as a unit to solve some task.

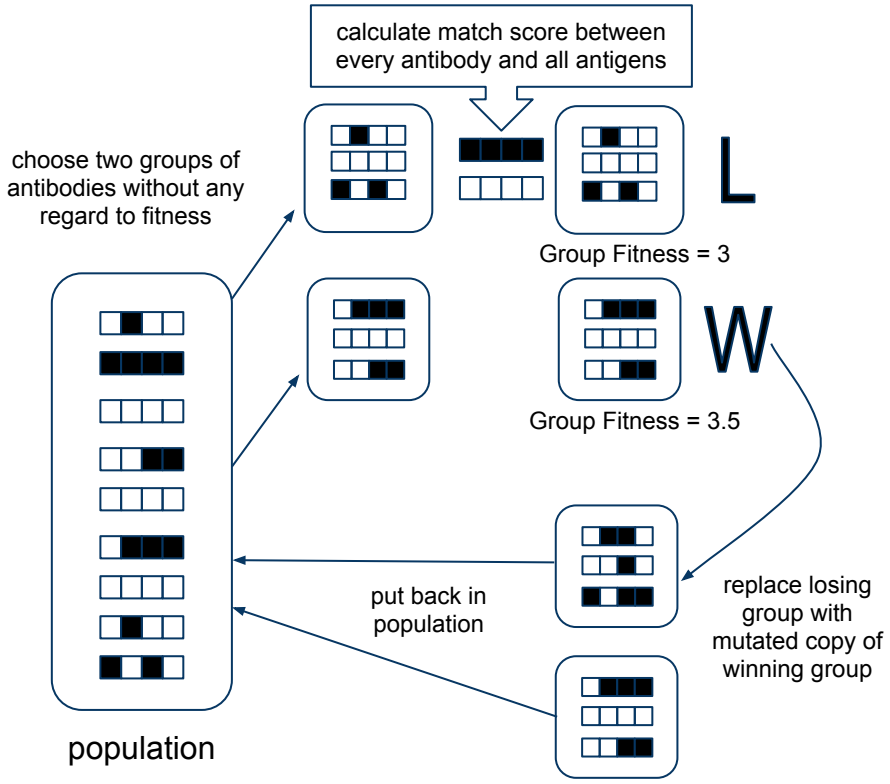

Figure 4.2: The Group GA as applied to an immune system task with two, 4-bit antigens. To evaluate a group of antibodies, all the antibodies in the group are matched against every antigen in the set and the average of the highest match scores against each antigen is the group fitness. For example, the top group of antibodies has a fitness score of 3 because the highest match score against the black antigen is 2 and the highest match score against the white antigen is 4 (Tomko et al., 2012).

Figure 4.2 illustrates how the Group GA is applied to the immune system task. Here two groups of antibodies are chosen from the population, evaluated against the antigens and then a mutated copy of the winning group overwrites the losing group. Although on this task one can calculate a measure of fitness for an individual, more generally the Group GA can be applied to tasks where individual fitness is meaningless because the Group GA randomly selects two groups of population members and uses them to construct

two higher level entities that are evaluated and assigned a fitness score. How fitness is calculated depends on what type of problem is being solved, but regardless of this, it is only the group fitness that matters when determining the tournament winner and loser.

4.4 Results

Here we show how, using the Group GA, a randomly initialised population of antibodies can be evolved to match a set of antigens. In the first experiment we evolve a population of antibodies to match a *fixed* set of four different antigens. This is equivalent to the Group GA solving a four-peaked fitness landscape. Then in the second experiment we evolve a population of antibodies to match a *variable* set of antigens, where antigens are added and removed during evolution. This second experiment simulates a task where the number of fitness peaks changes during evolution. In all the experiments in this section the antigen and antibodies were 64-bit binary strings, the antibody population size was 100 and the number of antibodies per group was 10.

4.4.1 Evolving Antibodies using the Group GA

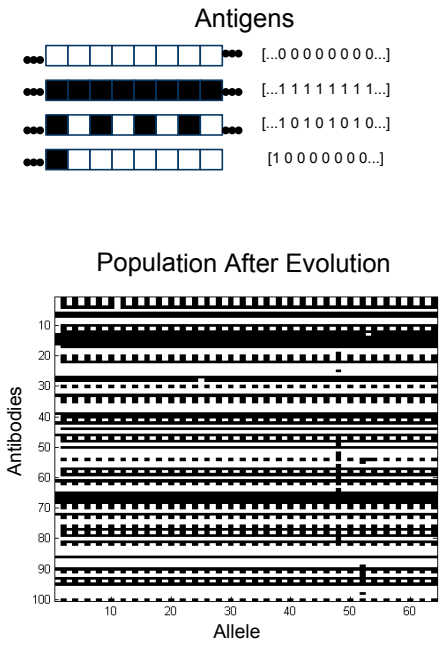

Figure 4.3: The antibody population after evolution on a 4 antigen task. The y-axis shows all 100 antibodies and the x-axis shows the value of each allele (gene) for a specific individual. Black corresponds to a gene value of 1 while white correspond to a gene value of 0.

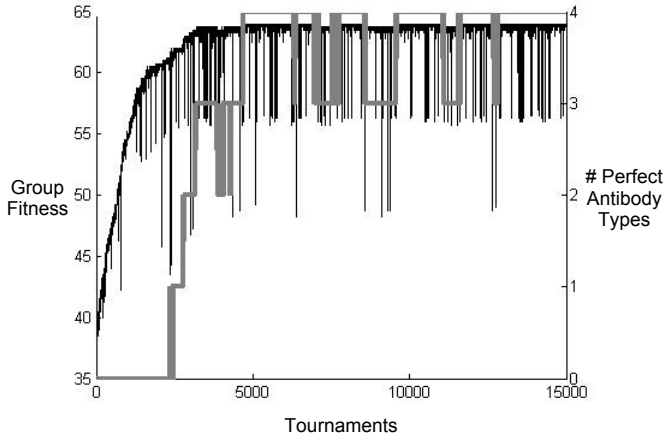

Figure 4.4: A plot of group fitness (black line) and 'population fitness' (gray line) over time for a single typical run of the 4 antigen task. The 'population fitness is the number of antigens covered perfectly by at least one antibody from the population as a whole.

Figure 4.3 shows how the antibody population niched after being evolved for 20K tournaments on a four antigen task. The four antigens used in this experiment were: [...0 0 0 0...], [...1 1 1 1...], [...1 0 1 0 ...], and [1 0 0 0...]. The first three antigens are specified by repeating these four bit patterns sixteen times to make up the full 64 bits antigen and the fourth antigen consists of a single '1' followed by 63 '0s'. The lower part of figure 4.3 (as with the similar plots in later figures) displays each binary genotype in the population horizontally above the next genotype, with white and black representing 0 and 1 alleles respectively. Over 50 runs there were an average of 12 perfect [...0 0 0 0...] antigens, 15 perfect [...1 1 1 1...] antigens, 15 perfect [...1 0 1 0...] antigens, and 11 perfect [1 0 0 0...] antigens at the end of evolution.

Figure 4.4 is a fitness versus time plot for this a single typical run of the four antigen task. The black line shows the group fitness of the tournament winning group of antibodies at each tournament, calculated as described above and the gray line shows the number of antigens covered perfectly by at least one antibody (from the whole population) at each tournament - this number can range from zero to the total number of antigens in the set. We believe that this 'population fitness' is an important measure of performance on this task because if you think of the goal of the antibodies in terms of protecting a body from invasion, then it is important that the population contains at least one antibody to match each antigen. In this figure you can see that throughout evolution the group fitness drops significantly for a tournament or two without decreasing the fitness of the population as a whole (number of perfect antibody types).

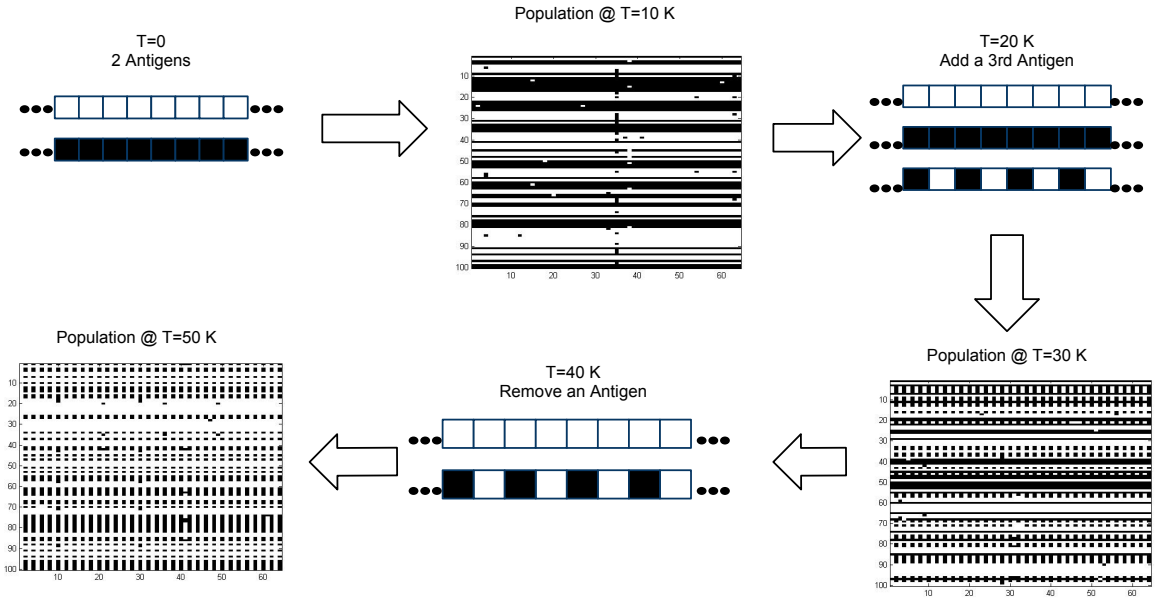

Figure 4.5: This figure shows how the antibody population adapts during evolution when 64 bit antigens are added and removed (T=10 K corresponds to tournament 10,000).

We also attempted to solve the four antigen task using the Microbial GA to show how standard GAs would struggle. As expected, when we used the Microbial GA, for all 50 runs the population always converged to match a single antigen in the set, failing to match the other three. As discussed later there are different ways that the Microbial GA can be modified so that it is able to solve the task but in these cases the UoE still needs to be a group of antibodies.

4.4.2 Changing the Number of Antibodies During Evolution

Figure 4.5 shows how the antibody population adapts when antigens are added and removed during evolution. In this experiment, the antigen set initially contained only two antigens [...0 0 0 0...] and [...1 1 1 1...]. At tournament 20K a third antigen [...1 0 1 0...] was added and evolution was resumed. At tournament 40K evolution was paused again and the [...1 1 1 1...] antigen was removed from the set before evolution was restarted. This figure clearly shows that when the antibody population is evolved using the Group GA the population can adapt to changes in the antigen set, adding and removing different types of antibodies as appropriate. Figure 4.6 shows the fitness versus time plot for this a single typical run of this task, in which antigens are added and removed during evolution. As this figure shows, when an antigen is added, the fitness of the population drops, then

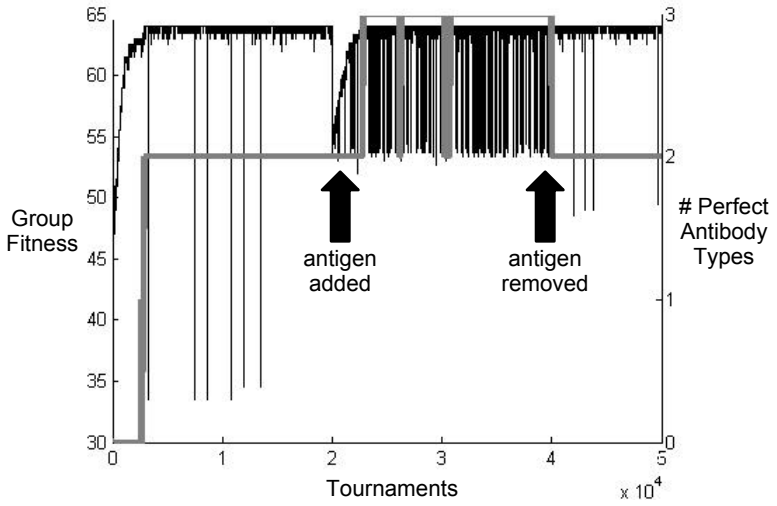

Figure 4.6: A plot of group fitness (black line) and number of antigens perfectly matched by at least one antibody (gray line) in the population over time for a single typical run of the task where antigens are added and removed during evolution (Tomko et al., 2011).

quickly recovers as the population adapts to match this new invader [1]. Obviously there is a limit to the number of antigens that can be matched by the population, this is discussed further in the next section.

4.5 Discussion

We have shown that changing the UoE and UoS from the individual level to the group level allows evolution to solve tasks that require a division of labour between component parts, something that is not possible using a standard GA. This was demonstrated using the Group GA to solve a multi-peaked artificial immune system matching task. Our results show that by evolving a population of antibodies using the Group GA, the population niches to match multiple antigens. We have also shown that when antigens are added and removed during evolution, the Group GA allows the antibody population to adapt to this change, and match new antigens that are presented.

Unsurprisingly, the Microbial GA, where individual antibodies are selected and evaluated, was unable to solve the multi-antigen task and ended up converging to match a single antigen every run. Compared to the niching methods described in the literature review, the Group GA has the following two advantages: (1) niching is accomplished emergently

[1] There are potential similarities between the adaptive mechanism of the Group GA and clonal selection (Burnet, 1959) that could provide a further avenue for exploration.

without having to know the appropriate niches ahead of time or pre-setting any parameter such as niching radius and; (2) fitness is evaluated at the group level which means that the Group GA can be used to solve symbiotic tasks where fitness is meaningless at the individual level.

For example, in cooperative coevolution (Husbands and Mill, 1991; Potter and De Jong, 1994) the number of species (or niches) needs to be set before evolution begins. Potter and De Jong (2000) solved the same immune system tasks using cooperative coevolution where the population was subdivided into n different species before evolution was started. This method was successful at evolving a population of antibodies to match different antigens as long as the number of different antigens was known *a priori* and the number of antigens remained constant throughout evolution. To overcome these limitations of cooperative coevolution, Potter and De Jong (2000) applied an evolutionary stagnation measure to determine when a new sub-population should be added. This allows antibody species to be added and removed during evolution in response to new antigens, but as Potter and De Jong (2000) state, the level of stagnation at which species should be added or destroyed is task dependent.

This task was also solved by Forrest et al. (1993) using a GA with a best-match fitness scoring scheme. In their algorithm, an antigen is chosen at random and matched against a group of antibodies from the population. Only the antibody in the group with the highest match score gets its fitness increased by its match score, the fitness of all other antibodies remains unchanged. This fitness evaluation step is repeated many times and then the population is evolved using a standard GA. Like the Group GA, this method allows the antibody population to niche to match a set of antigens without needing to know *a priori* how many antigens are present. The major difference between this method and the Group GA is that this best match method requires that the fitness of individual population members can be evaluated on their own. This is possible for this task because each individual antigen can be evaluated on its own by matching it against a single antigen, but for tasks where fitness can only be evaluated at the collective or group level the best-match method will not converge to an appropriate niched solution.

In general, other types of algorithms such as fitness sharing (Goldberg and Richardson, 1987) and crowding (De Jong, 1975) do not require knowledge about the exact number of niches but instead rely on a pre-set niching (similarity) radius or some sort of similarity calculation in order to get the population to niche. Another problem with some of these niching methods such as fitness sharing and crowding is that they are tailored for tasks

where each individual in the population can solve the task on its own or where the division of labour is known *a priori*, not for tasks where fitness can only be calculated at a group level or the optimal division of labour is unknown.

In general, the genetically based niching methods described earlier will struggle with this type of symbiotic task where individual fitness is meaningless. An example of this type of task is the evolution of artificial neural networks (ANN) task where the population is made up of partial sub-networks which have no fitness except when they are combined with other sub-networks to form a fully specified networks. In the next chapter we test the Group GA on an ANN task.

The Group GA is also similar to symbiotic GAs such as SANE (Moriarty and Miikkulainen, 1995, 1996), ecosystem evolution (Williams and Lenton, 2007) and SEAM (Watson and Pollack, 2000b), all of which evaluate groups of entities. One of the main differences between these algorithms and the Group GA is that they evaluate at the group level but select on the individual level so the UoE $\supset$ UoS. In the next chapter we will present the Binomics GA (Harvey and Tomko, 2010) where the UoE is the group and the UoS is the individual and compare it to the group GA to try to understand the differences.

There are many other potential ways of evolving solutions to this task. For example, instead of having each individual in the population being an individual antibody, a number of antibodies could be strung together so that each population member is a 'super antibody' that is able to match more than one antigen. If the number of antigens is known ahead of time and it is known that the number of antigens is not going to change during evolution then this could be a good way of solving the problem. Alternatively, one could make each super-antibody long so that one does not need to know how many antigens there are ahead of time. In both these cases the UoE is still a group of antibodies, so like the Group GA these methods should be able to evolve a niched population.

It is important to realise that modifying the Group GA so that the same antibodies are always in the same groups would be exactly the same as having a population of 'super antibodies' that can match more than one antigen. This would be equivalent to adding geographic structure (demes) to group construction and in some tasks, this may improve evolution. In the Group and Binomics GA presented in the next chapter, we have chosen to construct the group by randomly picking individuals from the population. We have not investigated how different methods of group construction impact evolution although this would be an interesting line of research and we would expect in some cases it would benefit evolution.

The idea of how to construct groups and the different ways of viewing groups versus individuals can also be thought of in terms of biological evolution. For example, viewing mammalian evolution solely in terms of its own genetic evolution could be equated to running standard GAs where the UoE and UoS are individuals. On the other hand, if one looks at the evolution of the entire mammalian metagenome, which includes both the genes of the given mammal and all the microbes that are an integral part of the animal, then evolution looks more like Group GA type evolution. To get a feel for the difference between the human genome and the human metagenome, one estimate is that there are ten times as many microbial cells in the human body than there are human cells and the human metagenome contains a hundred times more genes that the human genome (Qin et al., 2010). Obviously, there are many differences between natural and artificial evolution, but the point is that in natural evolution, depending on the goal, it will make sense to view evolution in specific ways. In artificial evolution depending on the task, different methods of group construction will be better than others. Our goal is that this view of evolution and the GAs presented opens up the field so that these interesting variations are tested and applied to real-world tasks.

Throughout this chapter, we have said that one of the advantages of the Group GA is that niching can be accomplished emergently without pre-setting any parameters. One could argue that the Group GA requires the group size to be pre-set so it is the same as existing algorithms where the niching parameter needs to be set ahead of time. In Chapter 6 we will show that the Group GA can actually solve the immune system task using a variety of different group sizes, unlike cooperative coevolution which requires the exact number of niches to be set ahead of time. We also demonstrate how the group size can be randomised or evolved during evolution, eliminating the need to pre-set this parameter at all.

We believe that the Group GA has the potential to be a useful algorithm that can use emergent niching to solve problems where the optimal division of labour is unknown. In the next chapter we continue to explore how varying the UoE and UoS impacts evolution by keeping the UoE at the group level but changing the UoS to the individual level.

Chapter 5

Changing the UoS: The Binomics GA

5.1 Introduction

In the previous chapter we showed how changing the UoE and UoS to the group level allows evolution to solve problems that require explicit niching and hence are not solvable using standard GAs. Now we explore the effects of varying the UoS while leaving the UoE at the group level. The questions we try to answer are: (1) Does changing the UoS make a difference to evolution? (2) Is the impact of changing the UoS the same for different tasks? and (3) Can we start to understand how to set the UoS to for different tasks?

To explore these questions we compare the Group GA, where the UoE and UoS are both groups, to the Binomics GA (Harvey and Tomko, 2010), where the UoE is a group and the UoS is an individual. First, these GAs are tested on the immune system task introduced in the last chapter to see what effect reducing the UoS has on a task that requires very explicit niching. Then we test both GAs on an artificial neural network (ANN) task that has been set-up to be symbiotic, meaning that it requires the UoE to be a group, but unlike the immune system, does not require the population to explicitly niche.

After comparing the Group and Binomics GA on these two tasks the Binomics GA is modified so that the number of individuals selected at each tournament may be varied. As we explain in section 5.6, this doesn't explicitly change the UoS from the individual to a group, it does allow us, however, to explore what happens when the amount of selection in the Binomics GA is increased, thus it is more like the Group GA. It also means that the optimal amount of selection can be found for each task, providing further data on how

changing the UoS impacts evolution. Finally, we compare both the Binomics and Group GAs to the Microbial GA, which is a standard steady-state GA with individual evaluation and selection.

Based on the results from comparing the Binomics, Group and modified Binomics GA by the end of this chapter we are able to make some recommendations regarding what the UoS should be set to for different tasks as well as discuss the limitations of both the Group and Binomics GA.

5.2 Tasks Used in this Chapter

The effect of varying the UoS is tested on the immune system task described in 4.2 and on an autoencoding artificial neural (ANN) network task described here.

Autoencoding (AE) neural networks (Hinton and Salakhutdinov, 2006) are ANNs with a feedforward succession of layers, potentially fully connected between each successive layer. When appropriate weights are found, they should reduce high-dimensional input data through a lower-dimensional bottleneck layer and then recover the input pattern and replicate it at the final output layer. Between input and bottleneck there is a hidden layer, which should encode the input pattern into the bottleneck; thereafter a further hidden layer should decode to the output.

We used AEs of the form N-h-M-h-N , where N is number of inputs/outputs, M is the size of the bottleneck layer, and h is the size of each hidden layer (see figure 5.1). In our implementation, all inputs were either 1 or -1. The hidden layer transfer functions were hyperbolic tangents, whereas the bottleneck transfer function was linear. The output layer transfer function was a discrete step function that mapped positive/negative values into +1/-1 respectively. For simplicity, no biases were used in any of the networks. The fitness of a network is calculated by testing every possible input pattern on the network and then counting the number of outputs that matched the inputs. For example, on a 3-12-2-12-3 AE there are three input neurons which means there are a total of eight (2^3) three bit inputs. When a specific network is evaluated all eight input patterns are shown to the network, and for each output that correctly matches the input the fitness score is increased by one, e.g. input 111 and output 101 scores 2. This means that the maximum fitness score of any 3-input AE is 24. In general, the maximum fitness score of an AE with N inputs is $N * 2^N$

The reason we chose to use AEs to test our algorithms is that they are simple to understand, the difficulty of the task can easily be increased by either increasing the

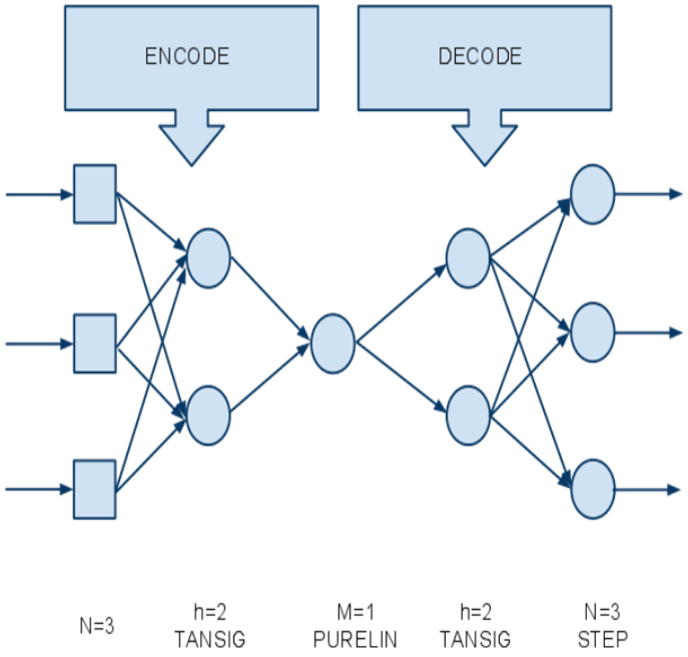

Figure 5.1: A sketch of a simple autoencoder that has 3 inputs and outputs (N=3), 2 nodes in the hidden layer (h=2) and a single node in the bottleneck layer (M=1). The transfer functions for each layer: hyperbolic tangent (TANSIG), linear (PURELIN) and discrete step (STEP) are listed at the bottom of the figure. When fully trained the output should replicate any binary input pattern.

number of inputs (N) or decreasing the size of the bottleneck layer (M). Since they can be evolved in a reasonable amount of time it is feasible to run a variety of different tests. Details of how the Group GA and Binomics GA can be used to evolve solutions to this task are in section 5.4.

5.3 The Binomics GA

The Binomics GA (Harvey and Tomko, 2010) is similar to the Group GA but instead of both the UoE and UoS being groups of population members, the UoE is a group of population members and the UoS is an individual population member; so UoE $\supset$ UoS. This means that individuals from the population are selected based on group fitness.

As discussed in section 3.2.2, when the UoE $\neq$ UoS then fitness needs to be assigned from the UoE to the UoS. In the Binomics GA the fitness of a UoE is equally passed to all of its constituent members. For example, if the fitness of the group is 5 then this fitness is used to update the fitness of every individual, on the basis of:

$$\text{NewIndFit} = \text{R} * \text{GroupFit} + (1.0 - \text{R}) * \text{OldIndFit}$$

OldIndFit and NewIndFit are used to refer to the previous fitness of the individual and the updated fitness of the individual respectively, while GroupFit is the fitness score that the group scored in the most recent evaluation. R, which can be set between zero and one, can be thought of as the leaky integrator term that weights the fitness score between the current and historical fitnesses. Setting R to 1.0 means that the individuals new fitness is solely based on the latest fitness evaluation, and the individuals historical fitness has no impact on its fitness. With an appropriate choice of R ($0.0<R<1.0$), the effective fitness of each individual is smoothed over several recent evaluations of different groups that it happens to have featured in, whilst still tracking any general changes in its environment. One way to think about the Binomics GA is in terms of evolving bacteria in a sea. The UoE can be thought of as a bucket of bacteria randomly drawn from the sea and the UoS are the individual bacterium that are selected based on the fitness of the bucket. The fitness smoothing parameter R determines how fitness is assigned from the bucket (group) to the individual.

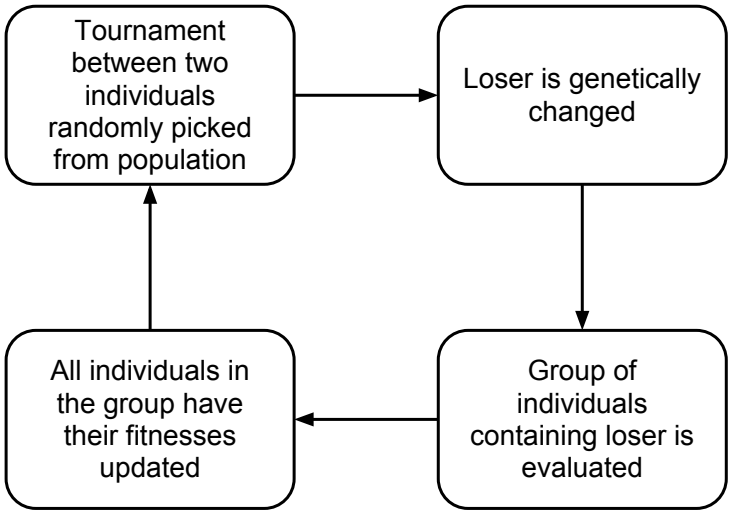

Figure 5.2: An illustration of the main steps of the Binomics GA.

The Binomics GA can be implemented in the following way (see figure 5.2). Initially, every individual in the population is given a fitness score of zero.

1. Randomly choose two individuals (the UoS) from the population and run a tourna-

ment to determine which is the fitter. If the fitnesses are equal a winner is randomly chosen.

2. Genetically modify the less fit individual (the loser) by infecting the loser with some proportion (usually 50%) of the winner's genes and then mutating the loser

3. Evaluate a group (UoE) of individuals from the population that contains the modified loser individual

4. Update the fitness of all individuals in the group and store these updated fitness scores

5. Repeat steps 1 to 4 until the stopping condition is reached

Like the Microbial GA described in section 2.1.7, genetic changes in the population are driven via tournaments involving selecting two individuals at random, comparing their currently stored fitnesses and designating one as the winner, the other as the loser. This is different from the Group GA where the tournament is between two groups of individuals randomly chosen from the population. In the group GA, the UoS is a group while in the Binomics and Microbial GAs the UoS is the individual.

5.4 How the GAs are Applied to the Tasks

In this chapter we compare the Binomics and Group GAs on both the AE and immune system tasks and describe how these GAs can be applied to each task. How the Group GA is applied to the immune system task was described in the previous chapter in section 4.3.

5.4.1 The Binomics GA Applied to the Autoencoder

The AE task was chosen because it can be set up as a symbiotic task where a group of individuals are required to solve the task. This can be done by having a population of partially specified networks that are combined into full networks. One way to do this is to start off with a population of fully-specified networks and then setting the majority of connection weights between nodes of each individual to zero. In effect, each individual in the population, therefore is a only a small part of the full ANN, and may very well be functionless on its own through having no connected path from inputs to outputs. Then, a full network can be constructed by randomly choosing a group of individuals (partial networks) from the population and summing (or averaging) the weights at each locus. The

proportion of zero weights in the population is pre-set at the beginning of evolution and is maintained using both an add-link and delete-link mutation operator.

Forcing each individual in the population to have some proportion of its weights be set to zero was done so that this AE task needs to be solved using group evolution. Obviously, the AE could be evolved using a standard GA if each individual in the population encoded a full network. However, the main goal of solving the AE in this way was to explore the effects of changing the UoS on evolution. In the next chapter, we will discuss some of the benefits of solving an ANN task in this way, including summarizing some recent research by Hinton and Srivastava (2012) that uses a weight zeroing-out method to improve performance of large feed-forward ANNs. This method which they call 'dropout' is used to randomly omit some of the neurons during training which has the benefit of reducing over fitting.

As applied to the AE, a single tournament of the Binomics GA is as follows:

1. Randomly choose two partial networks from the population and compare their stored fitnesses (initially every individual in the populaiton is given a stored fitness of zero)

2. Genetically modify the less fit network using infection and mutation

3. Randomly choose n other individual networks from the population to be combined with this tournament-losing network

4. Combine all these partial networks into a single full network by summing the weights at each locus

5. Evaluate this network on the task

6. The fitness of the network is used to update the stored fitnesses of all the individuals that made up the network according to: NewIndFit = R $*$ GroupFit + (1.0 $-$ R) $*$ OldIndFit

7. All partial networks are put back into the population and the cycle starting at step 1 is repeated.

8. The stopping condition occurs when a network in step 5 has a fitness score of $N * 2^N$

In the Binomics GA mutation occurs in two steps. First, standard mutation is applied by randomly choosing a single non-zero weight and mutating it by a normally distributed amount with mean zero and standard deviation which is set by a parameter. Then, the

number of zero and non-zero weights is modified so that each individual maintains the approximate ratio as set at the beginning of evolution. This is done as follows:

- Delete a non-zero weight (set it to zero) with probability:
  (Percent Zero)*(Num NonZero Wts)/(NumWts)

- Change a zero weight to non-zero (a small random number) with probability:
  (Percent Non Zero)*(NumZero Wts)/(NumWts)

For example, if we want each individual (partial network) to have approximately 90% zero weights then Percent Zero is set to 0.9 and Percent Non-Zero is set to 0.1.

In summary, each tournament takes two partial networks at random from the population, and determines a winner and loser based on their current fitnesses. The loser is modified using infection where some pre-specified proportion of the winner's genes overwrite the corresponding loser's genes before being mutated.

5.4.2 The Group GA Applied to the Autoencoder

To determine the effect of changing the UoS from the individual to the group the Binomics GA was compared to the Group GA on this task. As applied to the AE task, a single cycle of the Group GA is as follows:

1. Randomly choose two groups of partial networks from the population

2. Combine each group of partial networks into a full network by summing the weights at each locus

3. Calculate the fitness of each full network with the winning network being the one with the higher fitness

4. The group of partial networks that made up the losing network is infected with genes from the winning group and then mutated

5. Both groups of networks are put back into the population and this process is repeated

One difference between the Group and Binomics GA is that in the former, two networks have to be evaluated each cycle, while in the Binomics GA only a single evaluation needs to be made. For this reason comparisons are made using the number of evaluations it takes each algorithm to solve the task, rather than the number of tournaments.

5.4.3 The Binomics GA Applied to the Immune System

The Binomics GA can be applied to the immune system task as follows:

1. Randomly choose two antibodies from the population and compare their stored individual fitnesses.
2. The antibody with the lower fitness is genetically changed using infection and mutation.
3. This modified antibody is combined with a group of randomly chosen antibodies from the population.
4. All the antigens are matched against all the antibodies in the group.
5. The group fitness of this group of antibodies is calculated as the mean maximum match score in the group in the same way as it was in the Group GA (see section 4.2).
6. The fitness of all antibodies in the group is updated according to : New Antibody Fitness = $R * \text{Group Fitness Fitness} + (1.0 - R) * \text{Old Antibody Fitness Fitness}$
7. All antibodies are put back in the population and this cycle, starting a step 1 is repeated.

5.5 Results

In this section we present results from the following three experiments. First we demonstrate that changing how fitness is passed from the UoE to the UoS impacts the performance of the Binomics GA. Then we compare the performance of the Group to the Binomics GA on three AE tasks and the immune system task to understand how changing the UoS while keeping the UoE at the group level impacts performance. We find that on the 4-24-3-24-4 AE the Binomics GA outperforms the Group GA but on the 4-antigen immune system task the opposite is true.

5.5.1 Setting R

Before comparing the Binomics to the Group GA on both the AE and immune tasks we first want to show how the performance of the Binomics GA is impacted by changing the method of passing fitness from UoE to UoS. As discussed in section 3.2.2 anytime the

UoE $\neq$ UoS a decision needs to be made about how fitness of the UoE is assigned to the UoS. In the Binomics GA fitness is assigned equally from the UoE to all the UoS in the group before being time-smoothed over previous evaluations using the parameter R in the following way: NewIndFit = R $*$ GroupFit + (1.0 $-$ R) $*$ OldIndFit, where R (which can be thought of as a leaky integrator) is set between 0.0 and 1.0 To test how varying the method of passing fitness from UoE to UoS impacts performance in the Binomics GA we test the Binomics GA, with different R values on the 3-12-2-12-3 AE and the four antigen immune task.

For the AE task, each partial network in the population had 50% of its weights initialised to zero, and this ratio of zero to non-zero weights was approximately maintained throughout evolution. The population size was set to 50, the group size to 25 and the standard deviation of the mutation amount to 0.5. Performance was measured as the median number of evaluations it took to evolve a perfect AE (fitness score equals $N * 2^N$) over 50 runs, where evolution was stopped at a maximum of 20K evaluations.

R was also varied on the four, 64-bit antigen immune system task described in section 4.2. For this experiment we used the following parameters that were found to perform well based on a parameter sweep: the population size was set to 100, the group size to 10 and the mutation rate to 1/640. Performance was measured as the number of evaluations it took to evolve the first population that contained antibodies that perfectly matched all four antigens.

Figure 5.3 shows how varying R impacts performance on the AE and immune tasks respectively (full testing results can be found online at ntomko.wordpress.com). This figure, and all box and whisker plots used in this thesis are standard Matlab box and whisker plots where the red line shows the median of the data, the edges of the box show the 25th and 75th percentiles, the whiskers extend to show the most extreme data points not considered outliers and the red crosses show the outliers. An outlier is defined as those data points that are outside 1.5 times the difference between the 25th and 75th percentiles. These plots show that on the AE task optimal performance occurs when $R = 0.05$ and on the immune task the optimum is when $R = 0.75$.

When $R = 1.0$ the fitness of a population member is solely based on the most recent evaluation, its historical fitness doesn't factor into its current fitness. As R is decreased the current fitness is weighted more towards the historical fitness than the most recent evaluation. This means that on the AE task, the Binomics GA works better when the fitness of each partial network is more skewed to the past, while on the immune task it

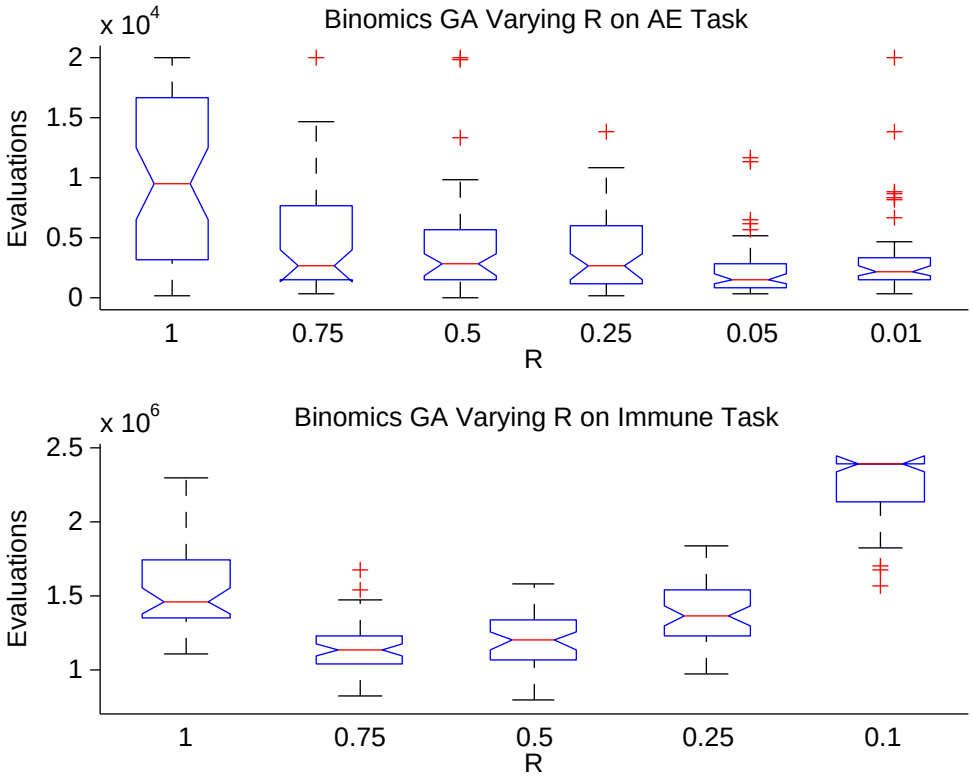

Figure 5.3: A box and whisker plot of the performance of the Binomics GA with different R (time smoothing parameter) values on the 3-12-2-12-3 AE task (top) and the 4 antigen, 64-bit immune system task (bottom). A full table of results for this figure can be found online at ntomko.wordpress.com .

works better when the fitness is based on the last couple of evaluations.

5.5.2 Comparing GAs on the Immune System Task

In the previous chapter we showed how the Group GA could be used to solve an artificial immune system task where the goal was to evolve a population of antibodies that matched a number of different antigens. We now test the Binomics GA on the same task and compare it to the Group GA.

The Binomics GA differs from the Group GA on this task in that the UoS is an individual antibody rather than a group of antibodies as it is in the Group GA. We compared the performances of the Group and Binomics GA on the 64 bit, 4 antigen task over 50 runs. For both GAs, the population size was set to 100, the number of antibodies per group was 10, the mutation rate was set to 0.1 per genotype, meaning that there was a probability of 1/640 of flipping a bit. On the Binomics GA the time smoothing parameter R was set to 0.75. These parameters were chosen because they were found to

Table 5.1: Comparison of the Binomics GA (BGA) and Group GA (GGA) on the 4 antigen immune system task. Performance was based on the number of evaluations it took to evolve a population that contained antibodies that perfectly matched all four antigens and statistical comparisons were done using p-values from the Wilcoxon rank sum test of equal medians and a Bayes factor over 50 runs.

	GGA Median(IQR)	BGA Median(IQR)	Bayes Factor	p-value
Immune Task	275 K (67 K)	1100 K (180 K)	$2 * 10^{-11}$	$7 * 10^{-18}$

be the optimal of those sampled in our parameter sweep.

Table 5.1 shows the performance of each GA on this task. Performance was based on the number of evaluations it took to evolve a population that contained antibodies that perfectly matched all four antigens. Again, statistical comparisons were made using p-values calculated using the Wilcoxon rank sum test of equal medians and Bayes factors (see Appendix B). A p-value of $7 * 10^{-18}$ and Bayes factor of $2 * 10^{-11}$ show that the null hypothesis can be rejected with high confidence and we can say that there is strong evidence that Group GA outperforms the Binomics GA on this task.

For the reasons discussed in Appendix B, all statistical comparisons in this thesis will be made using both Bayes Factors and p-values generated from the Wilcoxon rank-sum test of equal medians. As discussed in Appendix B the Bayes factors used in this thesis are in terms of the null hypothesis over the alternative hypothesis which means that small Bayes factors provide evidence in favour of the alternate hypothesis, which in the case of these comparisons is that the performance of the two GAs is not equal. In this thesis a result is considered significant if the p-value is less than 0.05 and if the Bayes factor is less than 1/3. Using a significance level of 5% for p-values is fairly standard and according to Jeffreys (1961) a good rule of thumb is that any Bayes factor less than about 1/3 provides strong evidence in favour of the alternate hypothesis. If the results of the two statistical measures contradict each other then it is assumed that there is not enough evidence to reject the null hypothesis.

Again, it should be noted that we are not using statistical tests in this thesis to claim any of these GAs are the optimal ways to solve any of these tasks. The main reason for using these statistical tests is so that we have some way to quantitatively measure how varying GAs using of our view of evolution impacts performance on different tasks.

5.5.3 Comparing GAs on the Autoencoder Task

The Group and Binomics GA are now compared on three different AEs: a 3-12-2-12-3 with 90% of the weights set to zero, a 3-12-2-12-3 with 50% of the weights set to zero and a 4-24-3-24-4 with 50% of the weights set to zero. One effect of increasing the proportion of zero weights in an AE is that it means that more individuals (partial networks) are required to work together to solve the task. Testing a 3-12-2-12-3 AE with both 50% and 90% zero weights allow us to see what impact this has on evolution. The 4-24-3-24-4 AE was also tested so that the effects of a more difficult landscape could also be explored.

To determine which GA performed better, on each AE a comprehensive parameter sweep was done and the best parameter combination of each GA was compared. In this parameter sweep the population size, group size, mutation amount, infection rate and R (for the Binomics GA) were all varied. Performance was measured in terms of median evaluations it took to evolve a perfect AE over 50 runs, so lower is better. If a perfect AE wasn't found in less than 20K evaluations for the 3-12-2-12-3 AEs and 40K evaluations for the 4-24-3-24-4 AEs then evolution was stopped and the run was deemed unsuccessful. It is important to highlight that the number of evaluations were being counted, not the number of tournaments and that in the Binomics GA there is one evaluation per tournament while in the Group GA there are two evaluations per tournament. The reason for comparing performance using evaluations rather than tournaments is that in most real-world tasks the evaluation step will be the most computationally intensive and time consuming.

Table 5.2 and figure 5.4 compares the performance of the best performing Group and Binomics GA on the three AEs. This table shows both the median and inter quartile range (IQR) evaluations it took to evolve a perfect AE over 50 runs as well as the results of the statistical comparisons. As these results show, on both 3-12-2-12-3 AEs there is not enough statistical evidence to reject the null hypothesis which is that both GAs perform equally well on this task. But on the 4-24-3-24-4 the null hypothesis can be rejected and therefore it can be said that there is strong evidence that the Binomics GA outperforms the Group GA on this task. Table 5.2 and figure 5.4 only show the performance of the best performing parameter combinations on each AE, the full results showing the data from the parameter sweep performed can be found online at ntomko.wordpress.com.

Comparing the Binomics GA to the Microbial GA

To get an idea of how the performance of the Binomics GA compares to a standard GA on the AE task we present the results from Harvey and Tomko (2010). Figures 5.5 and

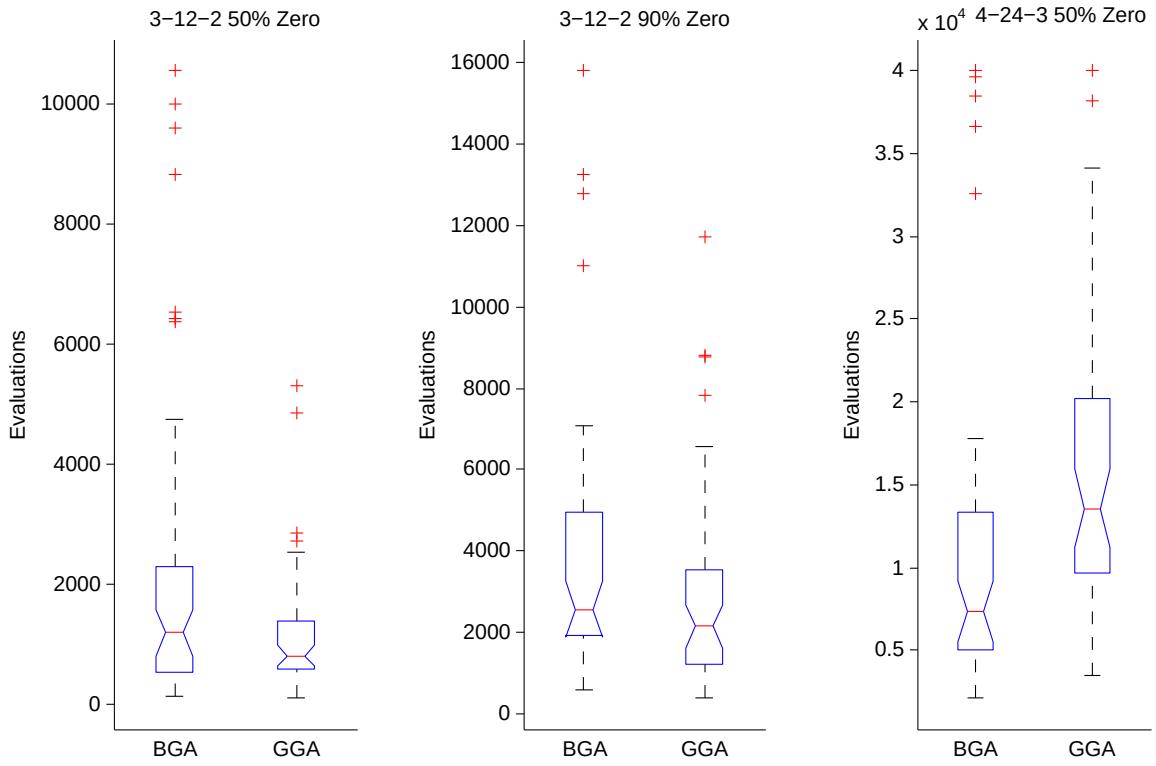

Figure 5.4: A box and whisker plot comparing the Binomics GA (BGA) to the Group GA (BGA) on the 3-12-2-12-3 AE with 50% zero weights and 90% zero weights and on the 4-24-3-24-4 AE with 50% zero weights.

5.6 which have been copied from Harvey and Tomko (2010) compare the Binomics GA to the Microbial GA on a 3-12-2-12-3 and 4-24-3-24-4 AE. These results show the Binomics GA outperforming the Microbial GA in terms of number of evaluations it took to evolve a perfect AE. As noted in the paper, these were preliminary tests performed to demonstrate in principle that the Binomics GA could be used to evolve ANN type tasks.

After Harvey and Tomko (2010) was published further research determined that the reason the Binomics GA outperformed the Microbial GA on these tests was due to how the weights were being scaled. In the Binomics GA, because a full network is constructed by summing up the weights at each locus of all the different partial networks in the group, there is an intrinsic scaling factor being added which significantly improved performance (see section 5.4.1). The degree of scaling is based on the group size, with larger group sizes leading to a higher scaling factor. Upon this discovery, we added a scaling factor to the Microbial GA and after optimising this new parameter, the Microbial GA outperformed the Binomics GA. Even though the optimised Microbial GA outperformed the optimised Binomics GA, this led us to investigate ways the group size could be evolved which would

Table 5.2: Comparison of the Binomics GA (BGA) and Group GA (GGA) on three different AE tasks: a 3-12-2-12-3 with 50% of the weights set to zero, a 3-12-2-12-3 with 90% and a 4-24-3-24-4 with 50% of the weights set to zero. Performance was measured in terms of median and IQR evaluations to evolve a perfect AE and statistical comparisons were done using p-values from the Wilcoxon rank sum test of equal medians and a Bayes factor.

Autoencoder	BGA Median(IQR)	GGA Median(IQR)	Bayes Factor	p-value
3-12-2 50% Zero	1196 (1659)	820 (795)	3	0.2
3-12-2 90% Zero	2573 (2891)	2148 (2252)	0.8	0.03
4-24-3 50% Zero	7385 (7938)	13573 (10297)	**0.009**	$\mathbf{1 * 10^{-4}}$

avoid the need to preset the scaling factor. These methods are presented in chapter 6.

5.5.4 Results Summary

In the this section we have shown that the Binomics outperforms the Group GA on a 4-24-3-24-4 AE but on the immune system task the Group GA outperforms the Binomics GA. We believe that this is due to the fact when the UoS is an individual, as in the Binomics GA, only a single population member is selected and modified for each evaluation but when the UoS is a group, as in the Group GA, an entire group of population members are selected and modified each evaluation. This means that the Group GA is a much more brutal algorithm than the Binomics GA and may be more suited to immune system type tasks where there is a very explicit division of labour. To further investigate how changing the amount of selection and mutation per evaluation impacts evolution in the next section we modify the Binomics GA so that the number of tournaments per evaluation can be set as a parameter.

5.6 The Modified Binomics GA

In the standard Binomics GA, which we have tested up to this point in the chapter, there is a single tournament per evaluation. In this tournament two individuals are randomly chosen from the population, their fitnesses are compared, and the winner infects the loser with some proportion of its genes before being mutated. This modified losing individual is then combined with a randomly chosen group of individuals from the population and evaluated on the given task. This means that in the standard Binomics GA the UoE is a

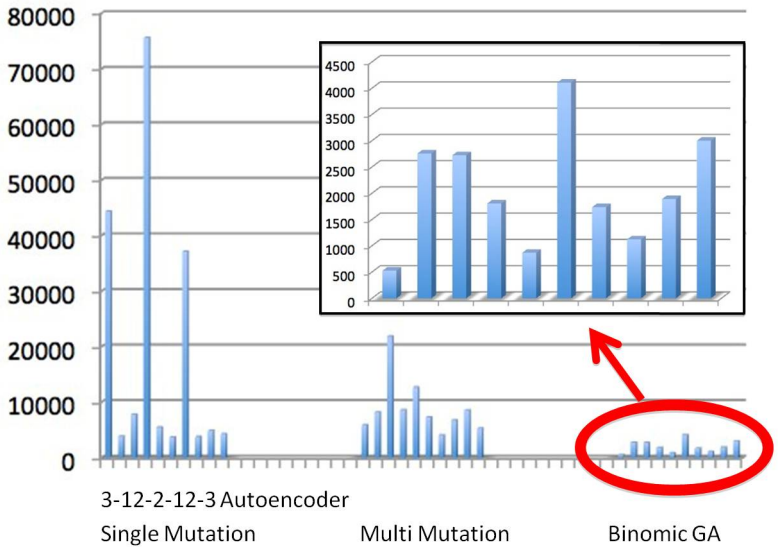

Figure 5.5: Number of evaluations needed to achieve a perfect score using 3 different GAs (10 runs each) on the 3-12-2-12-3 AE. The Microbial GA with single weight mutation, the Microbial GA with multi weight mutation and the Binomics GA. (this figure was copied from Harvey and Tomko, 2010)

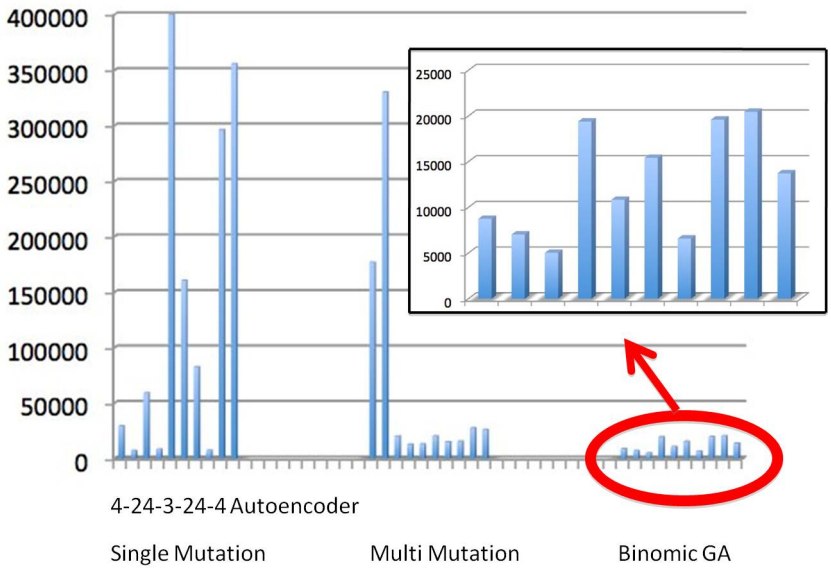

Figure 5.6: Number of evaluations needed to achieve a perfect score using 3 different GAs (10 runs each) on the 4-24-3-24-4 AE. The Microbial GA with single weight mutation, the Microbial GA with multi weight mutation and the Binomics GA. (this figure was copied from Harvey and Tomko, 2010)

group of population members and the UoS is an individual population member.

There is no reason why the number of tournaments per evaluation in the Binomics GA has to be limited to one, so in this section we test a modified version of the Binomics GA where the number of tournaments is set as a parameter that can be increased. A single cycle of this new, modified Binomics GA looks as follows:

1. Randomly choose two individuals from the population
2. Compare the stored fitnesses of the two individuals
3. The less fit individual is infected with genes from the fitter individual and then mutated
4. Repeat steps 1-3 NUMT times
5. The NUMT modified individuals are combined with a number of randomly chosen individuals equal to GROUPSIZE - NUMT from the population and evaluated on the task.
6. The fitness of all individuals in the group are updated by $\text{New Fitness} = \text{R} * \text{Network Fitness} + (1.0 - \text{R}) * \text{Old Fitness}$
7. All individuals in the group are put back into the population and we return to step 1
8. The stopping condition is when maximum fitness or maximum number of tournaments is reached

The only difference between the modified and original Binomics GA is that the number of tournaments per evaluation (NUMT) can be varied between one and the group size. This increases the number of individuals selected per evaluation but doesn't change the UoS. Even when NUMT = GROUPSIZE the UoS is still the individual. This is because at every tournament a single individual is being selected. This is different from the Group GA where at every tournament the entire 'fitter' group is being selected as a unit. At the end of this chapter we discuss whether varying NUMT is equivalent to changing the UoS.

We tested this modified Binomics GA on the three AE tasks and the 64-bit, four antigen immune system task. For each task we used a high performing set of parameters as found by our parameter sweep on the standard Binomics GA and varied the NUMT parameter. For the 3-12-2-12-3 and 4-24-3-24-4 AEs where 50% of the weights were set to zero, we tested the modified Binomics GA using population sizes of 100 and 500; for

Table 5.3: Comparison of the modified Binomics GA (mBGA) to the better of either the standard Binomics GA (BGA) or Group GA (GGA). Performance was measured in terms of median and IQR evaluations to evolve a perfect AE and statistical comparisons were done using p-values from the Wilcoxon rank sum test of equal medians and a Bayes factor.

Task	mBGA Median(IQR)	GGA or BGA	Median(IQR)	Bayes	p-value
3-12-2 50% Zero	919 (959)	GGA	820 (795)	4	0.3
3-12-2 90% Zero	1439 (1684)	GGA	2148 (2252)	0.02	$2 * 10^{-4}$
4-24-3 50% Zero	5285 (6935)	BGA	7385 (7938)	0.3	0.01
immune task	287K (71 K)	GGA	275K (67 K)	7	0.1

the 3-12-2-12-3, 90% zero weights AE we used a population of 500 and 1000; and for the immune system task, the population size was 100.

Figures 5.7, 5.8, 5.9, 5.10 show how performance on each of these tasks changes when NUMT is increased. The additional number of tournaments per evaluation in the modified Binomics GA are not factored into performance. The reason for calculating performance in this way is that in real world tasks, such as evolving robots, the cost of evaluation is going to be many orders of magnitude greater than the cost of running evolutionary tournaments and so the best GA will be one that can evolve an acceptable solution to a task in the minimum number of evaluations.

These figures show that on the AE tasks increasing NUMT up to a point improves performance when the population size is set to either 500 or 1000 but for the cases where the population was 100 increasing NUMT did not improve performance. For example, on the 3-12-2-12-3 AE with 90% zero weights, a population size of 500 and a group size of 250 the optimal NUMT was 5. On the immune system task, we can see that increasing NUMT to 5 or 10 significantly improved performance. On this task a population size of 100 and group size of 10 was used, so unlike the AE tasks, the optimal NUMT was approximately the same as the group size for the immune system task.

In table 5.3, we statistically compare the best modified Binomics GA with the better performing of either the standard Binomics GA or Group GA on each task. As these results demonstrate the modified Binomics GA is as good or better than the best Group or standard Binomics on every task. The parameter combinations used to generate these results are listed online at ntomko.wordpress.com.

Up to this point evolutionary performance on the immune system task has been meas-

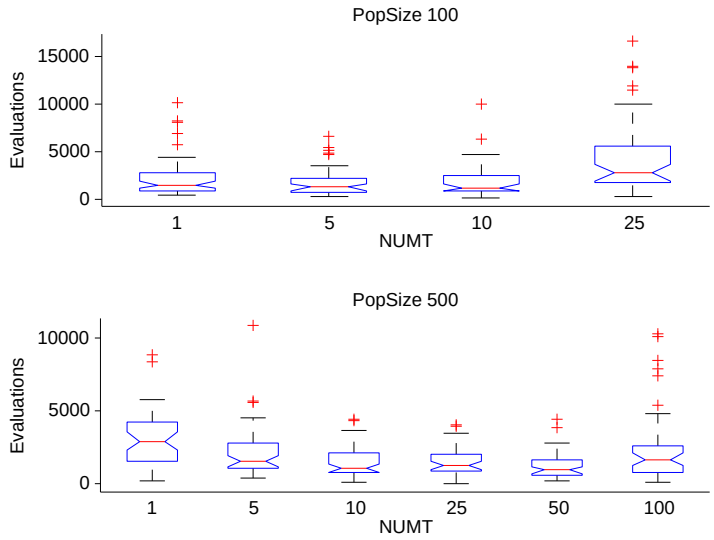

Figure 5.7: A box and whisker plot showing the effect of varying NUMT on the 3-12-2-12-3 task with 50% zero weights with a population size of 100 (top) and 500 (bottom). For both population sizes the group size was set to 50% of the size of the population.

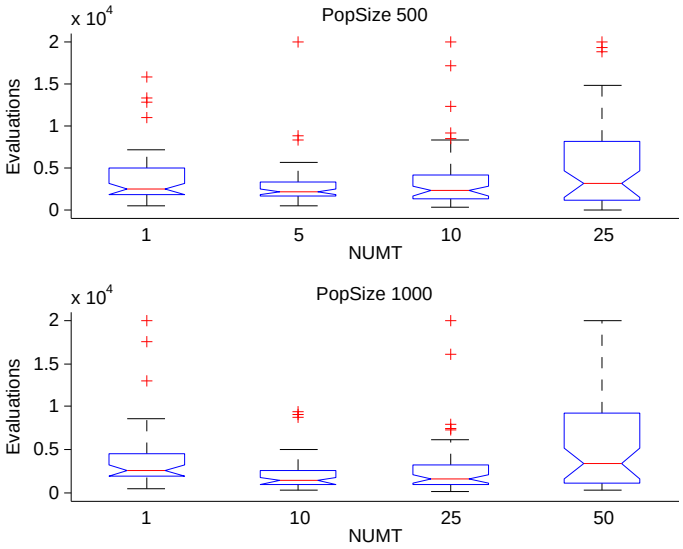

Figure 5.8: A box and whisker plot showing the effect of varying NUMT on the 3-12-2-12-3 task with 90% zero weights with a population size of 1000 (top) and 500 (bottom). For the population size 1000 case, the group size was set to 750, and for the population 500 case, the group size was set to 250.

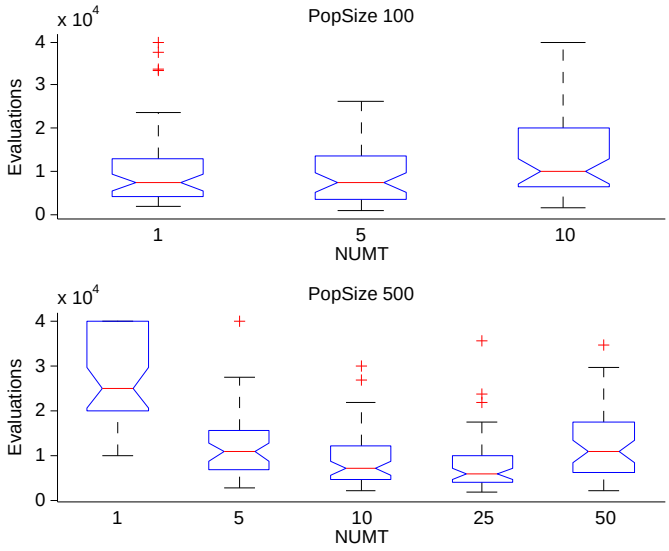

Figure 5.9: A box and whisker plot showing the effect of varying NUMT on the 4-24-3-24-4 task with 50% zero weights with a population size of 100 (top) and 500 (bottom). For the population 100 case, the group size was set to 50, and for the population 500 case, the group size was set to 100.

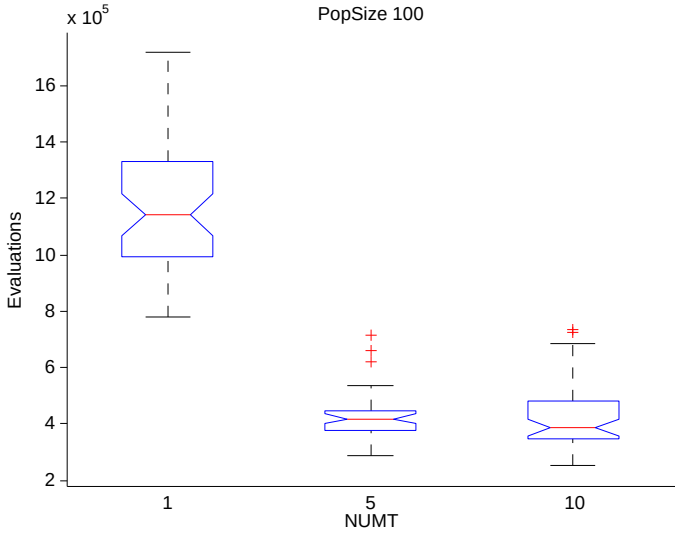

Figure 5.10: A box and whisker plot showing the effect of varying NUMT on the 4-antigen immune system task with a population size of 100 and a group size of 10. The GA was also run with NUMT=25 but because performance was so poor this data point is not shown on the plot.

ured in terms of the number of evaluations it takes to evolve the first population that contains four antibodies that perfectly match all four antigens. Another way to measure performance on this task is to run evolution for a fixed number of evaluations and determine whether at the end of evolution there are antibodies in the population that match all the antigens. This measures the ability of a GA to maintain niches throughout evolution. If we compare the modified Binomics GA to the Group GA in this way we find that the Group GA performs much better. For example, using the same parameters as were used in table 5.3 and running each GA for 2.4 million evaluations we found that the Group GA contained antibodies that matched all four antigens in 37/50 runs but the modified Binomics GA only contained antibodies to match all the antigens in 9/50 runs. So even though the number of evaluations it takes to evolve the first population that contains antibodies that match all of the antigens is about the same for both modified Binomics and Group GA, the Group GA is much better at maintaining these antibodies through evolution.

5.6.1 Modifying the Group GA

As was just shown, by increasing the number of tournaments per evaluation the Binomics GA becomes much more like the Group GA in terms of how much selection and mutation is occurring. Now we modify the Group GA so that it becomes more like the standard Binomics GA by reducing the number of individuals who are modified each tournament. In this modified Group GA a new parameter NUMMOD is introduced which specifies the number of individuals that are modified each tournament. This parameter can be set between 1 and the group size. When it is set equal to the group size this modified Group GA is identical to the original Group GA. With this new parameter, the Group GA is run as follows:

1. Randomly choose two groups of individuals from the population
2. Evaluate each group on the the task
3. Randomly pick NUMMOD individuals from the winning group to be recombined with a randomly chosen NUMMOD individuals from the losing group
4. Mutate the randomly chosen NUMMOD of individuals from the losing group that were just recombined
5. Put both groups of individuals back into the population

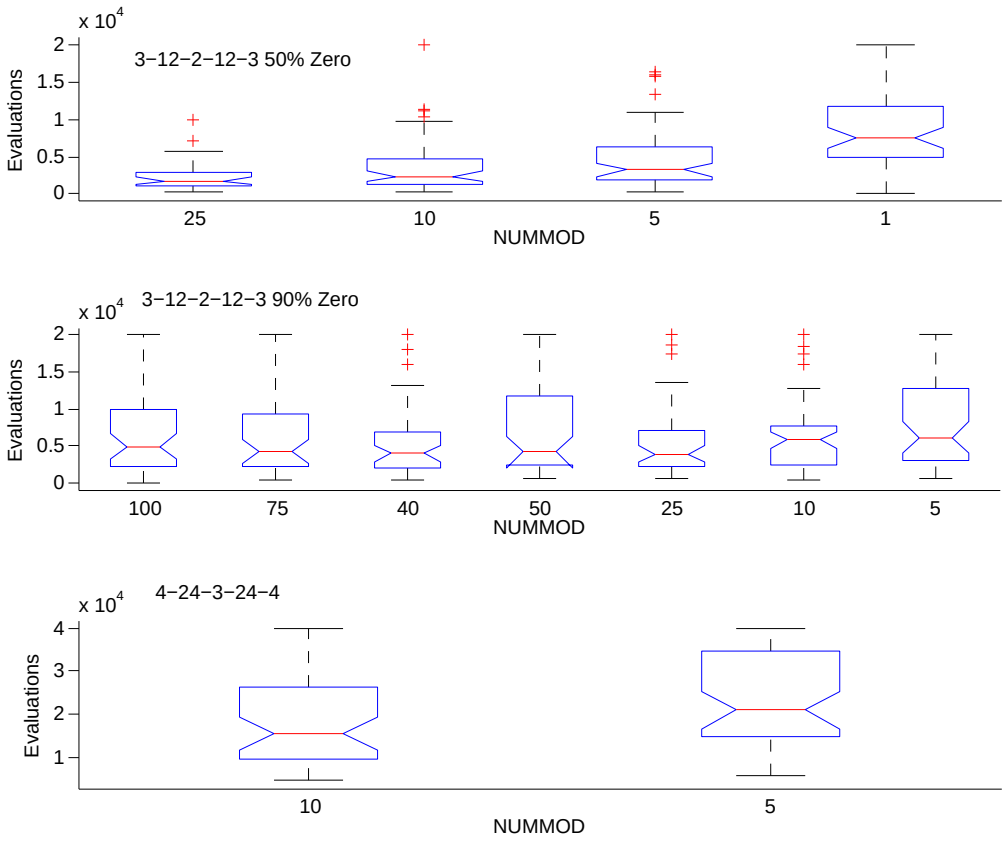

Figure 5.11: A box and whisker plot showing the effect of varying the number of individuals that are modified each tournament in the Group GA.

This GA was tested on all three AE tasks to see the effects reducing NUMMOD had on evolution. For each AE, we initially tested the case where NUMMOD was equal to group size, which is equivalent to running the standard Group GA. We then decreased NUMMOD to see how this effected performance. On the 4-24-3-24-4 AE only two data points are shown because when NUMMOD was reduced below 5 performance was extremely poor (full results are available online at ntomko.wordpress.com).

As figure 5.11 shows, reducing NUMMOD on both the 4-24-3-24-4 and 3-12-2-12-3 with 50% zero weights negatively impacts performance. On the 3-12-2-12-3 with 90% zero weights AE the impact of changing NUMMOD seems negligible. Based on the results of the previous section these results were a little surprising. This is because increasing the number of individuals being selected in the modified Binomics GA improved performance, so one would expect that reducing the number of individual selected in the modified Group GA to improve performance as well. Possible reasons for this are discussed in the next section.

5.7 Comparing the Binomics and Group GA to the Microbial GA on the AE Task

Here we compare the Microbial GA, which is a standard steady-state GA, with the Binomics, Modified Binomics, and Group GA (group evolution based GAs) on the AE task. The purpose of this comparison is to determine how the performance of each of these GAs varies as the percentage of zero weights in the AE is increased from 0% to 50% to 90%. Increasing the percentage of zero weights in the AE makes it more of a symbiotic task because at a certain point a single individual in the population will be unable to solve the task on its own. At one extreme, with no zero weights, a division of labour is not required because it is possible to evolve a single individual to solve the task. At the other extreme, where 90% of the weights are set to zero, a division of labour is a necessity because it is impossible for any single individual to solve the task on its own. The 50% zero weight case may be the most interesting because with this amount of zero weights a single individual is still able to solve the task, but as our results show, the group evolution based GAs outperform the Microbial GA in this case.

Comparisons between these GAs (figure 5.12) were made between the best performing parameter combinations of each GA over 50 runs and performance was measured in terms of the number of evaluations it took to evolve a perfect AE (full results used to construct this figure can be found online at ntomko.wordpress.com). These results can be summarised as follows:

- For all the GAs, increasing the percentage of zero weights makes the task more difficult.

- When the percentage of zero weights in the AE is set to zero, the Microbial GA significantly outperforms the other GAs but all the GAs are able to solve the task. This is confirmed by statistically comparing the Microbial GA to the Modified Binomics GA which results in a p-value of 0.008 and and a Bayes Factor of 0.3.

- At 50% zero weights, the Group and Modified Binomics GA are significantly better than the Microbial GA ($p=1 * 10^{-4}$ and Bayes Factor = 0.01 for the Group / Microbial comparison and p=0.04 and Bayes Factor = 0.14 for the Modified Binomics / Microbial comparison).

- At 90% zero weights the Microbial GA was unable to solve the task because there are no viable networks when 90% of the weights are set to zero. In this case the

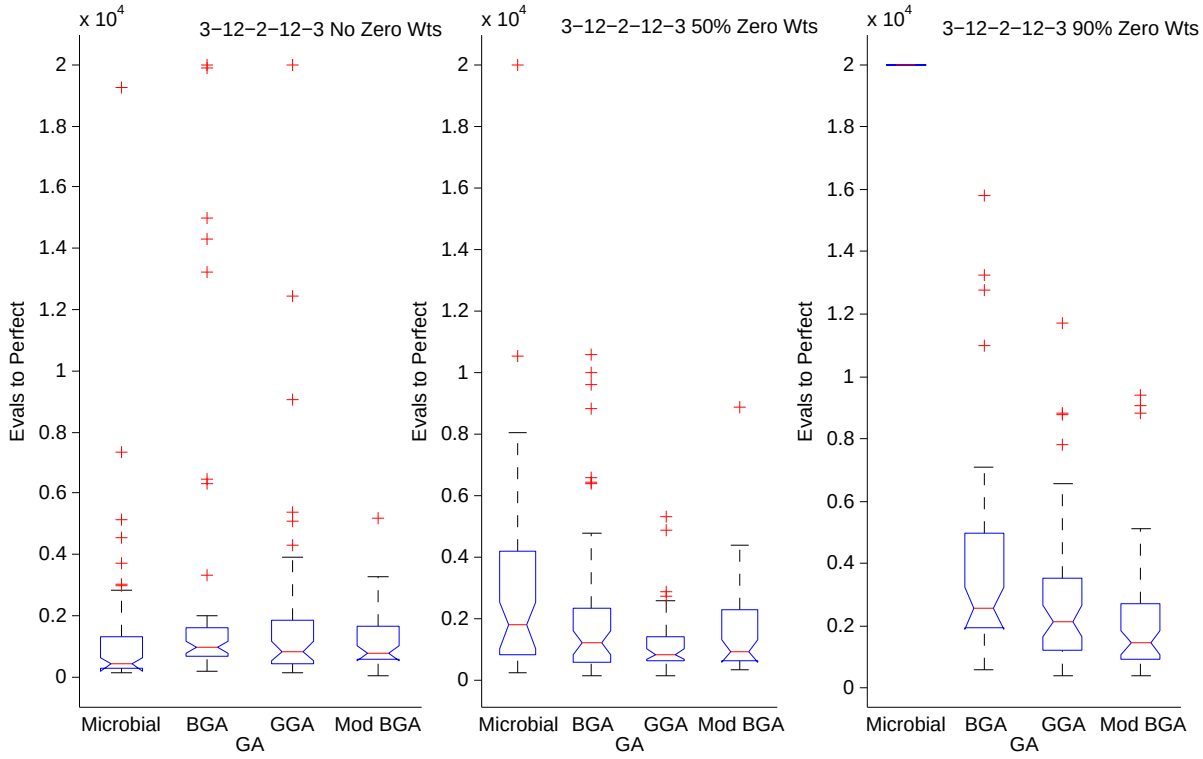

Figure 5.12: A box and whisker plot comparing the Microbial GA, the Binomics GA (BGA), Group GA (GGA) and the Modified BGA on a 3-12-2-12-3 autoencoder task with different percentage of zero weights.

Modified Binomics GA is the best performing GA.

It could be argued that a Bonferroni correction should be applied to these results to take into account the multiple comparisons. As discussed in Appendix A, two of the benefits of using Bayes factors are that a correction does not need to be applied in these cases and Bayes factors, unlike p-values, can be compared. The goal of these results is to show how group evolution based GAs are more robust to evolving networks with a large percentage of zero weights so Bayes factors were provided so that the reader can get an idea of the magnitude of the difference in performance. For these reasons we do not apply any multiple comparison correction.

These results show that the group evolution based GAs (Binomics, Modified Binomics and Group GAs) are more robust to changes in the percent of zero weights in the network. This can be quantified by looking at how the performance of each GA changes as the percentage of zero weights increases. For example, it takes the Microbial GA four times longer to solve the AE with 50% zero weights than it does with 0% zero weights and it is unable to solve the 90% zero weight case. On the other hand, the Modified Binomics GA solves the 0% and 50% zero weight case in approximately the same number of evaluations and solves the 90% zero weight case in less than double the number of evaluations as the other two cases. In general this means that if there is any possibility that a division of labour may be required to solve the given task, it probably make sense to apply a group evolution based GA.

Both Geoff Hinton's work on drop-out training (Hinton and Srivastava, 2012) and SANE (Moriarty and Miikkulainen, 1996, 1995) have shown success in applying zero weights to evolving neural networks so it was a little disappointing that in our case increasing the amount of zero weights reduced performance. That being said, the method of adding and maintaining zero weights we used was designed so that we could study how changing the amount of symbiosis in a task impacted evolution, it was not designed for optimal performance. An interesting area of research going forward would be to try to find a better way of zeroing out weights that doesn't reduce performance.

5.8 Discussion

In both standard GAs and the Group GA the UoE and UoS are the same, the difference between them is that in standard GAs, the UoE and UoS are individual population members, while in the Group GA, the UoE and UoS are groups of population members. In

this chapter we introduced the Binomics GA where the UoE was a group and the UoS was an individual so we could investigate the effect of changing the UoS while keeping the UoE at the group level. Below we summarise the key results from this chapter, make recommendations on how to choose the UoS, and finally compare the Binomics GA to existing algorithms.

5.8.1 Summary of Results

When the UoE and UoS are different, fitness needs to be passed from the UoE to UoS. In section 3.2.2 the Binomics GA was tested on both the immune system and AE tasks with different R values to determine the effect this has on evolution. On the immune system task the optimal R was 0.75, and on the AE task the optimal was 0.05. This means that taking into account an individual's historical fitness is more important on the AE task than it is for the immune system task.

In the results section we compared the performance of the Binomics to the Group GA on both tasks. Results show that on the easier 3-12-2-12-3 AEs there is no difference in performance but on the 4-24-4-24-4 network the Binomics GA is significantly better than the Group GA. On the 4-antigen immune system task the opposite is true, the Group GA is much better than the Binomics GA. One possible reason why all the GAs performed similarly on the 3-12-2-12-3 task is because the task was relatively simple and the effects of varying the UoS therefore had no impact on the performance of evolution.

Finally, the modified Binomics GA was tested to determine the effect of increasing the number of tournaments per evaluation (NUMT) on evolution. On the AE tasks the modified Binomics GA outperformed the standard Binomics GA when NUMT was set greater than one but less than the group size. On the immune system task, when performance is measured in terms of number of evaluations to evolve the first population that contained antibodies that matched all the antigens, the modified Binomics GA outperforms the standard Binomics GA to the point that it performs as well as the Group GA. However, when performance was measured in terms of how well evolution conserved the different antibody species in the population, the Group GA was much better than the modified Binomics GA.

5.8.2 Choosing the UoS

A goal of this chapter was to understand how varying the UoS, while keeping the UoE at the group level, impacted evolution. Our results show that in some cases it will make

sense to use the Group GA where the UoE and UoS are both groups and in other cases using the Binomics GA where the UoS is the individual will make more sense.

The immune system task was best solved when the UoS was a group which is equivalent to having a large amount of selection and mutation per evaluation. This result is backed-up by the fact that the Group GA outperformed the standard Binomics GA, and the modified Binomics GA worked best when the number of tournaments per evaluation was set equal to the group size. On the more difficult 4-24-3-24-4 AE task, the standard Binomics GA was significantly better than the Group GA, which would imply that on this type of ANN task, too much selection and mutation per evaluation reduces performance. This is confirmed by the fact that when the modified Binomics GA was tested on the AEs the optimal number of tournaments (NUMT) per evaluation was always much smaller than the group size.

Even though changing NUMT in the modified Binomics GA doesn't change the UoS (individuals are still being selected), changing how many individuals selected per tournament could be viewed as equivalent to changing the UoS. If this is the case, the results show that on some tasks having a UoS that is a partial group could be beneficial. To test this further the Group GA was modified so that instead of selecting an entire group, a randomly chosen subset was selected. As shown in section 5.6.1 on the AEs tested, this modified Group GA performed worse then the standard GA where the entire group is selected as a whole. The fact that changing the number of individual selected per evaluation in the Binomics GA improves performance, but on the Group GA selecting a partial group reduces performance, is something that needs further exploration.

Both the Binomics and Group GA will have to be tested on more tasks before it is possible to make any broad generalisations about when to set the UoS to a group, an individual or something in between. From a practical standpoint, even though the modified Binomics GA performs the best out of the algorithms we tested on these tasks, it does suffer from the fact that compared to the Group GA there are more parameters that need to be optimised; those being the time smoothing parameter R and the number of tournaments per evaluation NUMT. Also, the fact that the modified Binomics GA struggled to maintain different types of antibodies throughout evolution suggests that for tasks where explicit niching is important, setting both the UoE and UoS to the group, like in the Group GA, may be advantageous. Therefore when choosing between the Group and modified Binomics GA one needs to think about the trade-off between a potential performance boost and having to optimise extra parameters.

5.8.3 Comparing the Binomics GA to Existing Algorithms

The Binomics GA can be classified as a symbiotic GA because the UoE is a group of population members, so it is related to all the algorithms we reviewed in our literature review section 2.2.3 but here we limit our comparisons to algorithms where the UoE is a group of population members and the UoS is an individual population member.

SANE (Moriarty and Miikkulainen, 1996, 1995) which was first discussed in section 2.2.3 is a steady-state GA that has been successfully applied to the evolution of neural networks. In this context, SANE evolves a population of neurons whose fitnesses are determined by how well each one performs when grouped together with other neurons to form a full artificial neural network. In this algorithm the UoE are fully specified networks and the UoS are individual neurons, so like the Binomics GA, the UoE $\supset$ UoS. One of the differences between the Binomics GA and SANE is how fitness is assigned from the UoE to the UoS. In SANE an individual neurons' fitness is based on the average fitness of all the networks they took part in, which is different from how we time-smooth the fitness of individual neurons in the Binomics GA. Another difference between the two algorithms is that SANE is a generational algorithm, while the Binomics GA is based on the steady-state Microbial GA. The generation style of SANE means that the gene transfer between individuals is restricted to vertical transfer while the Binomics GA uses horizontal gene transfer or infection.

When Learning Classifier Systems (LCS) (Holland, 1976; Holland and Reitman, 1978) are analysed in terms of UoE and UoS we find that Michigan style LCS have UoE $\supset$ UoS as with the Binomics GA and SANE. One way to think about LCS is in terms of ANNs or bacteria in a sea where each individual classifier (rule) is equivalent to a single neuron or a single bacterium in the sea. In both Michigan and Pittsburgh style LCS, groups of these rules are combined to form complete rule sets, which means the UoE are complete sets of rules. The main difference between these two styles of LCS is that in the Pittsburgh LCS, the UoS are the complete set of rules so the UoE = UoS, while in the Michigan LCS, the UoS are individual rules so the UoE $\supset$ UoS. This means that in the Michigan LCS, there needs to be a way to assign fitness from the full classifiers (UoE) to the individual rules (UoS) (step 2 of our framework). Many different methods have been proposed to do this, including auctions with specificity-based arbitration mechanisms to allow default hierarchies to form, and bucket brigade algorithms for the temporal credit assignment problem. This has resulted in many different flavours of Michigan LCS. Because each individual in the population of a Michigan LCS is an individual rule that needs to be

combined with other rules to solve the task, it is similar to symbiotic GAs such as the Binomics GA and SANE.

SEAM (Symbiotic Evolutionary Adaptation Model) (Watson and Pollack, 2000b) is another GA that uses group evaluation and individual selection. In this algorithm two individuals are randomly chosen from the population and used to produce an offspring. This new offspring is then evaluated and if it is found to dominate its parents then it replaces them. Domination is determined using pareto co-evolution by building a template from randomly chosen individuals from the population and then superimposing both the offspring and the parents on this template to determine which is dominant (details can be found in (Watson and Pollack, 2000b)). In terms of UoE and UoS, we interpret this algorithm as evaluating groups of individuals using their pareto-coevolution templating process and the selecting the fitter of either the parents or offspring based on this group evaluation. Even though SEAM has the same UoE and UoS as the Binomics GA, how evaluation and selection is carried out are totally different.

In the final chapter of Part II of this thesis, we investigate how varying the group size parameter affects the performance of both the Group and Binomics GAs. We then present two methods that can be applied to these GAs that allow them to be run without having to pre-set the group size ahead of time, avoiding the need to optimse this parameter. We will will discuss some of the benefits of using group evolution rather than standard individual evolution to evolve ANNs.

Chapter 6

Evolving the Group Size

6.1 Introduction

In the previous two chapters we tested both the Group and Binomics GA to try to understand how varying the UoE and UoS impacts evolution. In both of these GAs the UoE is a group of population members which means that these GAs are able to evolve solutions to tasks that require a division of labour between component parts. In Chapter 4 we stated that one of the advantages the Group GA has over existing algorithms is that it can solve these types of tasks with minimal *a priori* knowledge about the optimal number of niches required. In their current form, both the Group and Binomics GAs need the group size parameter to be set before evolution begins. In this chapter we will explore how robust these algorithms are to changes in group size and then will present two methods that can be used so that the group size does not need to be set before evolution begins.

First we show how changing the group size affects the Group GA on the immune system task and the Binomics GA on the AE task. These examples were chosen because the Group GA performs better on the immune task and the Binomics GA performs better on the AE task. Then two methods that avoid pre-setting the group size are presented. In the first method, the group size is randomly chosen in evolution (random group size method) and in the second method, the group size is evolved (evolved group size method) by adding a group size weight gene to each individual in the population. We show how both these methods can be applied to either the Group or Binomics GA to solve the immune system and AE tasks.

Based on the results of this chapter, if the optimal group size is unkown then it makes sense to evolve the group size as part of evolution. This avoids the need to manually test a variety of different group sizes and, unlike randomly changing the group size during evol-

ution, evolving it means that less time is spent at sub-optimal group sizes. Comparisons made between the evolved group size Group and Binomics GA show that the Group GA is better at evolving appropriate group size due to differences in selective pressures between the GAs. Finally we discuss the benefits of applying the evolved group size algorithms to general ANN problems.

6.2 Tasks Used in this Chapter

As in the previous two chapters we use both the immune system and autoencoder (AE) tasks to test the different GAs. These tasks are described in detail in sections 4.2 and 5.2 respectively.

The other task used in this chapter is an NK fitness landscape (Kauffman and Johnsen, 1991; Kauffman, 1993). NK landscapes are encoded using bit strings of length N, where N controls the dimensionality of the landscape. The parameter K determines the degree of epistasis of the landscape, where epistasis can be thought of as determining the ruggedness the fitness landscapes is, with higher K's leading to more rugged landscapes. Fitness is calculated using randomly generated look-up tables that assign fitness to each locus (gene). This means that the fitness of a given bit string is the sum of the fitnesses at each locus. In Part II of the thesis, NK landscapes are used extensively to test how gene shuffling impacts evolution, so in section 7.2 we describe the NK landscape in detail. In this chapter the details of the NK landscape are not important, the only thing to remember is that NK landscapes are best solved by individuals, rather than groups of population members. This means that when the evolved group size method is tested on the NK landscape the group size should decrease throughout evolution.

6.3 The Effect of Varying the Group Size

To show how group size impacts the performance of evolution, we tested the Group GA on the immune system task and the Binomics GA on the AE task using a variety of different group sizes. The results show that even though there is an optimal group size for each task, both GAs can perform satisfactorily at a wide range of group sizes when the population is large enough.

6.3.1 Varying Group Size of the Group GA

The Group GA was used to evolve populations of either 100 or 1000 antibodies, at a variety of group sizes (for population 100, group sizes 5, 10, 15, 20, 25 or 50; for population 1000, group sizes 5, 10, 25, 50, 100 or 150). Figure 6.1 shows: (1) The number of evaluations it took to evolve the first population that contained at least one antibody that matched each of the four different antigens and (2) how many runs out of 50 the GA was able to conserve the fit antibodies throughout evolution, so that after 1600 K evaluations there were antibodies in the population that matched the four different antigens. One can see that the optimal group size (from the ones tested) was 10 at both population sizes, because at this size, the number of evaluations to evolve four antigens to match the different antigens is minimised and the number of runs where there are four fit antigens conserved at the end of evolution is maximised. These figures also show that the Group GA is more robust to group size changes at a population size of 1000. When the population size is 100, evolution fails (does not evolve a population of antibodies where there is at least one antigen in the population that matches each of the four antigens) when the group size is 25 or bigger, but when the population size is increased to 1000 the Group GA is able to solve the task with group sizes up to 100.

To explain these results, one needs to understand how varying the group size impacts the selective pressure on the fitness contributing antibodies in the group. The antibodies in any group can be separated into two different classes, the fitness contributing antibodies and the freeloading antibodies. The fitness contributing antibodies are the ones that closely match one of the four antibodies and contribute to the fitness score of the group, while the free-riding antibodies are the remaining antibodies in the group that get copied just because they are in a group with high fitness antibodies. At low group sizes, the probability of getting a high scoring group, one that contains antibodies that closely match the four different antigens is low, which is why the Group GA performs poorly when the group size is set below 10. As the group size is increased, the probability of getting a high scoring group increases, but at the same time the differential copying of fitness scoring antibodies versus freeloading antibodies decreases. This is because in the Group GA the entire winning group overwrites the entire losing group, which means that these free-riding antibodies in the fitter group are replicated along with the fitness contributing antibodies. As the group size becomes larger, the probability of having a high percentage of free-riding antibodies in the group increases which will make it more difficult to evolve a population where niching has occurred.

The relationship between group size and population size also impacts the performance of the Group GA. Figure 6.1 shows that with a population size of 100 the Group GA fails (there were no runs where antibodies that matched each of the four antigens were present in the population at the end of evolution) if the group size parameter is increased above 20. When the population size is increased to 1000 the algorithm can solve the 4-antigen task up to a group size 100. This is important because as the group size is increased, the probability of having the same antibodies in both groups increases which limits the genetic diversity between groups and reduces evolutionary performance.

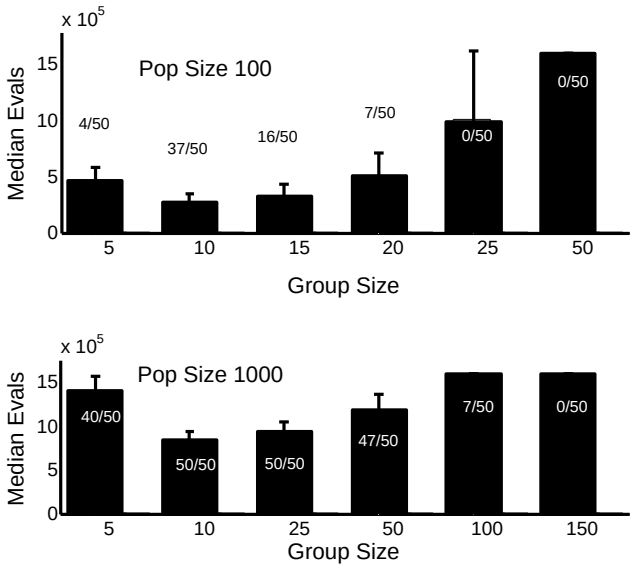

Figure 6.1: The performance of the Group GA on the 4 antigen task with different group sizes, where performance was measured as: (1) The median (height of bar) and Interquartile Range (shown as an error bar) number of evaluations over 50 runs that it took the Group GA to evolve the first population with antibodies matching all four antigens and (2) the number of runs out of 50 where there were antibodies that matched all the different antigens at the end of evolution; this is shown as the text at the top of each bar. At both a population size of 100 (upper plot) and of 1000 (lower plot) the optimal group size is 10 because the median number of evaluations is minimised and the number of successful runs is maximised.

In Forrest et al. (1993) the authors apply their diversity maintaining GA to the same 4-antigen task and discuss the effect of varying the antibody group size (which they call sample size) and antibody population size to the performance of evolution; they highlight issues closely related to those discussed here. They find that with their algorithm, the carrying capacity of the antibody population is 15 antibodies to 1 antigen, meaning that

a population of at least 60 antibodies is required to match and maintain four different antigens. They also show that on a four antigen task, a sample size (group size) of at least 7 is required for the population to niche to match four different antigens, but they do not explore whether there is a maximum sample size that allows this niching to occur.

6.3.2 Varying Group Size of the Binomics GA

To determine how group size affects performance of the standard Binomics GA on the 3-12-2-12-3 AE task we tested it with a population size of 100 and group sizes of 5, 10, 15, 50, and 75 and with a population size of 1000 and group sizes of 10, 50, 100, 250, 500 and 750. This simulation was set up so that each individual in the population was a partially specified network with 90% of its weights initialised to zero. The other parameters used are as follows: the rate of infection was set to 50%, mutation amount was set to 0.1, R (the fitness smoothing parameter) was set to 0.05, and the add and delete weight probabilities were set to 10% and 90% respectively to ensure that each individual in the population had around 90% of its weights set to zero throughout evolution. A full network was constructed from a group of partial networks by summing the weights at each locus. More details of how the Binomics GA is applied to the AE task can be found in section 5.4.

Figure 6.2 shows how group size impacts performance at populations sizes of 100 and 1000. Performance is measured as the number of successful runs and the median number of evaluations it took to evolve the first perfect AE over 50 runs. On this task, a successful run is one where a perfect network was found before 20 K evaluations.

At both population sizes, the Binomics GA performs best when the group size is set to around 60 to 75% of population size. This is probably because each individual in the population has approximately 90% of its weights set to zero and therefore a large number of individuals are required to construct a fully functioning network. Like the Group GA on the immune system task, performance is more robust when a larger population is used.

6.4 Randomly Generated Group Sizes

If the optimal group size is not known, one of the easiest ways to change the group size during evolution is to randomly select a new group size before each evolutionary tournament. The benefit of this method over keeping the group size constant throughout evolution is that it increases the chances of evolving high fitness solutions over problems where the optimal group size is unknown beforehand. For example, in the immune system task, if the number of antigens is unknown and too large or too small a fixed group size is

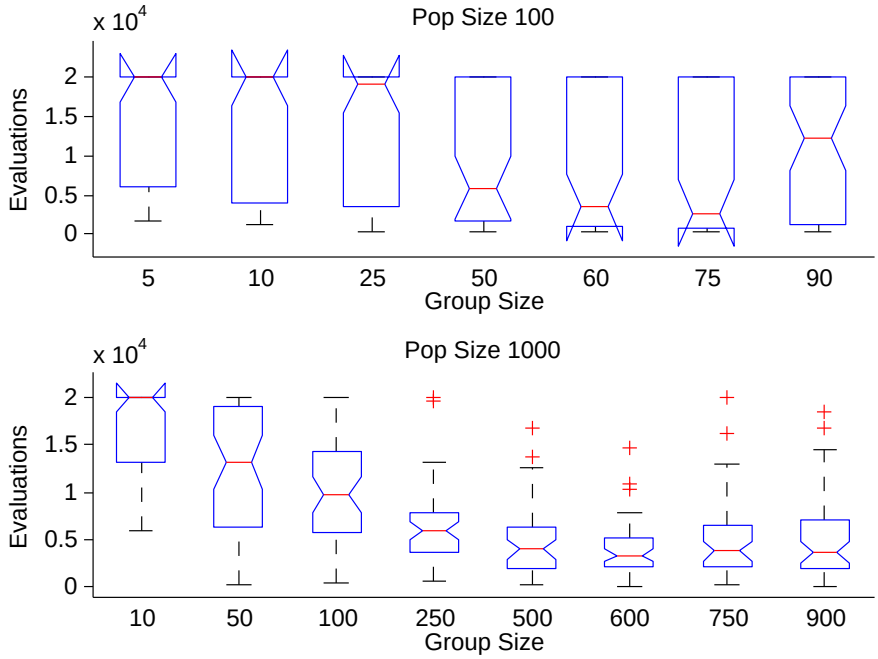

Figure 6.2: Box and whisker plots showing how varying the group size effects the performance of the standard Binomics GA on the 3-12-2-12-3 AE with 90% of the weights set to zero with population sizes of 100 (top) and 1000 (bottom). The large interquartile ranges on the population size 100 plot shows that performance is not as good as it is when the population size is set to 1000.

chosen, then the Group GA will struggle to evolve a niched population of antibodies. In that case, a parameter sweep through group sizes would be needed to find the appropriate value. As we show here, this parameter sweep can be avoided if the group size is randomly changed during evolution because this allows a range of group sizes to be used.

Figure 6.3 shows the performance of the Group GA on the immune task and the Binomics GA on the AE task where the group size was randomly changed each tournament. For both GAs, we tested population sizes of 100 and 1000 using the same parameters as were used in the previous section when the group size was manually changed. We limited the group size range between 2 and 25% of the population size for the Group GA on the immune system task, i.e. for population size 100, the group size was a randomly chosen integer between 2 and 25 and for population size 1000, the group size could be anywhere between 2 and 250. For the Binomics GA on the AE task, the group size range was set between 2 and 75% of the population size, i.e. for population size 100, the group size was a randomly chosen integer between 2 and 75, and for population size 1000, the group size could be anywhere between 2 and 750. The reason for limiting the group sizes to a certain

range is that without these limits evolution performed poorly.

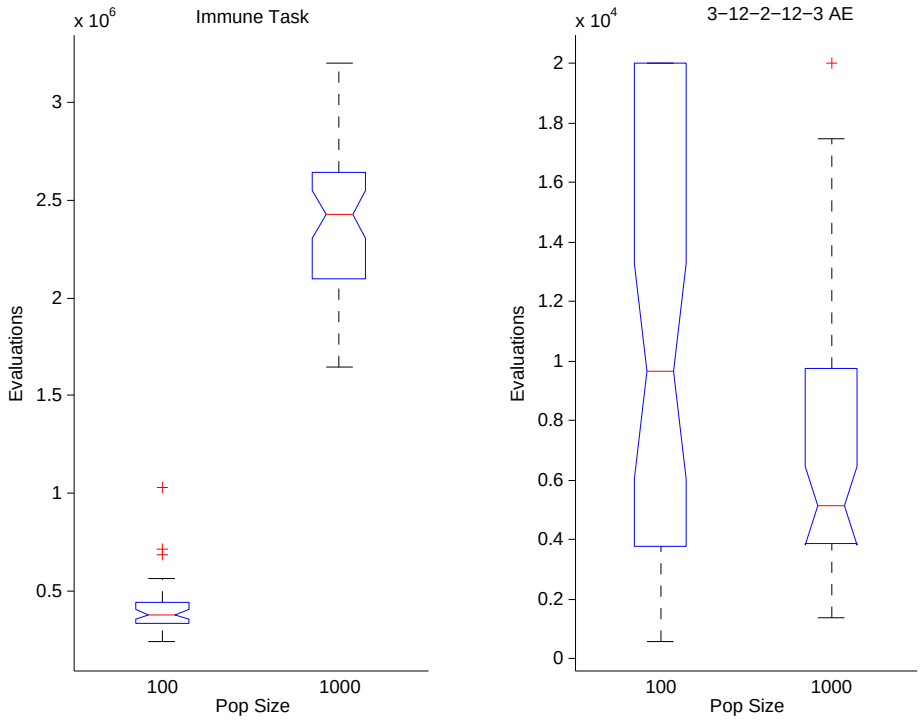

Figure 6.3: Box and whisker plots showing performance of random group size Group GA on the 64 bit, 4 antigen task (left) and the random group size Binomics GA on the 3-12-2-12-3 AE with 90% zero weights. On the immune task, performance was measured as the number of evaluations it took to evolve the first population that contained four antibodies that perfectly matched the antigens. On the AE task, performance was measured as the number of evaluations to evolve the first perfect AE. When performance on the immune task was measured in terms of number of runs in which perfect antibodies were conserved: pop 100 only 9/50; while with pop 1000 42/50. Therefore pop 1000 is better at conserving niches.

Both the Group and Binomics GA were able to solve their respective tasks when the group size was randomly varied during evolution. In both cases, performance was better at the higher population size which is to be expected because we found in the previous section that both algorithms were more robust to changes in group size at the higher population size. One drawback of this random group size method is that this range needs to be pre-specified and if it is not specified correctly the performance of the algorithm could be significantly reduced. For the tests run in this section, the group size range was set with some prior knowledge because we had already performed a parameter sweep to generate the results of the previous section. In reality, this information will not normally be

available, so there is a risk that the group size range chosen will lead to poor performance. For example, if the group size range of the Group GA on the immune system task was set between 2 and 75% of the population size, rather than 2 and 25% of population size, then it struggles to evolve a niched population of antibodies. On these tasks it takes the random method about twice as many evaluation to solve as the average over all the manually set group sizes. This means that each test using the random group size method is equivalent to testing two manually set group sizes. A logical starting point when using the random group size method is to limit the group size between 2 and 50% of the population size. Next, we modify both the Group and Binomics GA so that the group size is evolved as part of evolution. In this case the need to pre-set a group size or group size range is eliminated.

6.5 Evolving Group Sizes

Two shortcoming of the random group size method just presented are: (1) a group size range still needs to be specified and (2) much of the time the random group sizes chosen will be less than optimal, thus leading to inefficiencies. We now present a method where the group size is evolved as part of evolution. This is done by adding a single, randomly generated, real numbered gene between 0.0 and 1.0 which serves as the group size weight to every genotype in the population. These genes to determine the group size as follows:

1. Start with an empty group

2. Randomly choose an individual from the population

3. Calculate the total group size weight by adding the individual weights of the real numbered genes. If the total is greater than 1.0 then stop (the group is complete), otherwise, go back to step 2.

This means that smaller values of this new group size weight gene promotes larger group sizes, and *vice versa*. After the two groups (potentially of different sizes) have been constructed, the fitness of each group is calculated and then the losing group is infected with genes from the winning group before being mutated. If the size of the winning group is bigger than the size of the losing group then a randomly chosen subset of the winning group are recombined with all the individuals from the losing group; if smaller then all of the individuals from the winning group are recombined with a subset of the losing

group before mutation. This ensures that the population size stays constant throughout evolution.

In both the Binomics and Group GA the tournament loser inherits the group size weight gene of the tournament winner. In the case of the Group GA where two groups of individuals are involved in the tournament, the group size weight genes of the winning group overwrite the group size weight genes of the losing group. In our GAs, the group size weight gene is copied from winner to loser regardless of what the rate of infection is. Even if only 50% of the genes from the winning group are infecting the losing group, the group size gene will always be copied over. After the group size weight gene is copied from winner to loser it is mutated by adding a normally distributed random number to it. This mutator has mean zero, and a standard deviation that is set as a parameter. In the tests below we set the standard deviation to either 0.1, 0.01 or 0.001. Other than this modified method of choosing group sizes, both the Group and Binomics GAs remain the same as described in chapters 4 and 5.

We demonstrate this evolved group size method on the immune system task, an AE task and on a NK fitness landscape. Unlike the immune system and AE tasks the NK task is best solved using individual, rather than group evolution, so we would expect that the group size will be minimized by maximising the group size weight gene during evolution.

6.5.1 Evolving Group Size: Group GA on the Immune System Task

The evolved group size version of the Group GA was tested on the 4 antigen task with population sizes of 100 and 1000. Performance was measured over 50 runs using median and IQR evaluations. As figure 6.4 shows, evolving group size with the Group GA works well with a population size of 1000, but struggled to evolve antibodies that matched all four antigens at a population size of 100. We believe that this poor performance at the lower population size is due to the fact that at a population size of 100, unless the group size is between about 10 and 15, evolution is unable to maintain a niched population of antibodies. At higher population sizes, the range of viable group sizes increases which allows evolution to find group sizes where niching occurs.

One of the most interesting aspects of this evolved group size method is that it was able to evolve group sizes in the viable range between 10 and 50 even though there seems to be a selective pressure for groups to get bigger and bigger. This is because based on how the fitness of a group is calculated in the immune system task, the larger the group, the higher the probability of having a fitter group. The fitness of a group is calculated by

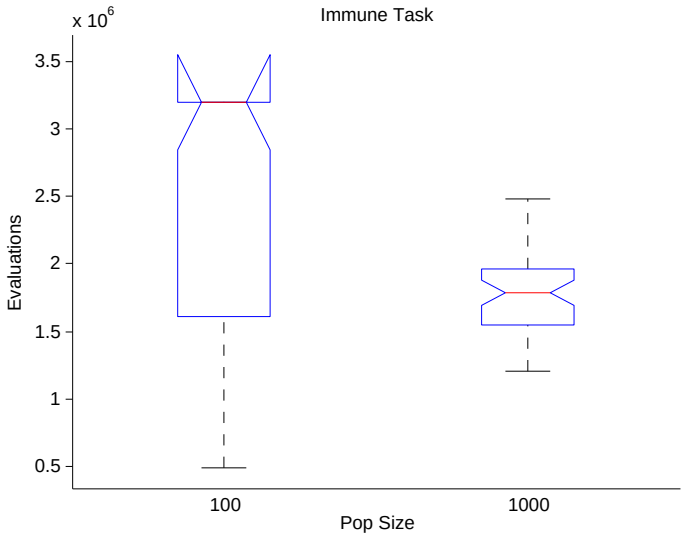

Figure 6.4: Box and whisker plots showing performance over 50 runs of the evolved group size version of the Group GA on the 64 bit, 4 antigen immune task with population sizes of 100 (left) and 1000 (right). For both cases the group size weight gene was randomly initialised between 0.0 and 1.0.

matching all the antibodies in the group to all the antigens and then taking the average of the best match scores. So for any population of antibodies, the highest possible group fitness score can be found by taking the entire population as the group, which would seem to imply that there is a selective pressure to evolve the largest groups possible. But what we found was that the average group size weight gene always converged to a value around 0.03, which is equivalent to a group size of around 30 (over 50 runs the average group size weight gene was 0.0290 with a standard deviation of 0.0039). Figure 6.5 shows how the average group size weight gene changed during evolution for five different runs. As this figure shows, in both the population size 100 and 1000 cases, the group size weight gene starts out high and then converges to a value in the 0.02 - 0.09 range. It is interesting to note the population size 100 plot is a lot more noisy than the population size 1000 plot. This is likely because when the population size is 100 the range of viable group sizes is much smaller and therefore evolution struggles to converge on an optimal group size weight gene.

The evolved values of the group size weight gene in these experiments is in the neighborhood of the optimal group size, but before concluding that selection was indeed responsible for this satisfactory result we carried out further tests. First, we tested the evolved group size version of the Group GA where the group size weight genes were all initialised to zero, meaning that in the early stages of evolution the group sizes would be relatively big.

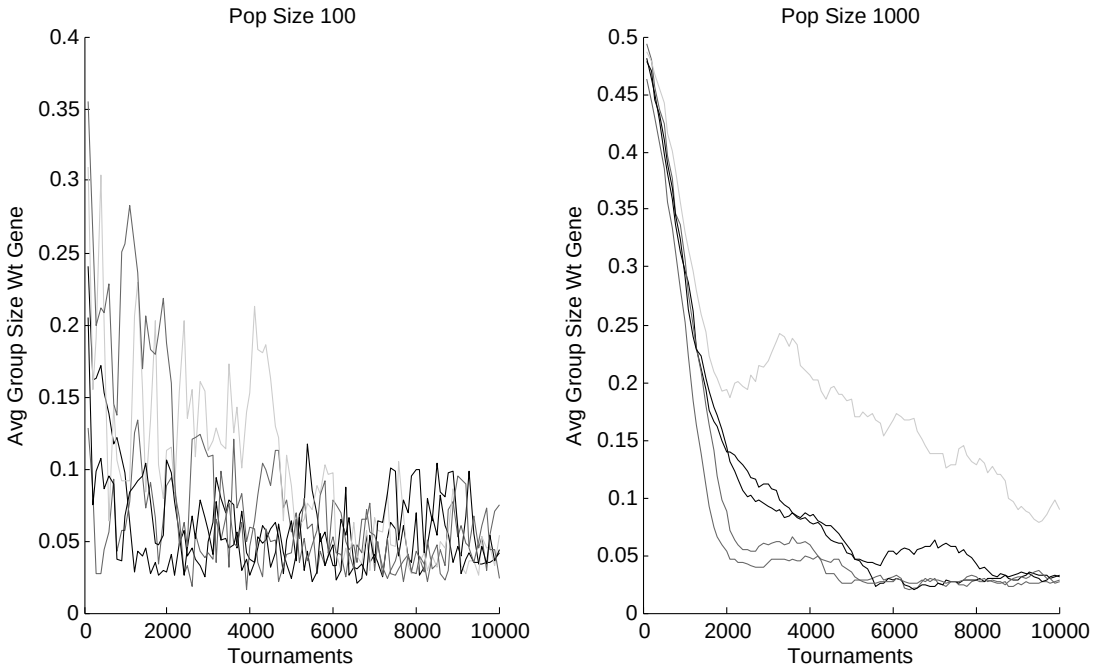

Figure 6.5: A plot of the average group size weight gene over time for 5 different runs using a population size of 100 and 1000.

Like the case where the group size weight genes were randomly intialised between 0.0 and 1.0, the Group GA was able to solve the 4 antigen task, and evolution of the group size weight gene again converged to a value around 0.03 (over 50 runs the average group size weight gene was 0.0284 with a standard deviation of 0.0027). We then ran it on a flat fitness landscape where each group of antibodies was given a random fitness score instead of a fitness score based on how well it matched the different antigens. This was done to see if there was any selective pressure on the group size weight gene that was not related to the fitness of the antibody groups. We ran evolution for one million tournaments with different group size weight gene mutation rates and initialising the weight genes to different values and found that the plots of the average group size weight gene over time looked like a random walk. In other words, there did not seem to be any selective pressure on the group size weight gene on a flat fitness landscape.

On the basis of these further tests we conclude that selection was indeed responsible for evolving a group size reasonably close to the optimal. Hence, we have shown, in this one example at least, that it is possible to evolve the group size. This avoids the two shortcomings of the random group size method described earlier, which were: (1) a group size range needs to be specified and (2) much of the time the random group sizes chosen are less than optimal. Next we apply this method to the Binomics GA and try evolving

group sizes of both the Binomics and Group GA on the AE task.

6.5.2 Evolving Group Size on the Autoencoder Task

Like the Group GA, the group size gene can also be used to evolve the group size of the Binomics GA. This evolved group size version of the Binomics GA is demonstrated on a 3-12-2-12-3 AE and then performance of the evolved group size versions of the Group and Binomics GA are compared.

Evolving Group Size of the Binomics GA

As applied to the AE task, the evolved group size version of the Binomics GA is as follows:

1. Randomly choose two individual partial networks from the population and compare their stored fitnesses
2. Genetically modify the less fit network using infection and mutation
3. Copy the group size gene from winner to loser and then mutate it by a small amount
4. Randomly choose n other individual networks from the population to be combined with this tournament losing network by:
   (a) Start with only the losing individual in the group
   (b) Randomly choose an individual from the population
   (c) Calculate the total group size weight gene by summing up all the individual group size weight genes. If the total is greater than 1.0 stop (the group is complete), otherwise, go back to step b. If the total group size weight gene is equal to zero then the group size is set equal to the population size.
5. Combine all these partial networks into a single full network by summing the weights at each locus
6. Evaluate this network on the task
7. The fitness of the network is passed to all individuals that made up the network according to: New Fitness $= \mathrm{R} * \text{Network Fitness} + (1.0 - \mathrm{R}) * \text{Old Fitness}$
8. All partial networks are put back into the population and the cycle is repeated

We applied this GA to a 3-12-2-12-3 AE task where each individual in the population had 90% of their weights initialised to zero and this approximate ratio was maintained through evolution using the appropriate add and delete weight operators. For these tests we used the following parameters: the population size was set to either 100 or 1000, the standard deviation of the mutator applied to the weight genes set to 0.1, the rate of infection was to 0.5, R was set to 0.05, the standard deviation of the mutator applied to the group size gene was set to either 0.1, 0.01 or 0.001 and the group size weight gene was initialised either to zero, randomly between 0 and 0.4 or randomly between 0 and 0.05. Evolution was run until a perfect network was evolved or to maximum of 40 K evaluations. If 40K evaluations was reached before a perfect AE was evolved the run was deemed unsuccessful.

In Table 6.1 the performance of the evolved group size Binomics GA is shown in terms of median evaluations over 50 runs it took to evolve a perfect network and number of successful runs. The average group size equivalent at the end of evolution is also listed. This value is found by taking the inverse of the average of the group size weight genes in the population. For example, if at the end of evolution the average of the group size weight genes in the population is 0.1 then this is equivalent to an average group size of 10.

From these results we can see that performance is much better with a population size of 100 than with a population of 1000 and when the group size weight gene is initialised to zero rather than randomly between 0 and 0.4. We know from manually varying the group size parameter that the Binomics GA performs best on this task when the group size is set to around 75 with a population of 100 and 600 for a population of 1000. When the group size weight gene is initialised to zero and an appropriate mutation rate is used the average group size weight gene in the population at the end of evolution specifies a group size in the same range as the optimal group size. With a population size of 100 and using a mutation amount of 0.01 the average group size at the end of evolution is 51, with the optimal group size being 75 and with a population size of 1000 and mutation amount of 0.001 the group size is 650, with the optimal being 600.

A group size of 75 is equivalent to an average group size weight gene of approximately 0.0133 (1/75) and a group size of 600 is equivalent to a group size weight gene of approximately 0.00167 (1/600). Because these group sizes require very small group size weight genes it is not surprising that the performance is better when the group size gene is initialised to zero, rather than randomly. As table 6.1 shows, when the initialisation range is narrowed so that the group size genes are randomly initialised between 0 and 0.05

Table 6.1: The performance of the evolved group size version of the Binomics GA on the 3-12-2-12-3 AE task. This table shows results at population sizes of 100 and 1000, with group size mutation rates of 0.01 and 0.001 and initialising the group size weight gene to either 0, randomly between 0 and 0.4 (RAND 0.4) or randomly between 0 and 0.05 (RAND 0.05). Performance is measured in terms of median evaluations over 50 runs and number of successful runs out of 50, where a successful run is one where evolution found a solution before 40K evaluations. The equivalent group size is calculated by taking the inverse of the average group size weight gene at the end of evolution and the optimal group size was found by manually varying the group size.

Pop Size	Group Size Gene Init	Group Size Gene Mut	Equiv Group Size	Median Evals (succesful runs)	Optimal Group Size
100	0	0.01	51	3.2 K (45)	75
100	0	0.001	89	31.2 K (28)	75
100	RAND 0.4	0.01	11	39.9 K (25)	75
100	RAND 0.05	0.01	40	4.9 K (45)	75
100	RAND 0.4	0.001	10	40.0 K (22)	75
1000	0	0.01	69	17.5 K (42)	600
1000	0	0.001	650	7.5 K (50)	600
1000	RAND 0.4	0.01	poor results		600
1000	RAND 0.4	0.001	poor results		600

rather than between 0 and 0.4 performance improves drastically. Next we test the evolved group size version of the Group GA on the same AE.

Evolving Group Size of the Group GA

The evolved group size version of the Group GA can be applied to the 3-12-2-12-3 AE task as follows:

1. Randomly choose two groups of partial AE networks from the population, with the size of each group determined using the following process:
   (a) Start with an empty group
   (b) Randomly choose an individual from the population and add to group

   (c) Calculate the total group size weight gene by summing up all the individual group size weight genes. If the total is greater than 1.0 stop (the group is complete), otherwise, go back to step b.

2. Combine each group of partial networks into a full network by summing the weights at each locus

3. Calculate the fitness of each full network with the winning network being the one with the higher fitness

4. The group of partial networks that made up the losing network is replaced with mutated copies of the partial networks that made up the winning network. If the size of winning group is bigger than the size of the losing group then a randomly chosen subset of the winning group are recombined with all the individuals from the losing group; if fewer then all the individuals from the winning group are recombined with a subset of the losing group.

5. Both groups of antibodies are put back into the population and this process is repeated

Again we set the population size to either 100 or 1000, the mutation rate of the network weight genes to 0.1, the group size weight gene mutation amount to 0.01 or 0.001 and initialised this gene to either zero or randomly between 0 and 0.4.

Table 6.2 shows the same performance metrics for the Group GA as were shown for the Binomics GA in the previous section. To determine whether the evolved group size is in the same range as the optimal group size, the Group GA was run with a variety of different group sizes. As this table shows, with a population size of 100, the optimal group size for this task is 25, and with a population of 1000, the optimal is 250. It is interesting to note that the optimal group size of the Group GA on this task is smaller than that of the Binomics GA.

Unlike the Binomics GA, the Group GA performed reasonably well regardless of whether the group size weight gene was initialised randomly between 0 and 0.4 or to zero, although it still performs better when the group size genes were initialised to zero. With a population size of 100 and a mutation amount of 0.01 the group size weight gene converged to a value that corresponds to a group size around 15 regardless of whether the group size gene was initialised to zero or randomly. This is in the same range as the optimal group size (25) so leads us to believe that evolution is optimising group size. When

Table 6.2: The performance of the evolved group size version of the Group GA on the 3-12-2-12-3 AE task where 90% of the genes were intiailised to zero. This table shows results at population sizes of 100 and 1000, with group size mutation rates of 0.01 and 0.001 and initialising the group size weight gene to either 0 or randomly between 0 and 0.4 (RAND 0.4). Performance is measured in terms of median evaluations over 50 runs and number of successful runs out of 50, where a successful run is one where evolution found a solution before 40 K evaluations. The equivalent group size is calculated by taking the inverse of the average group size weight gene at the end of evolution and the optimal group size was found by manually varying the group size.

Pop Size	Group Size Gene Init	Group Size Gene Mut	Equiv Group Size	Median Evals (succesful runs)	Optimal Group Size
100	0	0.01	15	8.9 K (49)	25
100	0	0.001	33	10.0 K (44)	25
100	RAND 0.4	0.01	14	9.4 K (45)	25
100	RAND 0.4	0.001	9	15.8 K (38)	25
1000	0	0.01	28	4.5 K (50)	250
1000	0	0.001	113	1.6 K (50)	250
1000	RAND 0.4	0.01	10	27.3 K (37)	250
1000	RAND 0.4	0.001	6	33.9 K (29)	250

the population size was increased to 1000, the best results occurred when the group size gene was initialised to zero and the mutation amount was set to 0.001. In this case the average of the final group size genes were equivalent to a group size of 113 which is the same order of magnitude as the optimal group size of 250. Randomly initializing the group size weight gene at a population size of 1000 reduced performance, likely because evolution struggled to move the group size weight genes of the entire population to a optimal size.

Figure 6.6 provides a good visual summary of the results presented in the last few sections. In this figure the performance of the Binomics GA and the Group GA with the different methods of determining group size (manual, random and evolved) on the 3-12-2-2-12-3 AE with 90% zero weights are compared. All of the GAs in this figure were run with a population size of 1000 and all other parameter other than the group size were fixed at near optimal levels. The manual results for both the Group and Binomics GA show the performance of the GA with a fixed optimal group size found using a parameter

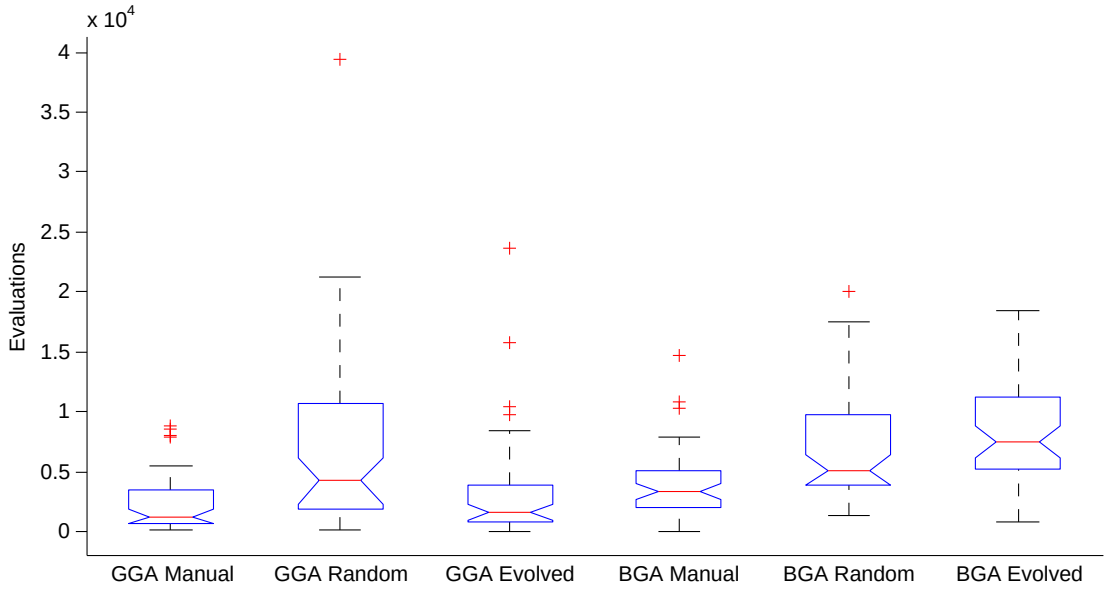

Figure 6.6: A comparison of the Group GA (GGA) and Binomics GA (BGA) with three different methods of determining group size: (1) Manual - where the optimal group size was found using a parameter sweep (2) Random - where the group size was randomly chosen during evolution and (3) Evolved where the group size was evolved. For both the Binomics and Group GA the population size was set to 1000 and all other parameters were fixed except for the group size.

sweep.

From this figure we can see that using the evolved group size method on the Group GA is more effective than using it on the Binomics GA. This can be seen by the fact that the random group size method performs better than the evolved group size method of the Binomics GA. This result combined with the fact that when the group size weight gene is initialised between 0.0 and 0.4 the Group GA performs reasonably well, but the Binomics GA performance is terrible. This leads us to believe that the selection pressure on the group size gene is stronger with the Group GA as opposed to the Binomics GA. To test this, in the next section we test both GAs on an NK landscape. This task, unlike the AE and immune tasks, is best solved when the UoE is an individual which means that we should see the group size gene steadily increase over time.

6.5.3 Evolving Group Sizes - The NK Landscape

To investigate why evolving the group size seems to work better with the Group GA than with the Binomics GA we tested both algorithms on an NK fitness landscape task (Kauffman and Johnsen, 1991; Kauffman, 1993) that has been deliberately set up so that

evolution performs best when the UoE is an individual, i.e. with a group size of one. This means that if the group size weight genes are all initialised to zero then as evolution proceeds the size of this gene should steadily increase, decreasing the size of the group.

As described in section 7.2 when evolving solutions to NK landscapes each individual in the population is a bit string of length N. This means that when using either the Group or Binomics GA the UoE is a group of bit strings and the fitness of this group is calculated as the average fitness of all the individuals in the group. Calculating fitness in this way means that as evolution proceeds there will be a selective pressure towards smaller groups because in small groups the fitness of high performing individuals is not averaged out. To determine how well an appropriate group size was evolved with both the Group and Binomics GA we compared the number of evaluations it took for the average of all the group size weight genes in the population to exceed 0.5, which is equivalent to a group size of 1.

Each GA was run on the same ten different 30-8 NK landscapes and the number of evaluations it took for the average of the group size weight genes in the population to become greater than 0.5 was recorded. The parameters used in both the Group GA and Binomics GA were as follows: population size was 100, maximum evaluations was set to 50K, infection rate was 0.5, mutation rate was 1/N, group size gene mutation standard deviation was 0.1 and the group size weight genes were initialised to 0.

For the Group GA it took approximately 5,600 evaluations (median) over 10 runs for the population average group size gene to increase above 0.5 and for the Binomics GA it took approximately 26,000 evaluations (median) which means that the Group GA reached the target five times faster. This implies that the selection pressure on the group size is much more significant when using the Group GA and that evolving the group size using the Binomics GA is not as effective which explains our results from the previous section.

6.6 Evolving ANNs where the UoE is a Group

Throughout Part I of this thesis autoencoding ANNs have been evolved where each individual in the population is a partial network. This was done by setting a proportion of each individual's genes (weights) to zero and maintaining this through evolution using add and delete weight operators. This has allowed us to test evolution where the UoE was a group of partial networks and the UoS was either an individual or a group. Up to this point we have avoided discussing the question of whether there are any benefits to evolving ANNs in this way as compared to using a standard GA where the UoE and UoS

are both individuals, because the focus has been on understanding how varying the UoE and UoS impacts evolution. To answer this, below are two examples from the literature that highlight the benefits of evolving ANNs where the UoE is a group of partial networks. We then explain how using the evolved group size GAs on ANN tasks can be used to find the optimal network scaling factor.

SANE (Moriarty and Miikkulainen, 1996, 1995), which has been discussed earlier, is a GA where the UoE is a group of neurons and the UoS are individual neurons. This is similar to the Binomics GA where the UoS is a partial network and the UoE is a group of partial networks added together to form a full network. SANE has been shown to outperform more standard evolutionary methods on an inverted pendulum ANN problem. Recent research by Geoff Hinton (Hinton and Srivastava, 2012) has also shown that deep-learning networks can be better trained using a method known as dropout, where a large proportion of the network weights (usually around 50%) are set to zero and then a number of these partial networks are combined. This is very similar to how we have evolved the AEs with the Binomics and Group GA. Both these examples provide evidence that evolving ANNs where the UoE is a group of partial networks could improve performance over standard GAs.

Another potential benefit of evolving ANNs using the Group or Binomics GA where the UoE is a full network constructed by summing the weights of a group of partial networks is related to the amount the weights are scaled. When evolving ANNs with standard GAs where the UoE and UoS are both individual, fully specified networks, it is possible that evolution will not work unless the network weights are scaled by some value. For example when evolving the the 3-12-2-12-3 AE with the Microbial GA the weights were randomly initialised between 0.0 and 10.0, but we found that performance was terrible unless the weights of each network were multiplied by 25 before being evaluated. This optimal scaling factor was only found after a time consuming parameter sweep. Using the evolved group size Group or Binomics GA, this parameter sweep can be avoided because evolving the group size is equivalent to finding an appropriate scaling factor. It is not even necessary to set a portion of the weights to zero for this to work. One could imagine having a population of fully specified networks and then using the evolved group size Binomics or Group GA to find the appropriate group size. Doing this avoids the need to test different scaling factors because when the group of individual networks are added together to be evaluated, the group size is equivalent to a scaling factor if the weights of all individuals in the population are around the same magnitude. Exploring this further is not in the

scope of this thesis, but understanding the benefits of changing the UoE and UoS when evolving ANNs is potentially a fruitful research area.

6.7 Discussion

One of the limitations of the Group and Binomics GA in their original form is that the group size needed to be set ahead of time and if the optimal group size is unknown performance of these algorithms could suffer. The goal of this chapter was to explore how varying group size affected these algorithms and then to present two methods that avoid the need to pre-set the group size before evolution begins. Below, the key results of manually varying the group size, randomly setting the group size and evolving the group size of the Binomics and Group GAs are summarised. Based on these results we then make some recommendations of the types of parameters to use when using the evolved group size method on a new task.

At the beginning of this chapter the group size was manually changed on both the Group and Binomics GAs to show the impact this parameter had on performance. The important result from these tests was that increasing the population size made both GAs more robust to changes in group size. Next we presented the random group size versions of both GAs which avoided the need to pre-set a specific group size before evolution begins. This method will increase the chances of evolving solutions to problems where the optimal group size is unknown without having to do an extensive parameters sweep. Two limitations of this method are: (1) evolution spends time at less than optimal group sizes and (2) a group size range needs to be set ahead of time.

We then showed how the group size of both the Binomics and Group GA can be evolved by adding a new group size weight gene to each individual in the population. Using the Group GA on the immune system task, as an example, we showed how this method was able to solve the task and evolve an appropriate group size even though as discussed earlier, from a pure fitness maximisation standpoint, bigger groups are always better. Next we compared the evolved group size versions of the Binomics and Group GAs on the 3-12-2-12-3 AE. Both GAs were able to evolve solutions to this AE and with appropriate parameters a near optimal group size was evolved but results showed that the Group GA actually performed better than the Binomics GA. One possible reason for this is that the selection pressure on the group size weight gene is stronger when using the Group GA than the Binomics GA, as was shown using the NK task.

When using the evolved group size versions of either the Group or Binomics GAs how

the group size weight genes are initialised in the population and how much mutation is applied during evolution significantly impacts performance. When applying these GAs to new tasks where there is no prior information about the optimal group size our results show that it probably makes sense to initialise all group size weight genes in the population to zero. Reasons for this have not been fully explored, but it may have to do with the fact that when the group size weight genes are randomly initialised, groups of very different sizes are used which could impair performance. Choosing an appropriate mutation amount for the group size weight gene is also important. For both the evolved group size versions of the Group and Binomics GAs the amount of mutation applied to group size weight gene was a normally distributed random number with mean zero and standard deviation set as a parameter. If the standard deviation of the mutation amount was too high then evolution was unable to converge on large group sizes and if it was too low then evolving the appropriate group size took too long. As a rule of thumb, setting the standard deviation of the mutation amount between 1/POP and 10/POP is probably a good place to start. It also makes sense to set the population reasonably large because as we have shown, on both the immune system and AE tasks, bigger populations are more robust to a variety of different group sizes.

A different method of genetically encoding the group size gene is to set the group size gene as a whole number and then invert this gene when constructing the groups. This would allow the group size gene to be mutated using whole numbers rather than using real numbers with mean zero and standard deviation which is a parameter. This means the mutation rate could could be set to 1 meaning that the tournaments loser(s) have their group size weight gene increased or decreased by 1. This could reduce the sensitivity of the evolved group size GAs to mutation and would also make the mutation rate more understandable.

Both the evolved and random group size version of the Binomics and Group GAs avoid the need to set a specific group size before evolution begins which reduces the amount of parameter testing that needs to be done. This makes both GAs useful tools for solving tasks where the optimal devision of labour is unknown.

Experiments II: Adding Gene Shuffling to Genetic Algorithms

In Experiments I of this thesis we presented two GAs that were developed by varying the relationship between the level of evaluation and selection. In Experiments II we focus on the latter half of the evolutionary cycle presented in Chapter 3 (which is reviewed below) that describes how new population members are generated from the UoS. Specifically, we focus on the effects of implementing massive amounts of horizontal gene transfer, showing how this can improve evolutionary performance on a number of different landscapes.

A common theme that ties the GAs in Experiments I and II together is the influence of bacterial evolution and metagenomics. In Experiments I, we showed demonstrated how group level evaluation allowed evolution to find symbiotic solutions to tasks. This group level evaluation was inspired by recent Metagenomic research that the line between the individual and the group in bacterial colonies is a very fuzzy one. In Experiments II we experiment with horizontal gene transfer, which is found in bacteria, and show its benefits when applied to GAs.

The first chapter in Experiments Part II introduces the Unconstrained GA (UGA), which implements horizontal gene transfer, and shows how it outperforms a standard GA (SGA) on landscapes with a lot of local optimum. Then in the next chapter we investigate this further, to try to figure out why this is and test how generalisable this result is to other landscapes and tasks. The work in this part of the thesis extends on the results presented in Tomko et al. (2013).

The majority of results in Experiments II are comparisons between the best performing parameter combination of different GAs on a variety of landscapes. Results of the parameters sweeps performed on the different GAs can be found online at ntomko.wordpress.com. For each different landscape, the results of testing different parameters combination on all GAs are shown. The best performing parameter combinations results for each GA are highlighted in red.

Chapter 7

The Unconstrained GA

7.1 Introduction

A common problem with using GAs to solve complex tasks is their tendency to get stuck on sub-optimal solutions (local optima). Increasing the mutation rate and/or population size increases the amount of evolutionary exploration, reducing the chances of getting stuck on a local optimum but does so at the expense of potential exploitation of good solutions. High mutation rates also risk knocking the population off a fit solution. To overcome this issue different diversity methods have been developed and applied to GAs (see Dick (2005) or Mahfoud (1995) for an overview). Here we present the Unconstrained GA (UGA) which is a novel GA that was developed by varying how new UoE are constructed from the UoS, and show how it outperforms a standard GA on highly rugged (yet non-random) landscapes with a large number of local optima. The process of constructing UoE from UoS is described by steps 4 and 5 of the evolutionary cycle which was first introduced in Chapter 3, but we re-introduce it below and in figure 7.1.

1. Evaluate the UoE according to the pre-defined fitness function
2. Assign fitness credits from the UoE to UoS
3. Select the UoS on the basis of the assigned credits
4. Use the selected UoS to produce the next generation of offspring
5. Use the offspring to generate new UoEs

In standard generational GAs (SGA), after all the population members have been evaluated, the fitter ones are selected to become parents for the next generation of offspring.

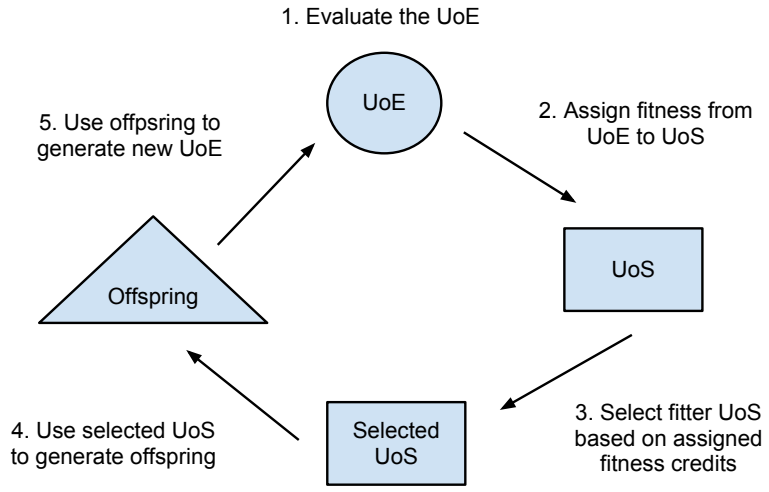

Figure 7.1: A flow diagram showing how the units of evaluation (UoE) and units of selection (UoS) interact in artificial evolution. In this figure the different evolutionary entities (UoE, UoS and offspring) are represented by different shapes (circle, rectangle and triangle respectively).

In the UGA the offspring are generated in the exact same way, but before this new population is evaluated the genes of the entire population are shuffled. This gene shuffling step can be thought of as being equivalent to implementing massive amounts of horizontal gene transfer (HGT). In nature, HGT mainly occurs in microbial communities such as bacterial colonies and is the process by which genes are transferred between individuals (Ochman et al., 2000). There are many different ways bacteria transfer genes horizontally but some of the most common include the use of plasmids or bacteriophages. Vertical gene transfer (VGT) occurs during sexual reproduction when the genes of two parents are recombined to produce one or more offspring. In most standard generation GAs gene transfer is limited to VGT but the UGA uses both HGT and VGT. Below we compare the UGA to an SGA on NK and NKp landscapes (Kauffman, 1993; Barnett, 1998) as well as an autoencoding (AE) ANN task. NK and NKp landscapes are well known benchmark tasks in the GA community, the difference between them is NKp landscapes have neutrality, while standard NK landscapes do not. There are good reasons to expect many fitness landscapes in both natural (biological) and artificial scenarios to have such neutrality and therefore it is important to compare the GAs on this type of landscape. The results show that the UGA outperforms the SGA on the more rugged, less neutral landscapes that have more local optima. The GAs were also compared on the AE task because it is more of a real world task, has a lot of neutrality and likely has a very rugged fitness landscape.

7.2 Tasks Used in this Chapter

NK landscapes (Kauffman and Johnsen, 1991; Kauffman, 1993) are tuneable, rugged, binary landscapes where N defines the dimensionality or number of bits (genes) in the landscape and K defines how rugged the fitness landscape is, where a more rugged landscape has more local optima. In both NK and NKp landscapes the amount of ruggedness (K) is controlled by changing the amount of genetic epistasis, where epistasis can be generally defined as the degree of interaction or cross-coupling between genes. The higher the K, the more genes have an impact on any individual gene's fitness contribution (Kauffman and Johnsen, 1991). Along with increasing the number of fitness peaks in the landscape, increasing K also increases the steepness of these peaks and reduces their height (Kauffman and Johnsen, 1991). If K is set to zero then there is no epistatic interaction between genes and the fitness landscape can be classified as a 'Mount Fuji' landscape which has a single global fitness optimum and no local optima. If K is set to 2 then the fitness contribution of any given gene depends on the fitness of its two direct neighbors - this

epistatic interaction increases the ruggedness of the landscape by adding local optima to it. The maximum value K can be set to is $N-1$, in this case the fully correlated genotype gives rise to a completely random fitness landscape (Kauffman and Johnsen, 1991).

NKp landscapes (Barnett, 1998) introduce neutrality into NK landscapes. One way to picture a neutral landscape is in terms of plateaus connecting what would be local optima in a non-neutral landscape. In NKp landscapes, the amount of neutrality is controlled by parameter p which can be set between 0 and 1, where a higher p corresponds to more neutrality. If p is set to zero then the NKp landscape is equal to a normal NK landscape and if p is set to 1 then the landscape is completely flat with no optima (Barnett, 1998).

A general procedure for generating an NK landscape, which we use in this thesis, is summarized in Mayley (1996):

1. Generate N look-up tables, one for each locus

2. Each look-up table has $2^{(K+1)}$ entries that are randomly generated between [0, 1]

3. The fitness of a given locus, $f(n)$ is found by taking the specific locus and K neighbors and finding the corresponding entry in the nth look up table.

4. The total fitness is equal to the average of all the loci fitnesses:

$$F(N,K) = \frac{1}{N}\sum f(n)$$

For example, if N=4 and K=2 then there will be four look-up tables each containing 8 entries. To find the fitness of the second locus in genotype [1 0 1 1] then one looks for the [1 0 1] entry of the second look-up table.

To generate an NKp landscape from an NK landscape a proportion of the entries in the look-up table, defined by p, are set to zero. So if p = 0.90 then a random chosen 90% of the entries in the look-up tables are set to zero.

We chose the NK and NKp landscapes as a test-bed for the UGA because the dimensionality (N), ruggedness (K), and neutrality (p) can be easily varied which allows us to test the UGA on a wide variety of landscapes in a reasonable amount of time. NK landscapes are also a well-known benchmark in the GA community and have been shown to be a good task to test the behaviour and performance of GAs (Aguirre and Tanaka, 2003).

7.3 The Unconstrained GA

The difference between the Unconstrained GA (UGA) (see Figure 7.2) and most standard generational GAs (SGA) is that the UGA includes both horizontal (HGT) and vertical gene transfer (VGT) while most standard GAs only apply VGT. In GAs, HGT can be simulated in a variety of different ways. For example, one could implement HGT as a transfer of genes where an individual in the population transfers one or more genes to another individual and in the process overwrites the recipient's gene(s). In this case, there would be an increase in the frequency of the genes transferred and a decrease in frequency of the genes that were overwritten. An alternative way to implement HGT would be as a gene swap, rather than a gene transfer. In this case, two individuals swap or trade genes so there is no change of gene frequency in the population. In the UGA we have chosen to implement HGT as gene swapping instead of gene transferring to ensure that there is no change in gene frequency in the population as a result of this process. To efficiently implement massive amounts of gene swapping, in every generation, the genes of each locus are shuffled. Another way to think about this process is that all the genes of a specific locus are put into bags and then randomly withdrawn to reconstruct the full genotypes in the population.

A single generation of the UGA is carried out as follows. ELITENUM is a parameter that controls how many of the fittest individuals are automatically copied into the next generation.

1. Evaluate the fitness of each individual genotype in the population
2. Rank each genotype according to fitness
3. The fittest ELITENUM individuals are automatically copied into the new offspring population unchanged
4. The remaining POPSIZE-ELITENUM offspring are generated as follows:
   (a) Select two parents from the population using the following tournament selection method:
      i. Randomly pick two individual genotypes from the population
      ii. Compare the fitness of the two individuals from the population with the fitter becoming a parent
      iii. If the fitness of the two individuals is the same then randomly pick a winner

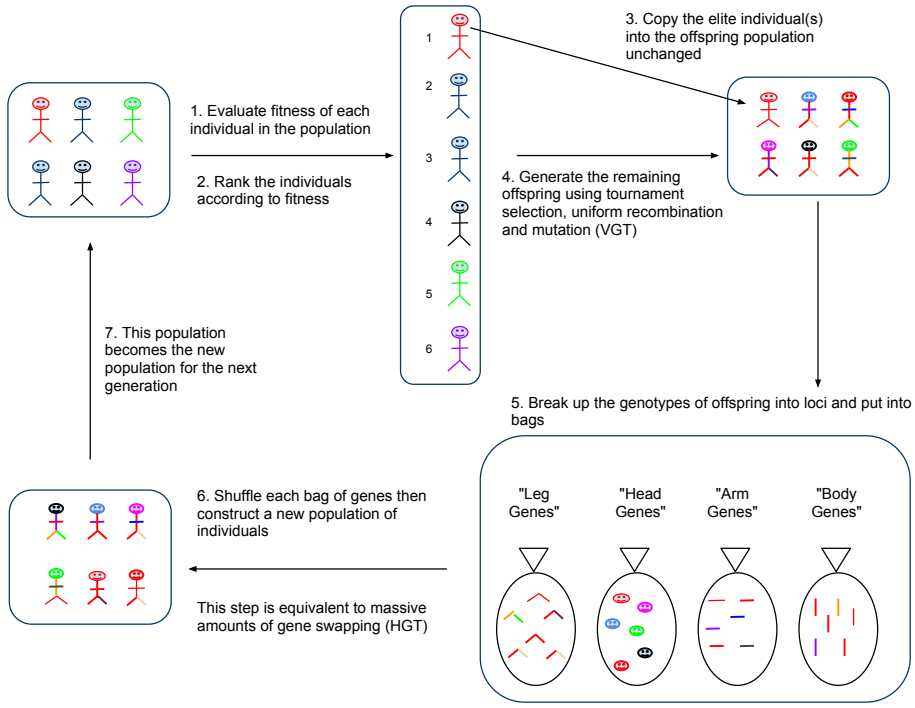

Figure 7.2: A general flow diagram showing the main steps of the UGA.

iv. Repeat steps i-iii to select a second parent

(b) Produce a single offspring from these two parents by randomly choosing a single gene from either parent at each locus (this is known as uniform recombination)

(c) Mutate this offspring at each locus with a probability of MUT (the mutation rate)

(d) Add this offspring to the new offspring population

5. Break up the genotypes of each offspring and put them into separate locus bags

6. Construct a new population by randomly choosing genes from each bag (this is equivalent to the population engaging in promiscuous HGT)

7. This new population becomes the population for the next generation

VGT occurs in step 4 where two parents are uniformly recombined to produce a single offspring, and horizontal gene shuffling is done in step 6. In an SGA, if elitism is used it conserves the fittest genotypes in the population by automatically copying the fittest individuals into the offspring population without mutating them. In the UGA, elitism is carried out in step 3, but because of the gene shuffling of step 6, the fittest genotypes are not conserved but rather the genes of the fittest individuals are conserved. The reason for presenting the UGA as above is so that it can easily be seen that when steps 5 and 6

are eliminated, the UGA becomes an SGA with no gene shuffling, i.e., with tournament selection and uniform recombination.

7.4 Results

7.4.1 UGA Compared to SGA on NK and NKp Landscapes

Here the performance of the UGA and SGA are compared on a variety of NK and NKp landscapes. The SGA used for comparison is exactly like the UGA described in the previous section except that steps 5 and 6, which implement gene shuffling, are eliminated (see section 2.1.7 for a detailed description of the SGA). For each family of landscapes, where a family refers to landscapes with the same N, K, and p parameters, performance was measured as the median of the maximum fitness reached on 50 different randomly generated landscapes of a given family over a fixed number of evaluations. To ensure a consistent comparison, the same 50 randomly generated landscapes were tested on both GAs.

The results presented in this section are the best performing parameter combinations of each GA on the different landscapes found with a parameter sweep. The population size was varied between 10 and 1000 and the mutation rate was varied between $0.1/N$ and $2/N$ where this mutation rate corresponds to the probability of flipping a bit at a given locus. In both GAs, elitism percentages of 0 and 5% were tested. An elitism percentage means that the number of elite individuals automatically copied to the next generation is equal to $ceil(POPSIZE * ELITEPCT)$ where ceil rounds up to the nearest higher integer. For example if the population size is 10 and the elite percent is 5% then the number of elite individuals (ELITENUM) is 1. The parameter combinations for the results presented in the following tables can be found online at ntomko.wordpress.com.

The NK landscapes tested were N30 K4, N30 K6 and N30 K8 and the NKp landscapes tested were N30 K4 p0.99, N30 K8 p0.99, N100 K8 p0.99, N100 K8 p0.90. Statistical comparisons between the two GAs were made using p-values from the Wilcoxon rank sum test of equal medians and Bayes factors (see table 7.1).

These results show that on the non-neutral NK landscapes, as K is increased the UGA starts outperforming the SGA. On the NKp landscapes there is no significant difference between the GAs except on the least neutral of them, that is the N100 K8 p0.90 landscape, Here, the UGA outperforms the SGA. This implies that high amounts of neutrality eliminates the benefit the UGA has over the SGA on non-neutral, rugged landscapes.

Table 7.1: Comparison of the Unconstrained GA (UGA) and a standard GA (SGA) on a variety of different NK and NKp landscapes. Fitness is measured in terms of median and IQR maximum fitness over 50 runs and statistical comparisons are made using Bayes Factors and p-values from the Wilcoxon rank sum test of equal medians.

Landscape	UGA Median (IQR)	SGA Median (IQR)	Bayes Factor	p-value
N30 K4	0.7708 (0.0269)	0.7663 (0.0267)	5	0.5
N30 K6	0.7739 (0.0258)	0.7670 (0.0263)	0.6	0.02
N30 K8	0.7709 (0.0208)	0.7635 (0.0208)	**0.01**	$\mathbf{1 * 10^{-4}}$
N30 K4 p99	0.0854 (0.0360)	0.0851 (0.0439)	6	0.6
N30 K8 p99	0.1345 (0.0183)	0.1278 (0.0245)	3	0.2
N100 K8 p99	0.1126 (0.0121)	0.1109 (0.0104)	3	0.2
N100 K8 p90	0.2600 (0.0176)	0.2518 (0.0235)	**0.2**	**0.007**

7.4.2 UGA Compared to SGA on the Autoencoder Task

Here the UGA and SGA are compared on an AE task that was used in earlier in the thesis. Figure 7.3 compares the UGA to the SGA (after a doing parameter sweep of both GAs) over 50 runs on a 4-24-3-24-4 AE task. Performance was measured in terms of the number of evaluations it took each GA to evolve a perfect AE, so unlike the NK and NKp results, lower is better. Using both the Bayes factor and p-value to compare the performance of these GAs shows that on this task the null hypothesis cannot be rejected and therefore we cannot say there is any significant difference in performance (p=0.1 and Bayes factor = 2). One possible reason for this is that like the NKp landscapes, the AE networks have a lot of neutrality and so the benefits of gene shuffling are reduced.

7.4.3 The UGA Compared to Gene Pool Recombination

In order to determine whether the performance difference between the UGA and the SGA on the rugged NK landscapes is due to HGT in general or the specific type of HGT implemented in the UGA we compare the UGA to Gene Pool Recombination (GPR) (Muhlenbein and Voigt, 1995). Based on the general description given in their paper, the main difference between UGA and GPR is that GPR constructs a new population by randomly choosing genes from the offspring population with replacement instead of shuffling the genes without replacement like in the UGA. This means that unlike the UGA, the gene frequencies can change during this step. Here we compare UGA to a

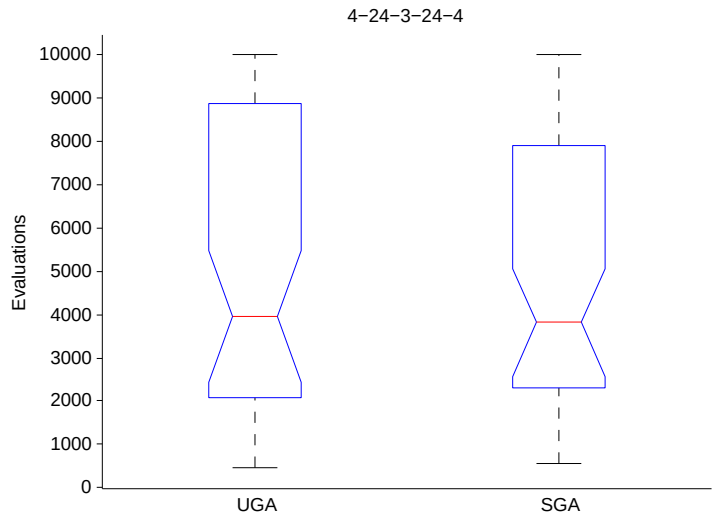

Figure 7.3: A box and whisker plot comparing the performance of the Unconstrained GA (UGA) to a standard generational GA (SGA) on the 4-24-3-24-4 autoencoder (AE). Performance was measured in terms of the number of evaluations it took to evolve a perfect AE over 50 runs.

modified version of the UGA, which we call GPR, where the new population is generated by randomly choosing genes from the offspring population. While this modified UGA may not be identical to GPR, it does allow us to determine what effect different types of HGT have on evolution.

Table 7.2 compares these two GAs on a variety of NK and NKp landscapes as well as the 4-24-3-24-4 AE. Performance was measured in the exact same way as the SGA and UGA comparisons in the previous sections. Again comparisons were made between the best performing parameter combinations of each GA, results from the parameters sweep can be found online at ntomko.wordpress.com.

These results show that on the both the non-neutral NK landscapes and on the AE task there is no significant difference between the two algorithms according to the statistical measures we have used. On the other hand, on all of the NKp landscapes, except for the N30 K4 p0.99 landscape, the UGA significantly outperforms GPR. This implies that on highly epistatic landscapes with a lot of neutrality, shuffling genes with replacement impairs evolutionary performance.

Table 7.2: The UGA compared to GPR on a variety of different NK and NKp landscapes as well as the 4-24-3-24-4 autoencoder. Performance was measured in terms of the median and IQR maximum fitness over 50 landscapes on the NK and NKp tasks and in terms of median evaluations it took to evolve a perfect autoencoder over 50 runs.. Statistical comparisons were made using Bayes Factors and p-values calculated using the Wilcoxon rank-sum test for equal medians.

Landscape	UGA Median (IQR)	GPR Median (IQR)	Bayes Factor	p-value
N30 K8	0.7709 (0.0208)	0.7653 (0.0262)	0.6	0.02
N30 K6	0.7739 (0.0258)	0.7694 (0.0236)	3	0.2
N30 K4	0.7708 (0.0269)	0.7679 (0.0267)	5.6	0.6
N100 K8 p90	0.2600 (0.0176)	0.2465 (0.0200)	**0.001**	$\mathbf{8 * 10^{-5}}$
N100 K8 p99	0.1126 (0.0121)	0.1065 (0.0145)	**0.08**	**0.02**
N30 K8 p99	0.1344 (0.0183)	0.1225 (0.0213)	**0.06**	**0.01**
N30 K4 p99	0.0845 (0.0360)	0.0774 (0.0352)	2	0.1
4-24-3-24-4	3815 (5468)	5665 (7148)	2	0.1

7.5 Discussion

This chapter introduced the UGA which is a novel generation GA that implements both VGT and HGT to produce new UoE from the UoS. The major difference between the UGA and more standard generational GAs (SGAs) is that most SGAs use only VGT. The UGA was compared to an SGA on both NK and NKp landscapes and an AE task. The difference between NK and NKp landscapes is that the NKp landscapes add neutrality. There is also significant amounts of neutrality in the AE task. This is because there are a large number of different combinations of connection weights that result in different networks with the same fitness. As mentioned earlier, there is good reason to believe that neutrality exists in many real wold optimisation problems and in natural evolution so we felt it important to see what effect it had on the UGA.

Results from this chapter show that on highly rugged (epistatic) landscapes with limited neutrality the UGA outperformed the SGA. On the tasks with neutrality (NKp and AE) the performance difference between the UGA and SGA was statistically insignificant. Our initial explanation for this difference in performance on the epistatic landscapes is that the gene shuffling maintains diversity in the population which reduces the chances that it will get stuck on a local optimum. The reason this advantage disappears when

neutrality is introduced could be because neutral landscapes do not have any local fitness optimum, instead they have fitness plateaus. This hypothesis will be explored further in the next chapter.

The UGA is related to GAs that implement some sort of multi-individual recombination, GAs inspired by bacterial evolution and diversity maintaining GAs all of which have been summarised in the Literature Review Chapter. As already discussed, Gene Pool Recombination (GPR) (Muhlenbein and Voigt, 1995) is probably the most similar to the UGA so we compared the two algorithms on a variety of NK and NKp landscapes. On the NK landscapes and on the AE there was no significant difference between the UGA and GPR according to our statistical measures, but on the neutral NKp landscapes with a high amount of epistasis (K=8), the UGA outperformed GPR. The only difference between the UGA and the version of GPR tested in this chapter is how HGT is implemented. In the UGA, HGT is done using gene shuffling (no replacement) which means that HGT does not change the gene frequencies of the population, all that is changed is which gene belongs to which individual. In GPR, HGT is done with replacement which means that there is a chance that the gene frequencies will change.

For some reason, implementing HGT in the same way it is implemented in GPR reduces performance on neutral, epistatic NKp landscapes but has no effect on the non-neutral NK landscapes or NKp landscapes with low K as compared the performance of the UGA. There was also no significant difference between these algorithms on the AE task. The AE task has significant amounts of neutrality but it is difficult to determine how epistatic it is. Assuming that the UGA and GPR perform about the same on task with low amounts of epistasis then these results imply that the AE task is not very epistatic, at least compared to NK and NKp tasks with K set to 8.

In summary we have tried to explore what massive amounts of HGT buys you when it is used to create new UoE and found that on rugged landscapes with no neutrality it improved performance. In the next chapter we try to understand why the UGA outperforms the SGA on rugged landscapes, focusing in on how elitism affects performance. We also compare the UGA to GPR on a couple of other tasks to try to determine when one method of implementing HGT is better than the other.

Chapter 8

The UGA and Elitism

8.1 Introduction

Last chapter we showed that the UGA outperforms a standard generational GA (SGA) on rugged, non-neutral NK landscapes. Our hypothesis was that the reason for this is that gene shuffling maintains diversity in the population, reducing the chances of the population getting stuck on local optima. We now further test this hypothesis as well as testing the UGA on both a two-peak and a Royal Road landscape as well as the autoencoding (AE) task.

One of the issues explored in this chapter is the role of elitism in the UGA. Elitism is the process of conserving the genotypes of the fittest member of the population from generation to generation. This means that the fittest individuals in the population are automatically copied to the next generation. The benefits are that it ensures that best performing individuals are kept from being modified by recombination and mutation but, on the other hand, it can also cause the population to converge on a suboptimal solution. In the UGA, when elitism is implemented it is not the genotypes as a whole that are maintained but rather the genes of the fittest individuals. This is because the shuffling step of the UGA breaks up the genotype of the elite individual every generation. As we will show, on some landscapes shuffling the genes of just the elite individuals in the population improves evolution.

This chapter is structured as follows. We first determine the effects of reducing the number of individuals whose genes are shuffled in the UGA. We then test two modified versions of the UGA. The first shuffles the genes of only the elite members of the population while the second shuffles the genes of all individuals but the elite individuals. Finally we test all the GAs on a two-peak, Royal Road and AE task to further determine the effects

of gene shuffling on evolution.

Even though the main focus of this chapter is to investigate the UGA and why it performs well on rugged landscapes, it is important to keep in mind how this work fits in with the overall themes of the thesis. All the GAs presented in this chapter and the previous chapter were developed using the UoE/UoS view of evolution. Specifically, these GAs vary how new UoE are constructed from UoS (steps 4 and 5 of the evolutionary cycle). So like the Group and Binomics GAs presented in Part I of this thesis, these GAs are even more examples of ways our view of evolution can be used to develop novel GAs.

8.2 Tasks Used in this Chapter

In this chapter we explore the reasons why the UGA performs well on rugged landscapes using the NK and NKp landscapes described in the previous chapter, an AE task (section 5.2) as well as on a two-peak and Royal Road landscape which are described here.

8.2.1 The Two-Peak Landscape

The two-peak landscape is a simple landscape that has two optima, a high peak and low peak. We developed this landscape to explore how well different GAs are able to move the population off the lesser optimum on to the global optimum. For a given two-peak landscape, the length of the genotype (N) and the height of the global optimum ($HighPeak$) need to be specified, where the height of the global optimum must be greater than $N/2$. The height of the lower optimum ($LowPeak$) is calculated as $N - HighPeak$ and the fitness of a given binary genotype is calculated as follows.

- if the number of 1's in the genotype is greater than or equal to $LowPeak$ then fitness is equal to $NumOnes - LowPeak$
- if the number of 1's in the genotype is less than $LowPeak$ then the fitness is equal to $LowPeak - NumOnes$

For example, for a 9-1 two peak landscape, $N = 10$, $HighPeak = 9$ and $LowPeak = 1$. The fittest genotype on this landscape is [1 1 1 1 1 1 1 1 1 1] which has a fitness of 9, the least fit genotype is any bit string where $NumOnes = LowPeak$ which will have a fitness of zero and a genotype of all zeros has a fitness equal to the height of the low peak.

8.2.2 The Royal Road Landscape

The Royal Road Landscape, like the NK and NKp landscapes, is a well known benchmark landscape in the GA community. It was first used by Mitchell et al. (1992) to explore the relationship of building blocks and crossover in artificial evolution. We use it to understand whether the results on the NK, NKp, and two-peak landscapes can be extended to other landscapes.

A specific Royal Road landscape is defined using two parameters, string length and schema length. The string length defines the total length of the landscape and the schema length defines the length of the fundamental building block or intermediate fitness levels of the landscape. In these landscapes, fitness is only credited for fully formed schemas. For example, in a simple 8-2 landscape, the string length of the landscape is 8-bits and the schema length is 2-bits. As per Mitchell et al. (1992) the fitness of any bit string is calculated as:

$$F(x) = \sum_{s \epsilon S} c_s \sigma_s(x)$$

Where x is a bit string, S is the set of schemas s1, s2, ... sn, c_s = order(s) and

$$\sigma_s = \begin{cases} 1 & \text{if } x \text{ is an instance of } s \\ & 0 \text{ otherwise} \end{cases}$$

This means that for the 8-2 landscape the fitness of the optimal bit string [1 1 1 1 1 1 1 1] is equal to $2 * 4 + 4 * 2 + 8 * 1 = 32$. Because fitness credits are only awarded for fully formed blocks or schemas the Royal Road landscapes have a lot of neutrality. For example, in the 8-2 landscape strings [1 1 0 0 0 0 0 0], [1 1 0 1 0 0 0 0] and [0 0 0 1 0 0 1 1] all have a fitness score of 2 even though they are all genetically different. These type of landscapes are also very epistatic because flipping a bit can destroy a fitness scoring schema, significantly reducing the fitness score of the individual.

8.3 Varying the Amount of Gene Shuffling

The hypothesis based on the results from the previous chapter was that the reason the UGA outperforms the SGA on rugged, non-neutral NK landscapes is that gene shuffling maintains diversity in the population, reducing the chance that the population gets stuck on a sub-optimal fitness peak. If this hypothesis is correct then reducing the number of individuals whose genes are shuffled each generation should reduce performance.

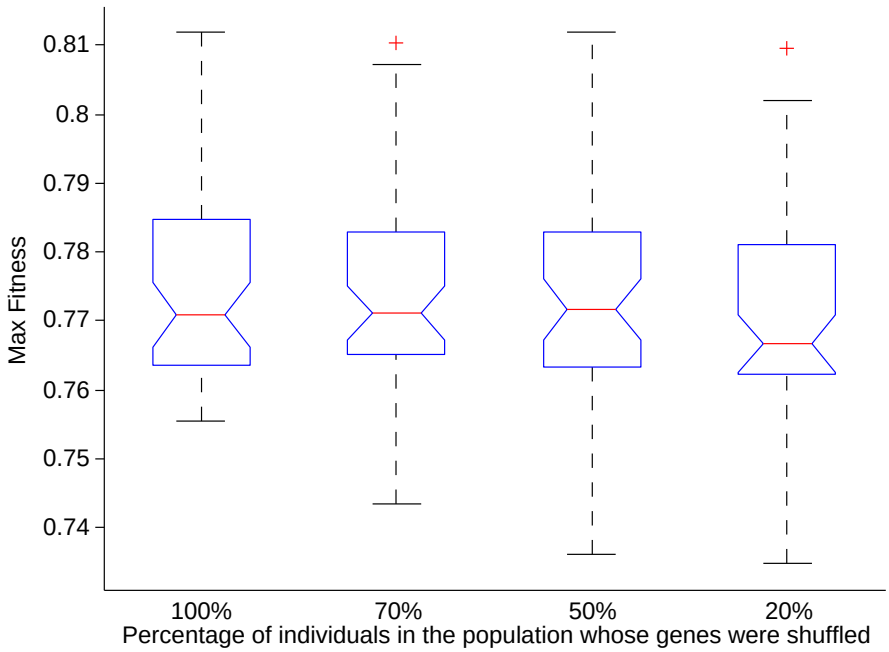

Figure 8.1: A box and whisker plot showing how varying the number of individuals whose genes are shuffled effects performance of the UGA on the N30 K8 landscape. The case where 100% of the population's genes are shuffled is equivalent to the original UGA.

To test this, a modified version of the UGA was run on the N30 K8 landscape where the number of individuals whose genes were shuffled was controlled by a parameter. In figure 8.1, the original UGA, where the entire population's genes are shuffled is compared to a modified UGA where 70%, 50% and 20% of the populations' genes were shuffled. This was done by randomly selecting the appropriate percentage of individuals from the population and only shuffling their genes. Statistical comparisons between the 100% case and the 20% case ($p = 0.2$ and Bayes factor $= 3$) show that performance is not significantly impacted by reducing the number of individuals whose genes are shuffled, which is not what was expected.

Interestingly, our results are similar to those in Eiben and Schippers (1996), who found that even though multi-parent uniform recombination improved evolutionary performance on NK landscapes, increasing the number of parents did not further increase performance. Their hypothesis was that the benefit of multi-parent recombination is a result of more gene mixing and a bigger sample size to base decisions on, but they could not explain why increasing the number of parents did not increase performance.

8.4 Elite Gene Sprinkling

Upon further investigation it was found that the key to the performance of the UGA on the rugged NK landscapes was not the amount of gene shuffling that occurred but instead was dependent on whether or not the genes of the elite individual(s) in the population were shuffled. This was tested by modifying the UGA so that instead of shuffling the genes of the entire offspring population, the genes of only the elite individuals were sprinkled through the population after the new offspring population was generated. The ESGA is illustrated in figure 8.2 and can be described follows:

1. Calculate the fitness of each individual in the population
2. Rank the individuals according to fitness
3. The genes of the fittest ELITENUM individuals are stored
4. A new population of offspring are generated as follows
   (a) Select two parents using the following tournament selection method
      i. Randomly pick two individuals from the population
      ii. Compare the fitness of the two individuals with the fitter becoming a parent
      iii. If the fitness of the two individuals is the same then randomly pick a winner
      iv. Repeat steps i-iii to select a second parent
   (b) Produce a single offspring from these two parents by randomly choosing a single gene from either parent at each locus (this is known as uniform recombination)
   (c) Mutate the offspring
   (d) Add this offspring to the new population
   (e) Repeat steps a-d until a POPSIZE number of offspring have been produced
5. Sprinkle all the genes from the elite individuals through the population by replacing randomly chosen genes in the population

Compared to the UGA, in the ESGA, only the genes of the elite individuals in the population are shuffled and sprinkled through the offspring population. Another difference is that unlike the UGA, the gene shuffling step of the ESGA does change population gene frequencies because the genes of the elite individual(s) are copied over other genes in the population.

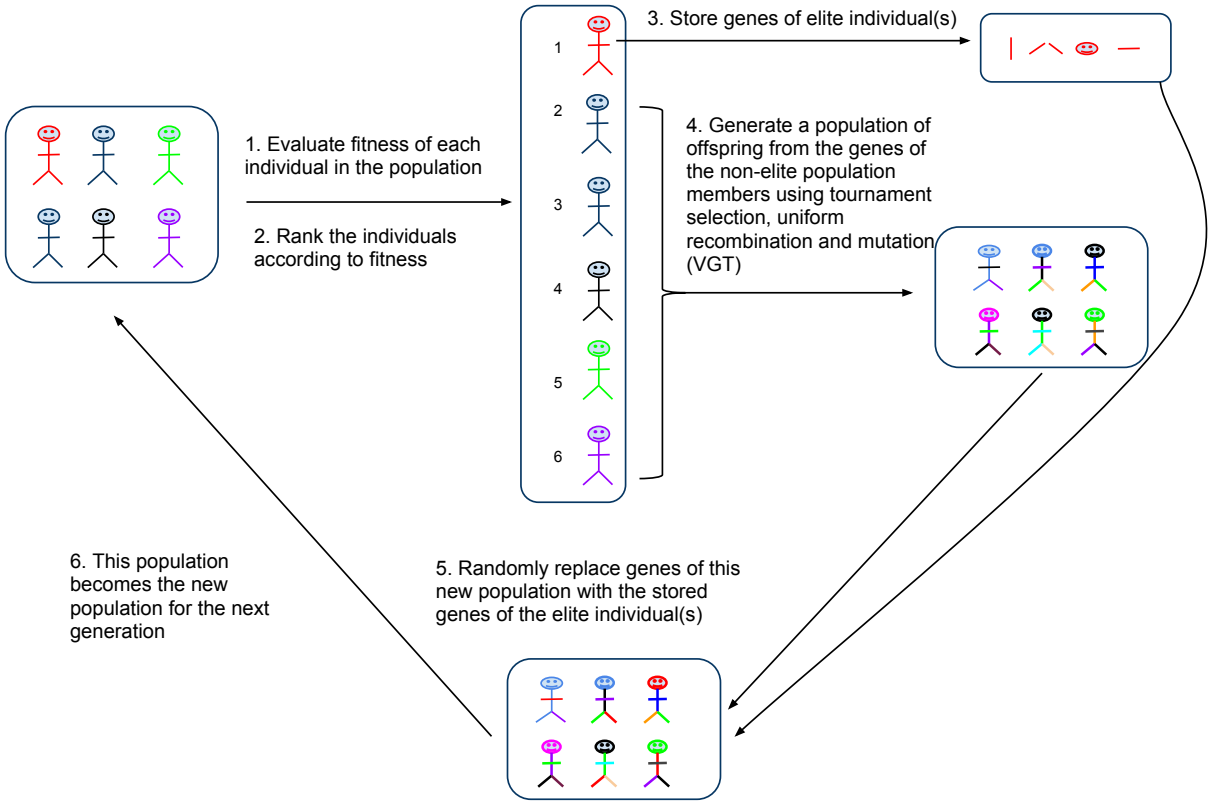

Figure 8.2: An illustration of the main steps of the Elite gene Sprinkling GA (ESGA).

The ESGA was compared to the UGA on N30 K4, K6 and K8 NK landscapes and on the N30 K4 p0.99, N30 K8 p0.99, N100 K8 p0.90, and N100 K8 p0.90 NKp landscapes. A parameter sweep was done to find the best parameter combination for the ESGA on each landscape but it ended up that the best parameters for the ESGA were the same as the best parameters for the UGA, which are listed online at ntomko.wordpress.com.

The results in table 8.1 show that according to the statistical tests used there was no significant difference between the UGA and the ESGA on any of the NK or NKp landscapes. This provides evidence in favour of the revised hypothesis that the UGA outperforms the SGA on rugged landscapes because it shuffles the genes of the elite individuals through the population.

8.4.1 UGA Without Elite Shuffling

To further confirm the benefit of elite gene shuffling on the NK landscapes, a modified UGA was tested where only the genes of the non-elite members of the population were shuffled. This UGAnES (UGA no Elite Shuffling) was tested on the N30 K8 and N100 K8 p0.90 family of landscapes which were two of the landscapes the UGA significantly outperformed the SGA on. If elite gene shuffling is responsible for improved performance

Table 8.1: The UGA compared to the ESGA on a variety of different NK and NKp landscapes. Performance was measured in terms of the median and IQR maximum fitness over 50 landscapes. Statistical comparisons were made using Bayes Factors and p-values calculated using the Kruskal-Wallis test for equal medians.

Landscape	UGA Median (IQR)	ESGA Median (IQR)	Bayes Factor	p-value
N30 K8	0.7709 (0.0208)	0.7684 (0.0215)	4	0.3
N30 K6	0.7739 (0.0258)	0.7731 (0.0252)	7	1
N30 K4	0.7708 (0.0269)	0.7675 (0.0228)	6	0.8
N100 K8 p90	0.2600 (0.0176)	0.2559 (0.0267)	2	0.09
N100 K8 p99	0.1126 (0.0121)	0.1111 (0.0109)	6	0.6
N30 K8 p99	0.1344 (0.0183)	0.1303 (0.0193)	6	0.6
N30 K4 p99	0.08540 (0.0360)	0.08665 (0.0385)	6	0.9

then the UGAnES should be significantly worse on these landscapes.

Figure 8.3 compares the performance of the UGA and UGAnES on the N30 K8 and N100 K8 p0.90 landscapes. Like the results presented in the previous sections, performance was calculated as the median of the maximum fitness score reached over 50 different landscapes of a given family and statistical tests were done using p-values and Bayes Factors. Except for the number of individuals being shuffled, the parameters of the two GAs were identical so that we could study the effect of shuffling the genes of the elite individuals. On the N30 K8 landscape the p-value was $8 * 10^{-9}$ and the Bayes Factor was $1 * 10^{-8}$, and on the N100 K8 p0.90 landscape the p-value and Bayes Factor was $4 * 10^{-5}$ and 0.004 respectively. These statistical tests confirm that on these two landscapes the UGA is significantly better than the UGAnES which provides more evidence that elite gene shuffling is responsible for the performance of the UGA on the high K, low p NK landscapes.

8.5 Testing Elite Gene Sprinkling on Other Landscapes

Based on the results of the previous section, our revised hypothesis is that the UGA outperforms the SGA on highly epistatic, non-neutral NK landscapes because the genes of the elite individuals get shuffled, reducing the chance that elitism causes the population to get trapped on a local optimum. To see if this hypothesis holds on other landscapes the UGA, SGA, ESGA, UGAnES and GPR are compared on two-peak, Royal Road and

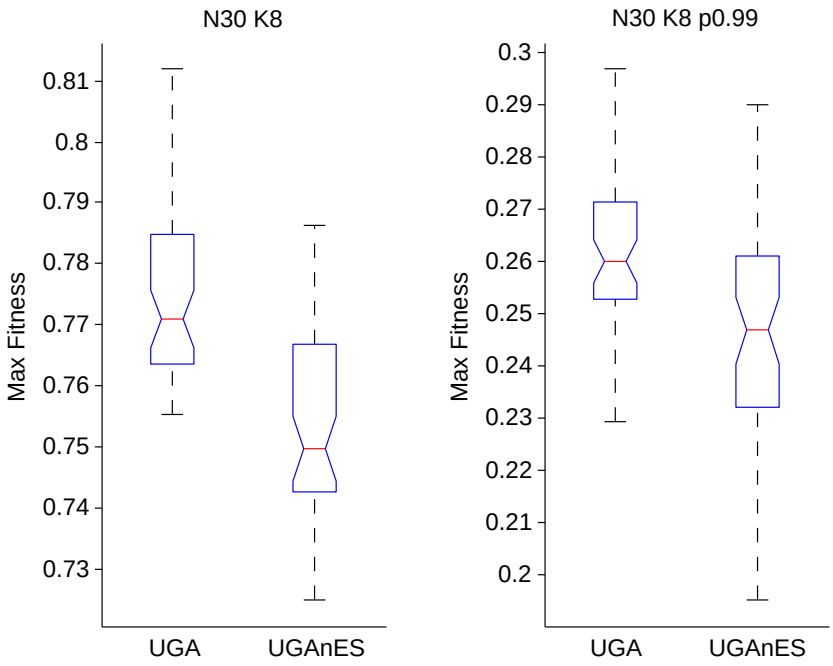

Figure 8.3: A box and whisker plot comparing the Unconstrained GA (UGA) with the Unconstrained GA no Elite Shuffling (UGAnES) on a N30 K8 and N100 K8 p0.90 landscapes.

autoencoder (AE) tasks.

8.5.1 Two-Peak Landscape Results

Here the UGA, SGA, ESGA, UGAnES and GPR are tested on a 90-10 and 70-30 two-peak landscape, where the all the individuals in the population were initialised with zero genotypes so that the entire population started on the top of the local optimum. This was done to test how well each GA could escape a local optimum.

For each GA, statistics were calculated over 50 different runs, where performance was calculated as the maximum fitness reached in a fixed number of evaluations. A parameter sweep was done on each GA to find the optimal parameters, so the results of figure 8.4 show the best performing parameter combination for each GA (parameters are listed online at ntomko.wordpress.com). If a perfect solution was not found before 20 K evaluations on the 70-30 landscape and 5K evaluations on the 90-10 landscape the run was stopped and declared a failure.

The results of figure 8.4 can be summarised as follows:

- On both the 90-10 and 70-30 landscape there is strong statistical evidence that UGA significantly outperforms all the other algorithms (the UGA versus GPR on the 90-10

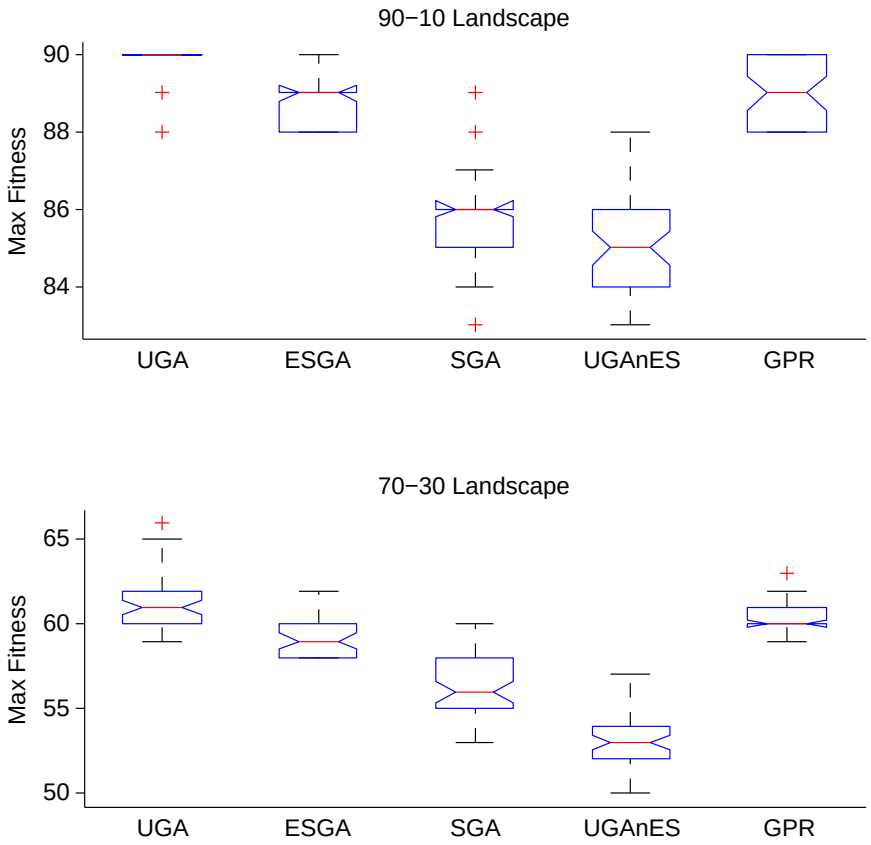

Figure 8.4: A box and whisker plot comparing the Unconstrained GA (UGA), Elite Shuffling GA (ESGA), Standard GA (SGA), UGA no Elite Shuffling (UGAnEs) and Gene Pool Recombination (GPR) on a 90-10 and 70-30 two peak landscape.

landscape results in a p-value of 4e-7 and a Bayes factor of 1e-4). The one exception is on the 70-30 landscape where it is not significantly better than GPR.

- All three GAs where the genes of the elite individual(s) are shuffled (UGA, ESGA, GPR) significantly outperform the GAs where no elite shuffling occurs (SGA, UGAnES)

In summary, on these two-peak landscapes, shuffling the genes of the elite individuals seems to help evolution move the population off the local optimum, but unlike the NK landscapes there is an added benefit to shuffling the genes in the entire population.

8.5.2 Royal Road Results

The five different GAs are now tested on a 64-4 Royal Road landscape. Performance was calculated as the number of evaluations it took each GA to evolve an optimal bit string over 50 runs, so in the plots lower values means better performance. It is important to note that this is different from how performance was calculated on the NK, NKp and Two-Peak landscapes, where fitness was calculated as the maximum fitness reached in fixed number of evaluations. Statistical comparisons were made using p-values and Bayes Factors. For each GA, a parameter sweep was performed where population size, mutation rate, and elite percentage were varied and the statistical comparisons and plot was made using the best performing parameter combination of each GA.

Figure 8.5 compares the performance of the UGA, ESGA, SGA, UGAnES and GPR on the Royal Road landscape. These results can be summarised as follows:

- Both the UGAnES and SGA, where the genes of the elite individuals are not shuffled, outperformed the other algorithms that shuffle the genes of the elite members of the population (UGA, ESGA, GPR). Although only the UGAnES was significantly better than all the other algorithms (including the SGA) according to statistical tests. For example, comparing the UGAnES to the UGA yielded a p-value of 1e-6 and a Bayes factor of 2e-4.

- The IQR of the ESGA is much greater than any of the other GAs. This is because the ESGA was only successful in solving the task in 32/50 runs while the other GAs were successful every run.

- The optimal population size of the SGA and UGAnES was very low (3 and 5 respectively) while the optimal population size for the other GAs was either 50 or 100

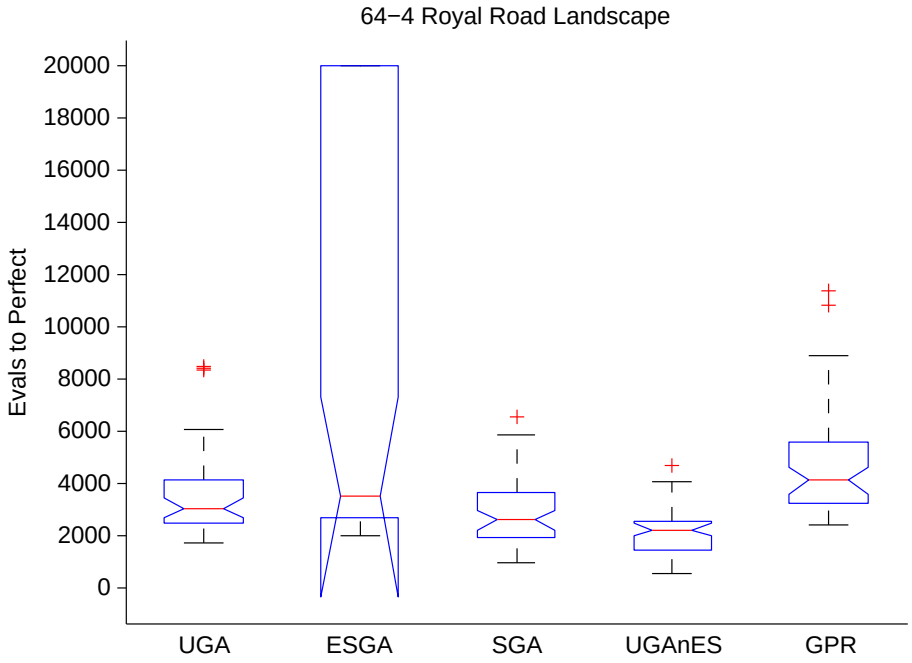

Figure 8.5: A box and whisker plot comparing the Unconstrained GA (UGA), Elite Shuffling GA (ESGA), Standard GA (SGA), UGA no Elite Shuffling (UGAnEs) and Gene Pool Recombination (GPR) on a 64-4 Royal Road landscape.

So unlike on the other landscapes tested (NK and two-peak), on this Royal Road landscape, shuffling the genes of the elite members of the population reduced performance. However, based on the fact that the UGAnES outperforms all the algorithms it does seem like shuffling the genes of the non-elite population members does improve performance.

8.5.3 The Autoencoder Results

The final landscape on which the UGA family of algorithms was tested was a 4-24-3-24-4 AE. Performance was compared over 50 runs where the number of evaluations required to evolve a perfect AE was recorded. Figure 8.6 shows that UGA and SGA performed best on this AE, but based on the results of the statistical tests the evidence is weak that there is any significant difference in performance between any of this GAs. For example, comparing the UGA and ESGA gave a p-value of 0.1 and Bayes factor of 2.

8.6 Discussion

The main goal of this chapter was to try to understand why the UGA outperformed the SGA on a rugged, non-neutral NK landscapes. To explore this we tested the SGA, UGA

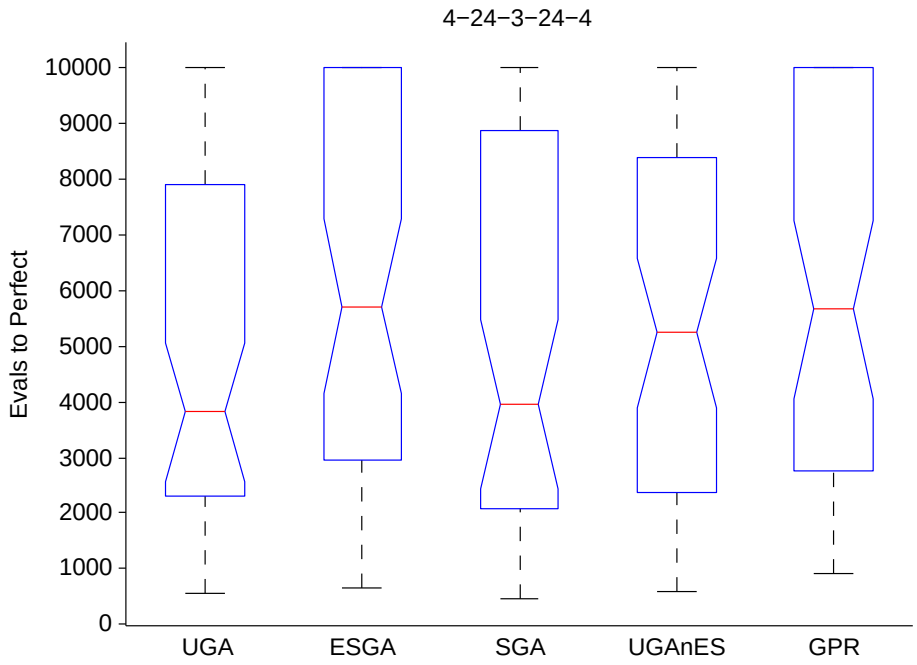

Figure 8.6: A box and whisker plot comparing the Unconstrained GA (UGA), Elite Shuffling GA (ESGA), Standard GA (SGA), UGA no Elite Shuffling (UGAnEs) and Gene Pool Recombination (GPR) on a 4-24-3-24-4 autoencoder.

and a number of UGA variants on NK, two-peak and Royal Road landscapes as well as on an autoencoder task. At the end of the last chapter, the hypothesis was that the gene shuffling improved performance because it maintained diversity in the population which reduced the chance of the population getting stuck on a sub-optimal solution. On the N30 K8 and N30 K9 p90 landscapes, this hypothesis was proven incorrect because reducing the number of individuals whose genes were shuffled did not significantly impact performance.

Upon further investigation, we found that the key to the performance of the UGA on the high-K NK landscapes was not the extent of gene shuffling, but instead whether or not the genes of the elite individual(s) were shuffled throughout the population. One potential explanation of these results is that adding elitism to GAs can be a double-edged sword. On one hand, elitism has the potential to improve performance because it preserves the fittest individuals in the population, but on the other hand it has the potential to reduce performance because it can increase the chance that the population gets stuck on a sub-optimal fitness peak. Our results imply that by shuffling the genes of the elite individuals through the population, the benefits of elitism are retained without increasing the chances of getting stuck on a local optimum. As mentioned earlier Eiben and Schippers (1996) found that increasing the number of parents in the multi-parent recombination GA did

not improve performance on NK landscapes. The details of the GA used in their paper were not given so we do not know whether or not elitism was used, but if it was, then our analysis could possibly explain their results.

The GAs were also tested on two-peak and Royal Road landscapes as well as an autoencoder task to see whether the benefits of shuffling the genes of elite individuals helped evolution on other landscapes. On the two-peak landscapes we found that both the UGA and EGSA, where the genes of the elite individuals were shuffled, significantly outperformed the GAs where the genes of the elite individuals were not shuffled (SGA, UGAnES), but unlike on the NK landscapes, the UGA was much better than the EGSA. This implies that on this type of landscape there is a benefit to shuffling the genes of the entire population on top of shuffling just the genes of the elite individuals.

Unlike on the NK and two-peak landscapes, on the Royal Road landscape, shuffling the genes of the elite individuals significantly impaired performance. This may be because on the Royal Road there are not really any local optima or fitness plateaus since each fitness scoring schema is required to form higher scoring individuals. This means that conserving the elite individuals as a whole can only improve performance on this landscape by providing a stepping stone to higher fitness levels. On this landscape the best performing GA was the UGAnES which shuffled the genes of all individuals in the population except for the elite individuals. This means that there is still some benefits to gene shuffling on this landscape as long as the elite individuals are left alone.

On the 4-24-3-24-4 AE there was no statistical difference in the performance of any of the GAs. This could be because of the high amounts of neutrality in this task because as the NKp results showed, higher amounts of neutrality seems to reduce the difference in performance between GAs.

Based on the results of the previous two chapters, the following conclusions can be made:

- On all the landscapes tested gene shuffling either improved performance or did not significantly affect performance of evolution
- On non-neutral landscapes with a lot of local optima, shuffling the genes of the entire population may improve performance
- The benefits of gene shuffling are most likely related to increasing diversity in the population as well as reducing the chance of elitism which may cause the population to get stuck on a local optima

- Neutrality is a bit of an equaliser which means that on very neutral landscapes the choice of GA is less important
- In the large majority of cases, the UGA significantly outperformed GPR so gene shuffling is preferable to HGT with replacement which can alter gene frequencies

From a broader perspective, our results relate to the schema theorem (Holland, 1975; Goldberg, 1989) view of evolution. In general, the schema theorem claims that if the genetic schema (building blocks) have above average fitness, then the number of these schema will increase exponentially in successive generations. This should mean that genetic operators such as uniform recombination and gene shuffling used in the UGA that potentially break up fully formed schema should reduce the performance of evolution. As the result in this chapter show, this is not the case, which is additional evidence that evolution may not progress by building block mechanisms. These results support the findings in the original Royal Road paper (Mitchell et al., 1992) which questioned the benefits of intermediate building blocks in evolution.

In Part II of this thesis we have shown how new algorithms can be developed by varying the way that the UoS and offspring are used to construct new UoEs. This reinforces how viewing evolution in terms of UoE and UoS is helpful in generating new algorithms.

Chapter 9

Discussion

The most general contribution of this thesis to the fields of artificial evolution and evolutionary computation is showing how the space of potential evolutionary algorithms is extended when evaluation is separated from selection in artificial evolution. As we have discussed throughout this thesis, viewing artificial evolution in this way, where selection and evaluation are distinct steps, is rarely done because most evolutionary algorithms are heavily inspired by natural evolution, where it is difficult to separate evaluation from selection.

In Chapter 3 we describe this new view of evolution in detail and introduce the concepts of Units of Evaluation (UoE) and Units of Selection (UoS). We then present a general evolutionary cycle using UoE and UoS that we believe gives a more complete view of artificial evolution than ones that described artificial evolution solely in terms of selection. This evolutionary cycle was then used to analyse both the breeding of battery hens (Craig and Muir, 1996) and a number of evolutionary robotic experiments by Floreano et al. (2008) and Waibel et al. (2009) to show how our view allows different types of evolution to be differentiated, something that can only be done when evaluation and selection are separated. As discussed, separating evaluation from selection in this way adds an additional degree of freedom to the development of novel GAs because it forces the experimenter to choose both what is being evaluated (UoE) and what is being selected (UoS). A good example of how failing to explicitly distinguish between evaluation and selection can limit the types of evolutionary algorithms, which we highlighted first in section 3.4, is summed up nicely by the following quote from Floreano et al. (2008):

> The level of selection is varied by either measuring team performance and selecting teams (team-level selection) or measuring individual performance and selecting individuals independently of their team affiliation (individual-level

selection)

As we discussed in section 3.4, if the authors had separated evaluation from selection they would not have been limited to these two types of algorithms where the level of evaluation and selection are the same.

In the remainder of the thesis we used this new way of looking at artificial evolution as a blueprint for developing novel GAs. In Experiments Part I, new GAs were developed by varying the relationship between the UoE and UoS. In Chapter 4, the Group GA, which is based on work in Tomko et al. (2011, 2012), is presented. In this GA, the UoE and UoS are both groups of population members which allows evolution to cause the population to niche, where different individuals are doing different jobs to solve the task. In the next chapter (Chapter 5), we study the Binomics GA which was first presented in Harvey and Tomko (2010). In the Binomics GA, the UoE is a group of population members, but unlike the Group GA, the UoS are individuals rather than groups. Comparing these two algorithms in Chapter 5 allowed us to study the effect of changing the UoS from the group to the individual. Both these GAs were tested on an artificial immune system and an autoencoding (AE) artificial neural network (ANN) task. The Group GA was found to outperform the Binomics GA on the immune system task, but on the AE, the opposite was true. To further explore the effects varying the amount of evaluation and selection has on evolution, both GAs were modified so that the amount of selection in the Binomics GA could be increased and the amount of evaluation in the Group GA could be reduced. From testing these modified GAs we found that increasing the amount of selection in the Binomics GA significantly improved performance on both tasks.

In the final chapter of Experiments Part I (Chapter 6), we addressed a potential criticism of the Group and Binomics GAs which is that for either of the GAs to perform well an appropriate group size needs to chosen. In this chapter, we presented a randomly chosen group size and evolved group size version of both GAs that were first tested in Tomko et al. (2012). These methods have the advantage of eliminating the need to run a group size parameter search. Testing showed that even though both methods improve performance as compared to manually varying the group size, the evolved group size method outperforms the random group size method.

In summary, the key findings in Experiments Part I were:

- Setting the UoE to be a group of population members allows evolution to cause niching or a division of labour in the population

- When the UoE and UoS are both groups of population members, like in the Group GA, there is no need to worry about how fitness is assigned to individual group members (the credit assignment problem)
- There is an optimal selection/evaluation balance for different tasks. This highlights the importance of separating evaluation from selection and testing various UoE / UoS combinations
- The evolved group size method combined with the Group or Binomics GA can be used to find the optimal group size, eliminating the need for a parameter sweep

Experiments Part II of the thesis shifts the focus from levels of evaluation and selection to studying how varying the method of constructing new offspring from the UoS impacts evolution. Specifically, we tested the effects of implementing massive amounts of horizontal gene transfer (HGT). In the Unconstrained GA (UGA) (Tomko et al., 2013), which was first described in Chapter 7, HGT is implemented by shuffling the genes of the new offspring population every generation. We found that on rugged (high K), NK landscapes the UGA outperformed a standard generational GA. The other key result from this chapter was that there was no significant difference in performance between the UGA and a standard generational GA when significant amouns of neutrality were added to the rugged NK landscapes.

Chapter 8 explored the reasons why the UGA performed well on rugged NK landscapes by testing a variety of modified versions of the UGA on different landscapes and tasks. One variation of the UGA shuffled only the genes of the elite individuals in the population. This Elite Shuffling GA (ESGA) performed as well as the UGA on the NK landscapes implying that shuffling the elite genes reduces the impact elitism has on causing the population to converge on a sub-optimal solutions. On the other landscapes we compared these GAs on, the UGA was found to outperform the ESGA. Based on these results it probably makes sense to always shuffle the genes of the entire population if HGT is to be implemented. All these GAs were also compared to a version of Gene Pool Recombination (GPR; Muhlenbein and Voigt 1995) which like the UGA, implements HGT, but instead of shuffling the genes without replacement they are shuffled with replacement. On none of the landscapes tested did GPR outperform the UGA and in most cases the UGA performed better. This could be due to the fact that shuffling with replacement can lead to a change in gene frequencies in the population, whereas shuffling without replacement will not impact gene frequencies.

In both parts of this thesis the goal was to present a set of novel GAs developed using the UoE / UoS view of evolution and to determine how they behave on a variety of different

landscapes. This work can now be used as a springboard to start thinking about what type of real-world tasks each of these GAs are most suited to solve. The Group and Binomics GA are tailored to solve problems that require either an implicit or explicit division of labour between component parts, where the optimal division of labour is unknown ahead of time. Engineering tasks which may benefit from being solved with these GAs include scheduling tasks, ANN tasks and co-operative evolutionary robotics tasks.

A logical next step would be to compare the performance of the Group or Binomics GA to other evolutionary and non-evolutionary optimisation methods on one or two engineering tasks. This work is outside the scope of this thesis because in our opinion, when developing new GAs, initially it is more important explore their behaviour on test-bed type tasks and landscapes rather than try to show that they are the 'best' way to solve specific tasks. Doing this allowed us to investigate the GAs interesting qualities and start to really understand why they work better on specific tasks. This thesis has provided the groundwork for people to start applying these GAs to more real-world tasks in a sensible way. A variant of the Group GA has already been successfully applied to the evolution of neurocontrollers on co-operative type tasks as part of a Masters level project (S. James, personal communication, 2013).

Our work has shown that the optimal relationship between the amount of selection (UoS) and evaluation (UoE) changes depending on the task. Based on testing these GAs on the immune system and AE tasks, we believe that it makes sense to set both the UoE and UoS to the group level when the division of labour is explicit, like in the immune systems task, but when the division of labour is more subtle, like in ANN tasks, it probably makes sense to reduce the amount of selection. Applying these GAs to more problems will provide more data points that can be used to determine if there is a good rule of thumb on how to set the UoE and UoS of various classes of tasks.

For tasks where the experimenter is unsure about the optimal division of labour required to solve the task, we recommend first trying to solve the task using the evolved group size version of the Group GA. The Group GA is the simplest GA to set up and has the least number of parameters which allows for easy testing. After evaluating the performance of the Group GA on the task, we recommend applying the modified Binomics GA to the task using the optimal group size found with the evolved group size version of the Group GA. Varying the NUMCOMPARE parameter of the modified Binomics GA will provide information regarding the optimal selection / evaluation balance for the given task which should allow the experimenter to better understand the type of task they are

trying to solve and which GA is best suited to solve it.

The method of gene shuffling used in the UGA family of GAs can be easily added to any standard steady-state GA. Based on the promising results of testing these types of GAs on a variety of different landscapes, we believe that it makes sense to test the UGA and ESGA on any task that is believed to have a very rugged fitness landscape with lots of sub-optimal solutions. Even though we showed that significant amounts of neutrality eliminated the performance advantage the UGA and ESGA had over a standard GA, these gene-shuffling based GAs always performed as well as the standard GA and therefore it still makes sense to apply them to tasks that are known to have neutral fitness landscapes.

Apart from applying gene shuffling to improve evolutionary performance on different tasks, the UGA family of GAs can also be used as the basis for a number of theoretical studies. These include investigating some of the theories on recombination, understanding how our results relate to the schema theory of evolution and further investigating why gene transfer without replacement (gene shuffling) improves performance but gene transfer with replacement in many cases was found to reduce performance.

There is a lot of room to build on the work done in this thesis to further study how to best apply group based GAs, such as the Group and Binomics GAs, to ANN tasks. On the AE task, we showed how using the evolved group size method allows evolution to find the appropriate scaling factor for network weights, eliminating the need to optimise this parameter manually. A potentially interesting area of further research would be to experiment with the add and delete weight mutation operators to see there is a way to improve them so that group evaluation based GAs outperform standard GAs on this type of task. The work presented in Hinton and Srivastava (2012) shows that on deep-learning networks, setting a portion of the weights equal to zero using dropout improves training and may be a good starting point for this type of research.

Separating evaluation from selection also opens up more speculative work such as the possibility of developing GAs where there are multiple UoE and UoS that are involved in a multi-step selection process. For example, going back to the hen breeding example, one could imagine a breeding algorithm where both the number of eggs produced by each individual hen and the total number of eggs produced by each cage is recorded and then the individual hens that lay the most eggs from the fittest cage are selected. In this example, individual hens and cages of hens are both UoE and UoS. Testing these types of GAs would be an interesting extension of our research. It is also possible that our way of looking at artificial evolution could be applied to natural evolution. In Chapter 1 we

discussed how in natural evolution, selection and evaluation interact differently than in artificial evolution. This means that our analysis may be able to shed further light on both the differences and similarities between natural and artificial evolution.

Overall this thesis has contributed the following to the field of evolutionary computation: (1) It has provided a new way of looking at artificial evolution that separates evaluation and selection and shown how this view can be used to develop novel algorithms; (2) It has presented two novel GAs (the Binomics and Group GAs) that can be used to evolve solutions to problems where the optimal division of labour is unknown; (3) It has shown the benefits of adding gene shuffling to standard GAs and shown how shuffling the genes of the elite members of the population can eliminate one of the issues with elitism in GAs. Our hope is that this thesis has provided a set of new GAs that can be applied to a wide range of interesting problems as well as a simple methodology that can be used to develop novel GAs.

Bibliography

Adami, C. and Brown, C. (1994). Evolutionary learning in the 2D artificial life system 'Avida'. In Brooks, R. A. and Maes, P., editors, *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 1–5. MIT Press. 27

Aguirre, H. and Tanaka, K. (2003). A study on the behavior of genetic algorithms on NK-landscapes: Effects of selection, drift, mutation, and recombination. *IEICE Trans. Fundamentals*, (9):2270–2279. 111

Baluja, S. and Davies, S. (1998). Pool-wise crossover in genetic algorithms: An information-theoretic view. In *In Foundations of Genetic Algorithms V*. 19, 20

Barnett, L. (1998). Ruggedness and neutrality: The NKp family of fitness landscapes. In Adami, C., Belew, R., Kitano, K., and Taylor, C., editors, *Artificial Life VI: Proceedings of the Sixth International Conference on Artificial Life*, pages 18–27. The MIT Press. 110, 111

Burnet, F. (1959). *The Clonal Selection Theory of Acquired Immunity*. The University Press, Cambridge, MA. 53

Carver, R. (1978). The case against statistical significance testing. *Harvard Educational Review*. 151

Collins, R. J. and Jefferson, D. R. (1991). Selection in Massively Parallel Genetic Algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 1–15. 8

Colwell, R. (1981). Group selection is implicated in the evolution of female-biased sex ratios. *Nature*, 290:401–404. 22

Committee on Metagenomics (2007). *The New Science of Metagenomics: Revealing the Secrets of Our Microbial Planet*. National Academies Press, Washington, DC. 25

Craig, J. and Muir, W. (1996). Group selection for adaptation to multiple-hen cages: beak related mortality, feathering and body weight respones. *Poultry Science*, 75:295–302. 28, 29, 30, 133

Damuth, J. (1988). Alternative formulations of multilevel selection. *Biology and Philosophy*, 3(4):407–430. 22, 37

Dawkins, R. (1982). *Replicators and vehicles.* Cambridge University Press, Cambridge, MA. 23, 37

Dawkins, R. (2012). The descent of Edward Wilson. *Prospect Magazine.* 20

De Jong, K. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, Ann Arbor. 15, 54

Dick, G. (2005). A comparison of localised and global niching methods. In *17th Annual Colloquium of the Spatial Information Research Centre (SIRC 2005: A Spatio-temporal Workshop)*, pages 91–101. 14, 15, 45, 108

Dienes, Z. (2011). Bayesian versus orthodox statistics: which side are you on? *Perspectives on Psychological Science*, 6(3):274–290. 152, 153, 154

Eiben, A. and Schippers, C. (1996). Multi-parent's niche: n-ary crossovers on NK-landscapes. In *Proceedings of the 4th Conference on Parallel Problem Solving from nature*, pages 319–328. Springer. 19, 122, 130

Eisen, J. (2007). Environmental shotgun sequencing: Its potential and challenges for studying the hidden world of microbes. *PLoS Biology*, 5(3). 25

Fisher, R. (1930). *The Genetical Theory of Natural Selection.* Clarendon Press, Oxford. 22

Floreano, D., Mitri, S., Perez-Uribe, A., and Keller, L. (2008). Evolution of altruistic robots. *Computational Intelligence: Research Frontiers*, pages 232–248. ix, 18, 38, 39, 133

Fogel, L., Owens, A., and Walsh, M. (1966). *Artificial Intelligence through Simulated Evolution.* John Wiley and Sons, New York. 6

Forrest, S., Javornik, B., Smith, R. E., and Perelson, A. S. (1993). Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation*, 1(3):191–211. 45, 46, 54, 88

Forrest, S. and Miller, J. (1990). Emergent behavior in classifier systems. *Physica D: Nonlinear Phenomena*, 42:213–227. 17

Goldberg, D. and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*. 9, 14, 34

Goldberg, D. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimisation. In Grefenstette, J., editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Lawrence Erlbaum Associates. 15, 54

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley. 6, 18, 132

Goodnight, C. and Stevens, L. (1997). Experimental studies of group selection: what do they tell us about group selection in nature? *American Naturalist*, 150:59–79. 18

Grefenstette, J. J. (1988). Credit assignment in rule discovery systems based on genetic algorithms. *Machine Learning*, 3(2-3):225–245. 34

Hamilton, W. (1964). The genetical evolution of social behaviour 1. *Journal of theoretical biology*, 7(1):1–16. 18, 22

Hamilton, W. (1975). Innate social aptitudes of man: an approach from evolutionary genetics. *Biosocial Anthropology*, 133:155. 22

Handelsman, J. (2004). Metagenomics: application of genomics to uncultured microorganisms. *Microbiology and Molecular Biology Reviews*, 68(4):669–685. 25

Harvey, I. (1994). Evolutionary robotics and SAGA : the case for hill crawling and tournament selection. In Langton, C., editor, *Artificial Life III*, volume XVI, pages 299–326. Addison Wesley. 14, 148

Harvey, I. (1996). The Microbial Genetic Algorithm. *http://www.sussex.ac.uk/Users/inmanh/Microbial.pdf*. 12, 13, 47

Harvey, I. (2001). Artificial Evolution : A Continuing SAGA. In Gomi, T., editor, *Evolutionary Robotics: From Intelligent Robots to Artificial Life, Proc of 8th International Symposium on Evolutionary Robotics*, pages 1–19. Springer-Verlag. 12, 13, 47

Harvey, I. (2011). The Microbial Genetic Algorithm. In Kampis, G., Karsai, E., and Szathmary, E., editors, *ECAL 2009: Advances in Artificial Life. Darwin Meets von*

Neumann - 10th European Conference, pages 126–133, Heidelberg. Springer. 8, 12, 13, 47

Harvey, I. and Tomko, N. (2010). Binomics : where metagenomics meets the binary world. In Fellermann, H., Dorr, M., Hanczyc, M., Laursen, L., Maurere, S., Merkle, D., Monnard, P., Stoy, K., and Rasmussen, S., editors, *Proceedings of Artificial Life XII, 12th Intl. Conf. on the Synthesis and Simulation of Living Systems*, pages 370–377. MIT Press. xii, xiv, 4, 13, 36, 39, 43, 55, 57, 59, 68, 69, 71, 134

Hinton, G. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504. 58

Hinton, G. and Srivastava, N. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv Pre Print*, pages 1–18. 62, 80, 103, 137

Holland, J. (1976). Adaptation. In Rosen, R. and Snell, S., editors, *Progress in Theoretical Biology, 4. Plenum.* 17, 34, 83

Holland, J. and Reitman, J. (1978). Cognitive Systems Based in Adaptive Algorithms. In Waterman, D. and Hayes-Roth, F., editors, *Pattern-directed Inference Systems.* Academic Press. 17, 34, 83

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*, volume Ann Arbor. University of Michigan Press. 1, 6, 132

Horn, J., Goldberg, D., and Deb, K. (1994). Implicit niching in a learning classifier system: Nature's way. *Evolutionary Computation*, 2(1):37–66. 15

Husbands, P. (1993). An ecosystems model for integrated production planning. *International Journal of Computer Integrated Manufacturing*, 6(1):74–86. 16

Husbands, P. and Mill, F. (1991). Simulated co-evolution as the mechanism for emergent planning and scheduling. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 264–270. Morgan Kaufmann Publishers. 16, 54

Jeffreys, H. (1961). *The theory of probability.* Oxford University Press, Oxfrord. 67, 154

Johnson, D. (1999). The insignificance of statistical significance testing. *The journal of wildlife management*, 63(3):763–772. 151, 152

Jong, K. D. (1987). On using genetic algorithms to search program spaces. In Grefenstette, J. J., editor, *Proceedings of the 2nd International Conference on Genetic Algorithms*. Erlbaum Associates. 34

Kauffman, S. (1993). *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press. 86, 101, 110

Kauffman, S. and Johnsen, S. (1991). Coevolution to the edge of chaos: coupled fitness landscapes, poised states, and coevolutionary avalanches. *Journal of theoretical biology*, 149(4):467–505. 86, 101, 110, 111

Koza, J. R. (1990). Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Technical report, Dept. of Computer Science Stanford University. 6

Mahfoud, S. W. (1995). *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign. 14, 15, 45, 108

Mayley, G. (1996). Landscapes, learning costs, and genetic assimilation. *Evolutionary Computation*, pages 1–21. 111

Maynard Smith, J. (1964). Group selection and kin selection. *Nature*, 201:1145–1147. 21, 22

Maynard Smith, J. (1976). Group selection. *The Quarterly Review of Biology*, 51(2):277–283. 21, 22

McIlhagga, M., Husbands, P., and Ives, R. (1996). A comparison of optimization techniques for integrated manufacturing planning and scheduling. *Parallel Problem Solving from Nature IV*, pages 604–613. 16

Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. MIT Press, London. 6

Mitchell, M., Forrest, S., and Holland, J. (1992). The royal road for genetic algorithms: fitness landscapes and GA performance. In Varela, F. J. and Bourgine, P., editors, *The First European Conference on Artificial Life*, number 1. MIT Press/Bradford Books. 121, 132

Moriarty, D. and Miikkulainen, R. (1995). Learning Sequential Decision Tasks. In Honavar, V., Patel, M., and Balakrishnan, K., editors, *Advances in the Evolutionary Synthesis of Neural Systems*. MIT Press. 16, 36, 55, 80, 83, 103

Moriarty, D. E. and Miikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, 22:11–32. 16, 36, 55, 80, 83, 103

Muhlenbein, H. and Voigt, H.-M. (1995). Gene pool recombination in genetic algorithms. *Meta-heuristics: Theory and applications*. 19, 115, 118, 135

Nawa, N. and Furuhashi, T. (1998). Bacterial evolutionary algorithm for fuzzy system design. In *IEEE Conference on Systems, Man, and Cybernetics*, pages 2424–2429. IEEE. 20

Nawa, N., Hashiyama, T., Furuhashi, T., and Uchikawa, Y. (1997). A study on fuzzy rules discovery using Pseudo-Bacterial Genetic Algorithm with adaptive operator. *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97)*, pages 589–593. 20

Nowak, M. A., Tarnita, C. E., and Wilson, E. O. (2010). The evolution of eusociality. *Nature*, 466(7310):1057–1062. 22

Ochman, H., Lawrence, J. G., and Groisman, E. a. (2000). Lateral gene transfer and the nature of bacterial innovation. *Nature*, 405(6784):299–304. 25, 110

Okasha, S. (2005). Multilevel selection and the major transitions in evolution. *Philosophy of science*, 72(5):1013–1025. 37

Okasha, S. (2010). Altruism researchers must cooperate. *Nature*, 467(7316):653–5. 20

Petrowski, A. (1996). A clearing procedure as a niching method for genetic algorithms. *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 798–803. 15

Potter, M. and De Jong, K. (1994). A cooperative coevolutionary approach to function optimization. In Davidor, Y. and Schwefel, H., editors, *Parallel Problem Solving from Nature (PPSN III)*, pages 249–257. Springer Verlag. 16, 54

Potter, M. and De Jong, K. (2000). Cooperative coevolution: an architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29. 45, 46, 54

Price, G. (1970). Selection and covariance. *Nature*. 34

Qin, J., Li, R., Raes, J., Arumugam, M., Burgdorf, K. S., Manichanh, C., Nielsen, T., Pons, N., Levenez, F., Yamada, T., Mende, D. R., Li, J., Xu, J., Li, S., Li, D., Cao,

J., Wang, B., Liang, H., Zheng, H., Xie, Y., Tap, J., Lepage, P., Bertalan, M., Batto, J.-M., Hansen, T., Le Paslier, D., Linneberg, A., Nielsen, H. B. r., Pelletier, E., Renault, P., Sicheritz-Ponten, T., Turner, K., Zhu, H., Yu, C., Li, S., Jian, M., Zhou, Y., Li, Y., Zhang, X., Li, S., Qin, N., Yang, H., Wang, J., Brunak, S. r., Doré, J., Guarner, F., Kristiansen, K., Pedersen, O., Parkhill, J., Weissenbach, J., Bork, P., Ehrlich, S. D., and Wang, J. (2010). A human gut microbial gene catalogue established by metagenomic sequencing. *Nature*, 464:59–65. 26, 56

Quinn, M. (2006). *The Evolutionary Design of Controllers for Minimally-Equipped Homogeneous Multi-Robot Systems*. PhD thesis, University of Sussex. 18, 40

Quinn, M., Smith, L., Mayley, G., and Husbands, P. (2003). Evolving controllers for a homogeneous system of physical robots: structured cooperation with minimal sensors. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 361(1811):2321–43. 18

Rechenberg, I. (1973). *Evolutionsstratategie: Optimierung technischer Syseme nach Prinzipien der biologicschen Evolution*. Frommann-Holzboog, Stuttgart. 6

Rouder, J. N., Speckman, P. L., Sun, D., Morey, R. D., and Iverson, G. (2009). Bayesian t tests for accepting and rejecting the null hypothesis. *Psychonomic bulletin & review*, 16(2):225–37. 154

Schwefel, H.-P. (1995). *Evolution and Optimum Seeking*. John Wiley and Sons, New York. 6

Spector, L. and Klein, J. (2005). Trivial geography in genetic programming. *Genetic programming theory and practice III*. 15

Swenson, W., Wilson, D. S., and Elias, R. (2000). Artificial ecosystem selection. In *Proceedings of the National Academy of Sciences*, volume 97, pages 9110–4. 19

Syswerda, G. (1993). Simulated Crossover in Genetic Algorithms. In *Foundations of Genetic Algorithms*, pages 239–255. Morgan Kaufmann, San Mateo, 2 edition. 19

Thomas, C. M. and Nielsen, K. M. (2005). Mechanisms of, and barriers to, horizontal gene transfer between bacteria. *Nature reviews microbiology*, 3(9):711–21. 25

Tomassini, M. (2005). *Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time*. Springer. 15

Tomko, N., Harvey, I., and Philippides, A. (2013). Unconstrain the Population : The Benefits of Horizontal Gene Transfer in Genetic Algorithms. In Harvey, I., Cavoukian, A., Tomko, G., Borrett, D., Kwan, H., and Hatzinakos, D., editors, *SmartData Privacy Meets Evolutionary Robotics*, pages 117–127. Springer. 5, 107, 135

Tomko, N., Harvey, I., Philippides, A., and Virgo, N. (2011). Many hands make light work : group evolution and the emergent division of labour. In Lenaerts, T., Giacobini, M., Bersini, H., Bourgine, P., Dorigo, M., and Doursat, R., editors, *Advances in Artificial Life, European Conference on Artificial Life 2011*, pages 318–325. MIT Press. xiii, 4, 36, 39, 43, 44, 53, 134

Tomko, N., Harvey, I., Virgo, N., and Philippides, A. (2012). Many Hands Make Light Work : Further Studies in Group Evolution. *Artificial Life*, 19(2):1–37. xiii, 5, 39, 43, 44, 49, 134

Wade, M. (1977). An experimental study of group selection. *Evolution*, 31(1):134–153. 19

Waibel, M., Floreano, D., and Keller, L. (2011). A quantitative test of Hamilton's rule for the evolution of altruism. *PLoS biology*, 9(5). 18

Waibel, M., Keller, L., and Floreano, D. (2009). Genetic team composition and level of selection in the evolution of cooperation. *IEEE Transactions on Evolutionary Computing*, 13(3):648–660. ix, 18, 38, 39, 133

Watson, R., Hornby, G., and Pollack, J. (1998). Modeling building-block interdependency. In Eiben, A., Back, T., Schoenauer, M., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature PPSN V*, pages 97–106. Springer. 18

Watson, R. and Pollack, J. (1999). Incremental commitment in genetic algorithms. In Banzhaf, W., Daida, J. M., Eiben, A., Garzon, M. H., Honavar, V., Jakiela, M. J., and Smith, R. E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999)*, pages 710–717. Morgan Kaufmann. 17, 18

Watson, R. and Pollack, J. (2000a). Combination and recombination in Genetic Algorithms. 18

Watson, R. and Pollack, J. (2000b). Symbiotic combination as an alternative to sexual recombination in genetic algorithms. In Shoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature PPSN VI*, pages 425–434. Springer. 17, 18, 28, 55, 84

West, S., El Mouden, C., and Gardner, A. (2010). Sixteen common misconceptions about the evolution of cooperation in humans. *Evolution and Human Behavior*, 321(3). 20, 21, 22

Whitley, L. (1989). The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. In *Proceedings of the Third International Conference on Genetic Algorithms*. 8

Williams, H. and Lenton, T. (2007). Artificial ecosystem selection for evolutionary optimisation. In Almeida E Costa, F. E. A., editor, *Advances in Artificial Life: Proceedings of the 9th European Conference on Artificial Life*, pages 93–102, Berlin. Springer Verlag. 18, 55

Wilson, D. (1977). Structured demes and the evolution of group-advantageous traits. *The American Naturalist*, 111(977):157–185. 22

Wilson, D. (2012a). Clash of Paradigms. *Huffington Post*. 20

Wilson, D. S. (1975). A theory of group selection. *Proceedings of the National Academy of Sciences*, 72(1):143–6. 22

Wilson, E. O. (2012b). *The Social Conquest of Earth*. Liveright Publishing Corporation. 20

Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82. 4

Wu, S. and Banzhaf, W. (2009). Investigations of Wilsons and Traulsens group selection models in evolutionary computation. In Kampis, G., Karsai, I., and Szathmary, E., editors, *Proceedings of the 10th European Conference on Artificial Life*, pages 1–8. Springer. 35

Wu, S. and Banzhaf, W. (2011). Evolutionary Transition through a New Multilevel Selection Model. In Lenaerts, T., Giacobini, M., Bersini, H., Bourgine, P., Dorigo, M., and Doursat, R., editors, *ECAL 11, Eur. Conf. on Artificial Life*, pages 874–881, Paris. MIT Press. 22

Wynne-Edwards, V. (1963). Intergroup selection in the evolution of social systems. *Nature*, 200:623–626. 21

Appendix A

Selection Pressure of GAs

Since many people are not familiar with the Microbial GA it is worth spending some time explaining that the selective pressure of the Microbial GA is exactly the same as any other steady-state GA which is equivalent to the generational GA described in the Literature Review chapter. For the purposes of this thesis, selective pressure will be defined as a measure of how exploitative an algorithm is, in terms of the evolutionary exploitation versus exploration balance. Exploitation in genetic algorithms can be thought of as the 'centripetal' or selective force that causes the population to converge to a single genotype, and exploration can be thought of as the 'centrifugal' force that causes the population to spread out. This selective pressure is not related to 'how effective' a GA is in relation to a basket of optimisation problems, which is another way selective pressure is sometimes defined. This section is a summary of a report that can be found at http://goo.gl/1yHa1 and is based on work in Harvey (1994).

The measure of selective pressure we use can be calculated for any GA by taking a large population (of size P) of identical genotypes, all with zero fitness. Then a single beneficial mutation is given to one member of the population so that this individual has non-zero fitness. The GA of interest is then run for a single generation, or a generation equivalent if a steady state GA is being tested, and the expected number of copies of this beneficial gene in the population is counted. This is repeated for a large number of generations, each time starting with a single beneficial gene in the population, and the average over these runs is taken to be the selective pressure.

If we calculate the selective pressure of the generational GA with tournament selection described earlier we find that it has a selective pressure of 2. This is because to generate a single offspring 4 parents are picked from the population so that two parental tournaments can be run. This means that the individual with the beneficial mutation can be expected

to be picked 4/P times every time an offspring is generated. Since this individual has a higher fitness than everyone else in the population it will win every tournament it enters and therefore become a parent. Each offspring gets 50% of its genes from each parent so the expected number of copies of the beneficial gene in the offspring is (4/P)/2 = 2/P. If this generation of a new offspring is run P times which is equal to a single generation then the total number of expected copies of the beneficial mutation in the population is (2/P)*P which is 2. It is important to note that in this type of generational GA, the type of recombination does not impact the selective pressure of the GA. Whether uniform recombination, one-point or two-point crossover, or even using 10% of one parent's genes and 90% of the other's to construct the offspring, the selective pressure will still be 2.

In **all** steady-state GAs, including the Microbial GA, the selective pressure is (for a large population) e, or around 2.7. The reason the selective pressure of steady-state GAs is slightly higher than that of the generational GAs is because their current generation and next generation are not kept separate. By definition, a steady-state GA updates the population in 'real time' which means that after the first tournament, there will be, on average, more than one copy of the mutant already in the population. This increases the chances of the beneficial mutation replicating in the succeeding tournaments so the selective pressure is 2.7, which is slightly higher than that of the generational GA, but in practice will likely make little difference.

Unlike recombination in the generational GA, increasing or decreasing the infection rate in the Microbial will alter the selective pressure. For example, if the infection rate is increased to 100% where the offspring is just a copy of the fitter parent then the selection pressure becomes $e^2 \approx 7.2$. This is equivalent to applying selection to both the choice of parents and the choice of who dies.

It is important to realise that in the Microbial GA elitism is implicitly built in to the algorithm. This is because the fitter of the two randomly chosen parents always survives meaning that the best individual in the population will survive until a fitter individual is evolved.

In both the Microbial GA and the generational GA, the selection pressure is the same but implemented in different ways. In the generational GA, a selective pressure is applied to the choice of both parents through tournament selection but the choice of who is going to die is unbiased because the entire preceding generation is killed off without any favouritism. In the Microbial GA, by randomly choosing the parents from the population and then replacing the weaker parent with the offspring there is a selective pressure applied

to both the choice of who is going to survive and who is going to die. These two methods result in the exact same amount of selective pressure.

Appendix B

Statistical Testing Methods

An important part of this thesis is comparing the performance of different evolutionary algorithms. These comparisons can be made using either orthodox or Bayesian statistical analysis. The large majority of scientific papers use orthodox statistics, ignoring many of the issues with these methods. As we will discuss in this section, Bayesian statistics avoid many of these issues and in our view are the more logical and sound way to make statistical comparisons. For these reasons we will present both orthodox and Bayesian statistical analysis in this thesis.

B.1 Orthodox Statistical Comparisons Using p-values

One of the most common orthodox methods of statistical comparisons is using p-values. Johnson (1999) describes the four main steps required to analyse data using p-values as follows:

1. Develop some null hypothesis which is usually the opposite of what you want to show. So if you want to show that one algorithm is better than another the null hypothesis would be that the two algorithms perform equally well.

2. Collect data, run simulations or tests.

3. Run a statistical test of the null hypothesis to generate a p-value.

4. Interpret the p-value

P-values have been interpreted in many different ways, most of which are incorrect. Carver (1978) believes the three most common statistical p-value fantasies are as follows, and in his paper explains why they are not a valid way of interpreting p-values.

1. *Odds-Against-Chance Fantasy* which is interpreting the p-value as the probability that the experimental results were due to chance. In other words, small p-values show that there was a very small chance that the results obtained were due to random chance.

2. *Replicability or Reliability Fantasy* is interpreting 1-p as the probability of getting the same results if the same experiment was run again.

3. *Valid Research Hypothesis Fantasy* is believing that 1-p is the probability that the research hypothesis is true.

According to Johnson (1999) the correct interpretation of a p-value is that it is the probability of getting the observed data, or more extreme data given: (1) the null hypothesis is true, (2) the assumed model is correct and (3) sampling is done randomly. Because the p-value is always calculated assuming the null hypothesis is true then you can only reject the null hypothesis, a low p-value does nothing to confirm your experimental hypothesis.

Even if p-values are interpreted correctly there is still the question of how useful they are in understanding experimental results. One well known issue with p-values is that because they are a function of the difference between reality and the null hypothesis and sample size, p-values can be reduced by increasing the sample size. Dienes (2011) expands on this further, explaining how if the null hypothesis is false then increasing the sample size will drive the p-value to zero, but if the null hypothesis is false then the p-values are normally distributed meaning that there is equal probability of getting any p-value. Most scientific journals accept a result as 'significant' if the p-value is less than 0.05, ignoring the fact that this is a very arbitrary choice; two other problems with this are: (1) Increasing the sample size can get you a 'significant' results and (2) even if the null hypothesis is true there is a 5% chance of getting a p-value that is interpreted as 'significant'. This issues also raise questions such as how many tests should I run and whether my results are valid if I've made a planned versus post-hoc hypothesis. The next section will summarise how using Bayesian methods avoids many of these issues.

B.2 The Benefits of Bayes

In this section we summarise some of the arguments put forward by Dienes (2011) to why using Bayesian statistics is better than orthodox statistics when it comes to analysing experimental data.

Since p-values are sensitive to the number of samples or experimental runs, the question of when to stop an experiment or simulation is brought into question. For example, assume that when two algorithms are compared over 25 runs a p-value of 0.07 is found but when they are compared over 100 runs a p-value of 0.01 is found. In the first case, the results would not be considered 'significant' but in the second case they would be 'significant' even though the only thing that has changed is the number of times the algorithms were run. This means that using orthodox statistics a results can go from not significant to significant just by increasing the number of experimental runs or samples. Bayesian statistics like the Bayes factor avoids this problem because when the null is true or closer to the truth than the alternative the Bayes factor is driven to zero Dienes (2011). As discussed earlier, in orthodox statistics, when the null hypothesis is true the p-value isn't driven to any value but is uniformly distributed between zero and one. This means that the Bayes factors can be quantitatively compared while p-values cannot.

Another issue that Bayesian statistics avoids is the whole problem of multiple testing. This is especially relevant in this thesis because in many chapters we compare multiple algorithms on a specific task. In orthodox statistics you have to correct for the number of comparisons done in a given family. One problem with this is that it can be difficult to determine what counts as being part of a family. For example if I test five different algorithms on a given task, how do I determine which are part of the same family and should be statistically compared together? With Bayesian statistics this is not an issue because it doesn't matter how many hypothesis are tested as long as all the relevant data are taken into consideration (Dienes, 2011).

The final advantage Bayes has over orthodox statistics that we will mention is how it handles planned versus post hoc comparisons. In orthodox statistics it matters whether a hypothesis is formulated before or after the data is observed. With Bayesian statistics the strength of evidence is the same irrespective of whether it is analysed before or after a hypothesis is formed.

For the reasons mentioned in this section we believe that Bayesian statistics are a better way to compare the performance of different algorithms. But because p-values are the still the industry standard in academia we will present both types of statistics.

B.3 Tests Used in this Thesis

Here we describe both the orthodox and Bayesian statistical tests we use in this thesis.

B.3.1 Orthodox Statistical Tests Used

One factor in determining which statistical test to use is whether the data is normally distributed or not. In most cases the results from testing our genetic algorithms were not normally distributed so we chose to report results in terms of medians and interquartile ranges (IQR) rather than using means and standard deviations which should be used for data that is known to be normally distributed. For comparing two algorithms we used the Wilcoxon rank sum test of equal medians and for comparing multiple algorithms we used the Kruskal-Wallis test for equal medians which is suitable for testing three or more groups from non-Gaussian distributions.

B.3.2 Bayesian Statistical Tests Used

We chose to use the on line two-sample t-test Bayes factor calculator ($pcl.missouri.edu/bayesfactor$) described in Rouder et al. (2009). The Bayes factor is one of the more common Bayesian statistical tests and is described on Dienes' web page as follows:
($www.lifesci.sussex.ac.uk/home/Zoltan\_Dienes/inference/Bayes.htm, April 2013$)

> The Bayes factor tells you how strongly data support one theory (e.g. your pet scientific theory under test) over another (e.g. the null hypothesis). It is a simple intuitive way of performing the Bayesian equivalence of significance testing, telling you the sort of answer which many people mistakenly think they obtain from significance testing, but cannot.

Unlike p-values, which do not provide evidence for or against the null hypothesis, Bayes factors can be directly interpreted as odds ratios and therefore can be interpreted as evidence in favour of or against your hypothesis over the null hypothesis (Dienes, 2011; Rouder et al., 2009). Because Bayes factors are odds ratios that compare one theory against another, they can be reported without reference to cut-offs (Rouder et al., 2009). For readers not familiar to Bayes factors a good rule of thumb is to follow the convention suggested by Jeffreys (1961) which is to take Bayes factors of greater than 3 (or less than 1/3) to indicate substantial evidence for or against the null hypothesis.

The Rouder et al. (2009) Bayes factors we use in this thesis are structured in terms of the null hypothesis over the alternative hypothesis which means that large Bayes factors imply that there is strong evidence in favour of the null hypothesis, while small Bayes factors signify that the evidence is in favour of the alternative hypothesis.