Many Hands Make Light Work: Further Studies in Group Evolution

Nicholas Tomko, Inman Harvey, Nathaniel Virgo and Andrew Philippides

CCNR, Evolutionary and Adaptive Systems Group, University of Sussex, Brighton UK

nt79@sussex.ac.uk, inmanh@gmail.com, nathanielvirgo@gmail.com, andrewop@sussex.ac.uk

July 25, 2012

Abstract

When niching or speciation is required to solve a task that has several different component parts, standard GAs struggle. They tend to evaluate and select all individuals on the same part of the task which leads to genetic convergence within the population. The goal of evolutionary niching methods is to enforce diversity in the population so that this genetic convergence is avoided. One drawback with some of these niching methods is that they require *a priori* knowledge or assumptions about the specific fitness landscape in order to work; another issue is that many such methods are not set-up to work on cooperative tasks where fitness is only relevant at the group level. Here we address these issues by presenting the Group GA, first described in [28], which is a group based evolutionary algorithm that can lead to emergent niching. After demonstrating the Group GA on an immune system matching task we extend the work of [28] and present two modified versions where the number of niches do not need to be specified ahead of time. In the Random Group Size GA, the number of niches is varied randomly during evolution and in the Evolved Group Size GA the number of niches is optimised by evolution. This provides a framework in which we can evolve groups of individuals to collectively

perform tasks with minimal *a priori* knowledge of how many subtasks there are or how they should be shared out.

1 Introduction

In biology, speciation and niching are complex concepts but we suggest that they can be broadly described as the evolutionary process by which a single type of biological organism differentiates into multiple "specialised" types, that for instance, take advantage of different resources available in a given environment. In some cases niching produces competing species, but niching can also occur within a single species to produce different specialists that work together to solve a given task. An example of this would be a bacterial colony, where within any single colony there are groups of different bacteria doing different jobs, all of which are contributing to the collective well being of the colony. In this case the fitness of the colony depends on the collective symbiotic functionality rather than the fitness of any individual bacterium. This view of bacteria working together for the good of the colony is supported by work done in the relatively new field of Metagenomics [9, 2, 6, 16] and specifically shot-gun sequencing techniques which allow groups of bacterial genomes to be sequenced [30].

Most standard artificial evolutionary and genetic algorithms (GAs) [12] tend to take a very individual centric view of evolution, where the fittest individuals are selected to produce the next generation of individuals. These types of GAs work well on problems with a single optimum, where each individual can solve the task on its own; but they are unable to find multiple solutions to multi-peaked problems or solve problems cooperatively, where there is a division of labour between population members which requires different genotypes. For a GA to be able to find cooperative solutions to problems, 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 [5] and [18] 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 *a priori* knowledge about the specific fitness landscape to work; in particular whether n is 2 or 5 or some different number. We believe that most of these explicit niching methods can be classified as either direct or indirect methods to determine the appropriate number of niches. We define direct niching methods as those where the number of species (or niches) is set before evolution begins, an example being cooperative coevolution [25]; and indirect methods as those that rely on a pre-set niching (similarity) radius or some sort of similarity calculation in order to get the population to niche, such as fitness sharing [8] and crowding [4]. The other problem with these niching methods 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. One of the more striking examples of the benefits of group evaluation is the evolution of battery hens [3]. In this paper it was demonstrated that when battery hens were evolved based on the number of eggs produced per cage, rather than the number of eggs produced per hen, total 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 is done at the individual hen level, 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 (groups) 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. In this example, evaluating hens at the group level ensured that peaceful, egg producing hens were evolved – something that was not possible using individual evaluation.

Here we present a novel genetic algorithm, the Group GA, which niches based on the evaluation and selection of groups of individuals and therefore can be used to solve tasks that require individuals working together doing different jobs. The Group GA has the added benefit of accomplishing this niching with minimal *a priori* knowledge of the

fitness landscape and it is able to niche without knowing how the different jobs should be shared out. So unlike the more common niching methods it does not require the exact number of niches to be set ahead of time nor does it require setting any indirect niching parameter such as a similarity or niching radius.

Even though there is an ongoing debate regarding whether group selection actually occurs to any significant extent in nature (see [23, 31, 22, 17] for a summary), it is important to note that many of the issues surrounding group selection in natural evolution are not relevant 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 on the same or different time scale as individual turnover, as desired, 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. While it may be possible to relate the Group GA to some of these issues in biology, doing so is not part of the scope of this paper. We only mention the biological connection to highlight the fact that in artificial evolution many of the issues in biology can be ignored because the experimenter is free to choose the level of selection and evaluation.

We demonstrate the emergent niching ability of the Group GA on an artificial immune system matching task. 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.

This paper is an extended version of the Many Hands Make Light Work: Group Evolution and the Emergent Division of Labour paper [28]. The original paper introduced the Group GA and demonstrated its ability to evolve a niched population of antibodies on both a four antigen task and on a task where the number of antibodies changed during evolution. One question that was raised after presenting this paper at the 2011 European Conference on Artificial Life (ECAL 2011) was how sensitive the Group GA was to changes in the group size parameter. Here we address this in a new section entitled 'Varying the Group Size'. We present an in depth analysis of the impact of group size on the Group GA and then introduce two variations of the Group GA that allow it to be implemented without having to preset a specific group size. In the Random Group Size GA (RGGA), the group size is randomly varied during evolution and in the Evolved Group Size GA (EGGA), the group size is evolved as part of evolution. We show that both the RGGA and EGGA can solve the four antigen task and explain how these algorithms allow the Group GA to be easily applied to tasks where the optimal group size is unknown at the beginning of evolution.

In the next section we will briefly review some of the common niching methods as well as a few related evolutionary algorithms that can solve problems symbiotically, where a division of labour is required. Following our literature review we describe the artificial immune system task and the Group GA in detail. We will then show how the Group GA can be used to evolve a population of antibodies to match a set of four antigens, as well as how it can be used to evolve a population of antibodies that adapts to the addition and removal of antigens during evolution. The new section on varying group size follows this. Finally, we compare the Group GA to other evolutionary methods and discuss the types of tasks we feel the Group GA is best suited to solve. We believe that this provides a framework from which we can evolve groups of individuals to collectively perform a group of tasks or subtasks, with minimum prior knowledge of how many subtasks there are or how they should be shared out.

2 Literature Review

We start by reviewing the most common niching methods in artificial evolution. These are appropriate when a set of different tasks or subtasks needs to be performed collectively, and the purpose of these niching methods is to stop the population from genetically converging on a single task as happens when using a conventional GA. All of these niching methods below can be classified as *explicit* because they either require the number of niches to be set *a priori* or require an indirect method of enforcing diversity in the population.

We will also briefly discuss SANE [20, 21] and the Binomics GA [11] which are two GAs that are set-up to allow *implicit* niching to evolve symbiotic solutions to problems. Unlike explicit methods, these algorithms attempt to evolve a diverse, niched population emergently using group evaluation. They also differ from the genetically based niching methods in that they do not require that each individual in the population can solve the task on its own.

2.1 Genetically Based Niching Methods

Here we briefly describe the common genetically based niching methods. These are based on the assumption that each individual in the population has its own fitness. For a more in depth summary see [5] and [18].

2.1.1 Fitness Sharing and Clearing

Fitness sharing [8] 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 [24] 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 [13] it is shown that in Learning Classifier System models where fitness is shared amongst cooperating individuals implicit niching can occur.

2.1.2 Crowding

Crowding was first introduced in [4] as a method of removing similar individuals from a population, with the goal of trying to maintain diversity during evolution. Deterministic Crowding [18] 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 Demes and Spatially Structured GAs

Alternatives to genetically based niching methods include spatially structured GAs; for a good review see [5] and [27]. 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.

2.2.1 Cooperative Coevolution

Cooperative coevolution was first introduced in [15, 14] for the application of job shop planning and scheduling, and has been further studied in [25] and [19]. 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. Here speciation is not emergent because the number of subpopulations needs to be determined before evolution begins.

2.3 Symbiotic GAs

SANE [20, 21], the Binomics GA [11] and simulated ecosystem evolution [32] are three examples of GAs that cause implicit niching in the population and attempt to evolve symbiotic solutions to problems. In SANE and the Binomics GA, groups of individuals are evaluated together and then the individuals that are part of the fittest groups are selected to pass on their genes to the next generation. This differs from most standard GAs where individuals are evaluated and then the fittest individuals are selected. These algorithms are relevant to our discussion of speciation/niching because any time a problem is solved symbiotically then implicit niching must be occurring.

SANE and the Binomics GA have been successfully applied to the evolution of artificial neural networks (ANNs). When doing this the individuals in the population are partial networks that are combined to form fully specified ANNs which are then evaluated. The fitness score of each individual partial network is based on the fitness of the full ANNs that each individual partial network participated in. This means that over time, the individual partial networks that were part of the fittest ANNs will be selected for, while the partial networks that were part of the least fit ANNs will be modified using mutation and recombined with other partial networks. The goal is to evolve a population of partial networks that symbiotically work together to form high fitness fully specified ANNs.

3 The Artificial Immune System Task

We have chosen an artificial immune system matching task to demonstrate the emergent niching abilities of the Group GA that is presented below. The task is introduced in this section before we describe the Group GA in detail so that we can use the immune system task to clearly illustrate the Group GA. This task, which has previously been used in [7] and [26] 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 [7] this task was used to study adaptation in the immune system and in [26] different variations of this task were solved using cooperative coevolution. We will compare the results of these two papers to the Group GA results later in the paper.

The goal of this task is to evolve a population of antibodies to protect the body from a set of antigens. Very 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 body-guards who mark these antigens for removal. This is a group task since defense against only a subset of the antigens is insufficient; the full set of attacking antigens needs to be covered. Antibodies in natural immune systems need to be adaptive so that they can combat new and different antigens that 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 variable set of antigens.

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 in such a way so that it contains specialists to combat each different antigen. Obviously the more antigens there are, the more difficult the task becomes, because the antibody population needs to evolve and maintain a larger number of specialists.

4 The Group GA

The Group GA is a novel evolutionary algorithm presented in the original version of this paper [28] for the first time. It is based on the Microbial GA [10] which is a steady-state GA that uses tournament based selection.

We will first describe the Group GA in general terms and then describe it in terms of the artificial immune system task we present in this paper. What differentiates it from more conventional GAs is that *groups* of population members (of some fixed 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. A single cycle (tournament) of the Group GA can be broken up into the five following steps:

1. Randomly choose two groups that may have common members from the population without regards to fitness.

2. Calculate and assign a fitness score to each group of population members based on the group's performance on a given task. Fitness is assigned on group performance only; there does not need to be a way to define or calculate an individual's fitness in isolation.

3. All members of the less fit group are removed from the population and replaced with mutated copies of the members of the fitter group.

4. The members of the fitter group are put back in the population unchanged.

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

When we apply the Group GA 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.

A single cycle (tournament) of the Group GA can be broken-up into the five following steps when applied to the immune system task described in the previous section (see figure 1).

1. Randomly choose two groups of antibodies from the population without regard for fitness

2. Calculate the match scores between all the antigens in the set and each of the antibodies in each group

3. Each group as a whole is assigned a fitness score which is calculated as described above.

4. The group with the lower fitness score is replaced with mutated copies of the antibodies of the more fit group

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

We have set up this simulation in such a way that groups of antibodies are randomly chosen from the population and then assigned a fitness based on the ability of this group

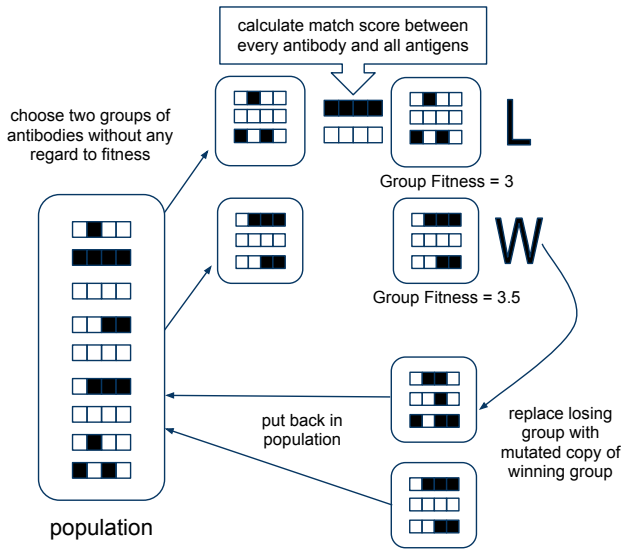

Figure 1: The Group GA as applied to a immune system task with two, 4-bit antigens. To evaluate a group of antibodies, all the antibodies in the group are matched against every antigen in the set and the average of the highest match scores against each antigen is the group fitness. For example the top 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.

to match the different antigens in the antigen set. We understand that because an individual antibody can always be assigned its own fitness, some of the genetically based niching methods we reviewed earlier would be able to solve this task without any type of group evaluation. We have used this task to demonstrate the Group GA because, as we will see below, it clearly shows how the Group GA causes emergent niching.

Although in 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.

5 Evolving Antibodies using the Group GA

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 these experiments the antigen and antibodies were 64-bit binary strings. The antibody population size was 100 and the number of antibodies per group was 10 (which implies 100 tournaments is broadly equivalent to a generation in a corresponding generational GA). The mutation rate was set to 0.1 per genotype, meaning that at each locus there was a probability of 1/640 of flipping that bit.

Figure 2 shows the antibody population after being evolved for 20,000 tournaments

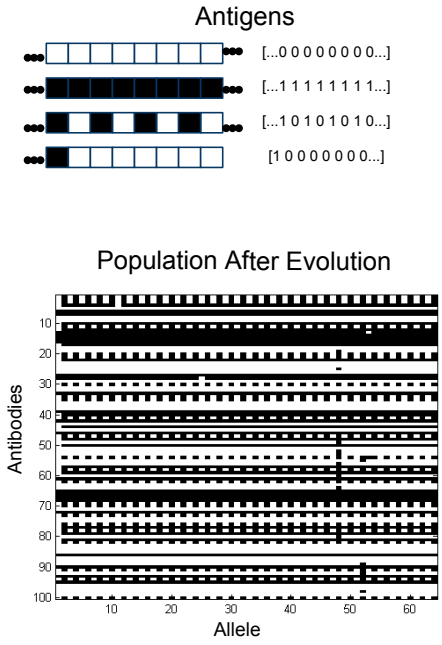

Figure 2: The antibody population after evolution on a 4 antigen task.

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'. These specific antigens were chosen to try to make the task as difficult as possible. The lower part of figure 2 (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. Figure 2 shows how the antibody population has niched during evolution at the end of a typical run. 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 3 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

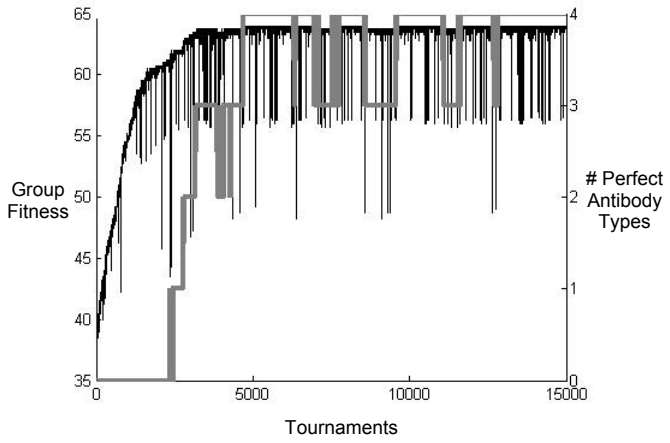

Figure 3: A plot of group fitness (black line) and 'population fitness' (gray line) over time for a single typical run of the 4 antigen task. The 'population fitness' is the number of antigens covered perfectly by at least one antibody from the population as a whole.

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 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 5 shows the fitness versus time plot for this a single typical run of this task, where antigens are added and removed during evolution. As this figure shows, when an antigen is added, the fitness of the population drops before quickly recovering as the population adapts to match this new invader [1].

5.1 Comparison to Other Methods

To get a feel for how well the Group GA is able to solve on this task we compared it to both the Microbial GA [10] and the Binomics GA [11] on the 4 antigen task described above. Using the Microbial GA to solve this task is equivalent to solving it using any standard GA where the fittest individual antibodies are evaluated and selected. As expected, when we ran the Microbial GA for 100 runs, each run the antibody population converged to match a single antigen in the antigen set, failing to match the other three.

A more interesting comparison is between the Group GA and the Binomics GA. We chose the Binomics GA as opposed to a genetic based niching method such as fitness sharing or crowding because like the Group GA, the Binomics GA was developed to solve cooperative tasks using emergent niching where group fitness is the driver for selection. The Binomics GA can be applied to this 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 mutation and recombination.

3. This modified antibody is combined with a group of randomly chosen antibodies from the population.

[1] There are potential similarities between the adaptive mechanism of the Group GA and clonal selection [1] that need to be investigated further.

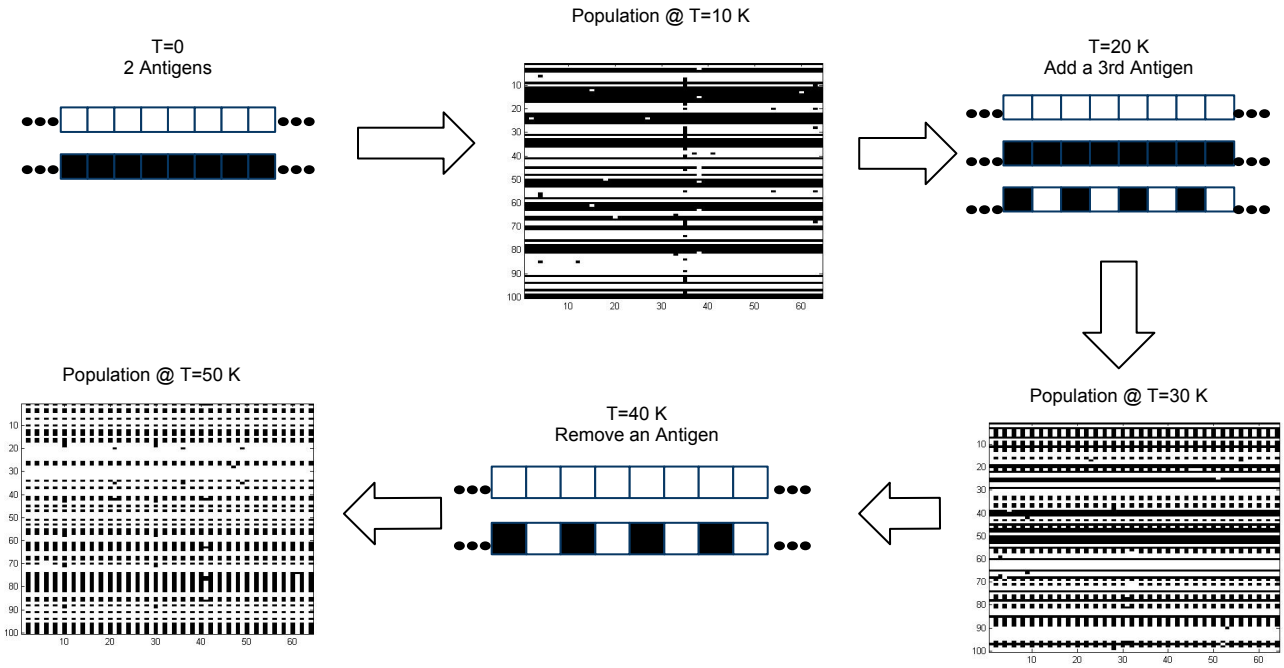

Figure 4: This figure shows how the antibody population adapts during evolution when 64 bit antigens are added and removed (T=10 K corresponds to tournament 10,000).

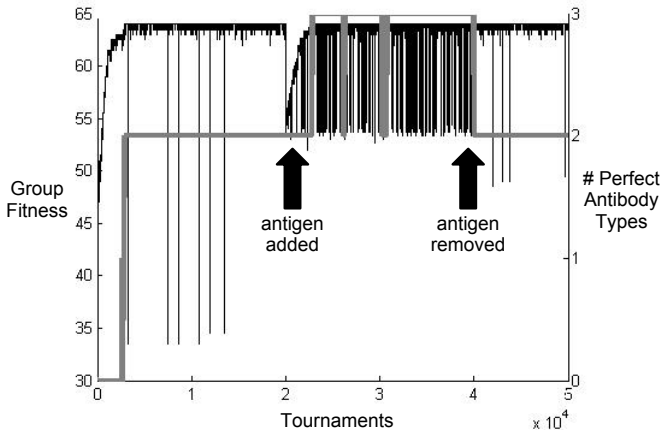

Figure 5: A plot of group fitness (black line) and number of antigens covered perfectly by at least one antibody (gray line) in the population over time for a single typical run of the task where antigens are added and removed during evolution.

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.

6. All antibodies in the group have their individual fitness updated towards this shared group-fitness value, using a time smoothing algorithm that takes into account both their historical individual-fitness and this newly calculated group-fitness.

7. All individuals are put back in the population and this cycle is repeated.

The difference between the Group GA and the Binomics GA is that in the Group GA, groups of antibodies are being both evaluated and selected, while in the Binomics GA groups of antibodies are being evaluated, but it is individual antibodies that are being selected based on this group fitness. This illustrates a distinction between Units of Evaluation and Units of Selection that is discussed further below.

We compared the performances of the Group GA and Binomics GA on the 4 antigen task over fifty runs. We set the parameters of both the Group GA and Binomics GA equal to the parameters used in the previous experiments described in Section 4. We based the comparison on the number of evaluations it took to evolve a population that contained antibodies that perfectly matched all antigens in the set. Over 50 runs the Group GA took a median number of 275 K evaluations with an interquartile range (IQR) of 67 K evaluations, while the Binomics GA took a median number of 1200 K evaluations with an IQR of 269 K evaluations. Using the Wilcoxon rank sum test of equal medians at the 5% significance level we found that the Group GA significantly outperformed the Binomics GA on this task.

The same immune system task was solved by [26] 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, [26] 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 [26] state, the level of stagnation at which species should be added or destroyed is task dependent.

This task was also solved by [7] 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. In contrast with the Group GA, however, their best

match method requires the fitness of each individual population members to be evaluated on its own. This is possible for this task by matching each individual antibody against a single antigen, but tasks where fitness can only be evaluated at the collective, group level will not be able to be solved using this best-match method.

In general, the genetically based niching methods described earlier will struggle with this type of symbiotic task where individual fitness is meaningless. An example would be 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 network. Both SANE and the Binomics GA discussed earlier have been used to solve ANN tasks in this way. It is important to point out that the group size parameter is different from the number of species parameter in cooperative coevolution. As we showed, the Group GA can solve the 4 antigen task with the group size parameter set to a variety of different values, but for cooperative coevolution to be able to solve the same task the number of species needs to be set to equal the number of antigens.

6 Varying Group Size

One potential criticism of the Group GA is that for it to work, an appropriate group size needs to be chosen for the given task. Here we go beyond [28] to investigate sensitivity to group size and to demonstrate how appropriate values for group size could be evolved. In this section we will show that the Group GA can solve the 4 antigen task with a range of different group sizes and explain why some group sizes work better than others. We will then present two modified versions of the Group GA where the group size is varied during evolution. In the first version the group size is randomly changed before each tournament (the Random Group Size GA), while in the second version the group size is evolved as part of evolution (the Evolved Group Size GA). These two variants allow the Group GA to be easily be applied to a wide variety of tasks because a specific group

size does not need to be set at the beginning of evolution. We believe that for many real world tasks the optimal group size will be unknown and therefore it will make sense to use either the Random or Evolved Group Size GA. For example, if the goal is to evolve a group of robots to solve some task, it is quite possible that the experimenter does not know ahead of time how many robots should be grouped together to best solve the task.

6.1 The Effect of Group Size on the 4 Antigen Task

To understand how group size impacts the performance of the Group GA, we evolved 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). The results show that even though there is an optimal group size for this task, the Group GA can perform satisfactorily at a wide range of group sizes around such an optimum and this range becomes larger at bigger population sizes.

Figure 6 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) is 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 ratio of group size to population size also impacts the performance of the Group GA. Figure 6 show 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 but when the population size is increased to 1000 the algorithm can solve the 4-antigen task up to a group size 100. This ratio is important because as the group size is increased, the probability of different antibodies being part of the tournament groups is reduced, which limits the genetic diversity of the groups. This lack of diversity negatively impacts the performance of evolution to the point that above a certain ratio of group size to population size the Group GA cannot cause niching in the population.

In [7] 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.

6.2 The Random Group Size GA

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 to problems where the optimal group size is unknown *a priori*. For example, if the number of antigens is unknown and too large or too small a fixed group size is chosen then the Group GA will struggle to evolve a niched population of antibodies and 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 7 shows the performance of the Random Group Size GA (RGGA) where the group size is randomly chosen before each evolutionary tournament with population sizes of either 100 or 1000. It is compared to the EGGA which will be described in the next section. In both cases, the group size range was set between 2 and 25% of the population size; 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

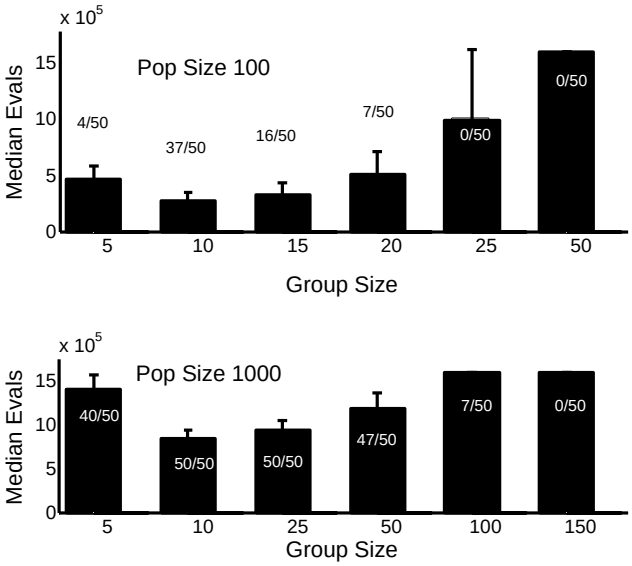

Figure 6: The performance of the Group GA on the 4 antigen task with different group sizes, where performance was measured as: (1) The median (height of bar) and Interquartile Range (shown as an error bar) number of evaluations over 50 runs that it took the Group GA to evolve the first population with 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.

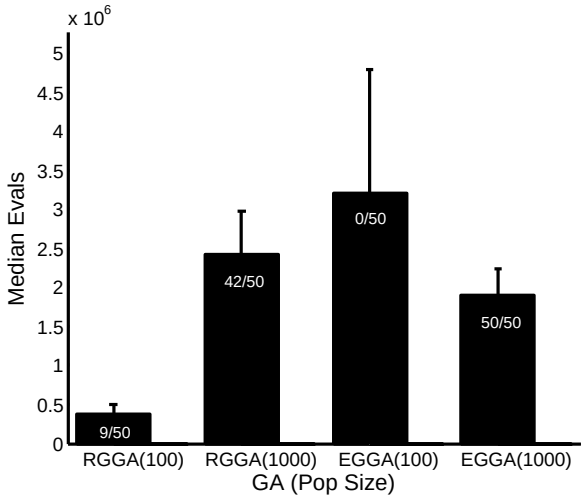

Figure 7: The performance of the Random Group Size GA (RGGA) and Evolutionary Group Size GA (EGGA) at population sizes of 100 and 1000. Performance is illustrated in the same way as in Figure 6.

250. This figure shows that the RGGA is able to solve the 4 antigen task. As expected, the algorithm performs better when the population size is set to 1000 since (as discussed above), with a bigger population the Group GA is effective over a larger range of group sizes.

Choosing the group size range will be task dependent. For some tasks there may be practical reasons for limiting the population and/or the group size. But for other tasks, like the immune system task, there may be no *a priori* constraints on how big a group or population is used. For the reasons described above if the group size is unknown then the Group GA should initially be run with a large population (greater than 500) and a random group size where the upper limit of the group size is set no higher than 25% of the population size. This should increase the chances of evolving high fitness solutions with minimal amount of parameter exploration.

6.3 The Evolved Group Size GA

In the previous section we showed how the Group GA could be run with a group size that was randomly varied during evolution. The RGGA is useful because it avoids the need to do a group size parameter sweep to find an appropriate group size to solve the task. Two shortcomings of the RGGA 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. In this section we present the Evolved Group Size GA (EGGA) which is a version of the Group GA where the group size is evolved. In the EGGA, the group size preference of each antibody is genetically encoded and changes during evolution based on the performance of the groups each antibody takes part in. This is accomplished by adding a single, randomly generated, real numbered gene between 0 and 1 to each antibody and then using these genes to determine the group sizes as follows:

1. Start with an empty group
2. Choose a random antibody from the population
3. Calculate the total group size weight by summing up the individual group size weight 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 for these new genes promotes larger group sizes, and *vice versa*. After the two groups of antibodies (potentially of different sizes) have been constructed, the fitness of each group is calculated in the same way as before and a mutated copy of the winning group overwrites the losing group. If the number of antibodies in the winning group is greater than the number of antibodies in the losing group then a randomly chosen subset of winning antibodies overwrite all the antibodies losing group; if fewer then all of the antibodies from the winning group overwrite a

subset of the losing group before mutation. This ensures that the population size stays constant throughout evolution. After the winning group has overwritten the losing group that entire group is mutated. Mutation is carried out in the same way as it was in the standard Group GA except that the real numbered group size weight gene is mutated by adding a small normally distributed random number to it.

We tested the EGGA on the 4 antigen task with population sizes of 100 and 1000. Performance was measured over 50 runs using the same metrics that were used to evaluate the RGGA. As figure 7 shows, the Evolved Group GA performs 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 the EGGA 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 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 when we applied the EGGA to this task, 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 8 shows how the average group size weight gene changed during evolution for

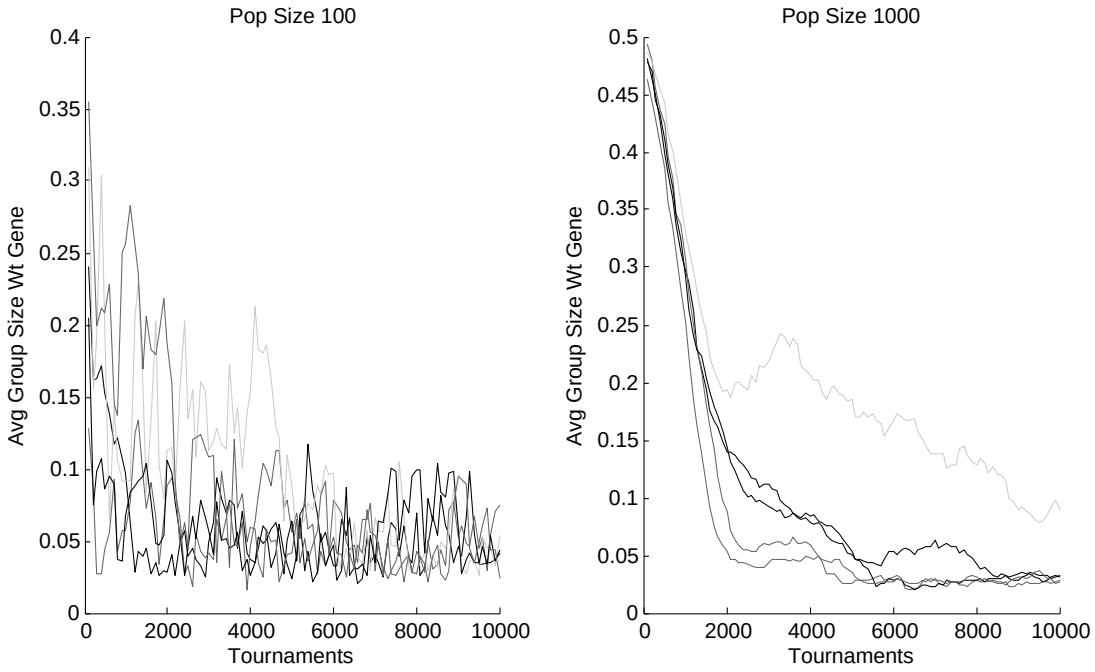

Figure 8: A plot of the average group size weight gene over time for 5 different runs using a population size of 100 and 1000.

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 a 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 EGGA 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. Like the case where the group size weight genes were randomly intialised between

0 and 1, the EGGA was able to solve the 4 antigen task and evolution 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 the EGGA on a flat fitness landscapes 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 RGGA described earlier. Many interesting questions remain, such as: why does the EGGA cause the group size weight gene to converge around 0.03, if we change the number of antigens will the evolved group size weight genes change, and what is the mechanism behind the evolution of group size in the EGGA? These questions show that studying group selection and evaluation in GAs seems like a viable, interesting research avenue and one that could open up new artificial evolution research areas.

7 Discussion

The aim of this work is to open up the study of artificial evolution of groups of entities that can cooperate on a single task or a group of tasks. We have noted that there is scope for unjustified assumptions and for misunderstandings when talking about group selection in biological evolution, where it is necessary to appreciate where the balance may lie between selection at the individual and the group level. However in artificial

evolution we are able to dictate just where and how such balances arise.

We have presented a novel evolutionary algorithm that can cooperatively solve problems using emergent niching, where fitness is evaluated at the group level; further, that this algorithm remains effective even whilst the number of subtasks in the problem is changing. We demonstrated this by using the Group GA to solve a multi-peaked artificial immune system matching task where the number of antigens changes during evolution. Our results show that by evolving a population of antibodies using the Group GA, the population niches to match multiple antigens and when antigens are added and removed during evolution, the Group GA allows the antibody population to adapt to this change matching new antigens that are presented. We then presented and tested two modified versions of the Group GA, the Random Group Size GA (RGGA) and the Evolved Group Size GA (EGGA) which do not require a specific group size to be chosen ahead of time. In the RGGA the group size is varied randomly during evolution and in the EGGA the number of groups is evolved.

We demonstrated the benefits of this novel algorithm by showing how the Group GA outperforms both the Microbial GA and the Binomics GA on the 4 antigen task. The difference between these algorithms is easily seen by viewing them in terms of what is being evaluated and what is being selected. In artificial evolution, as opposed to natural evolution, evaluation and selection are two distinct steps and by viewing artificial evolution in this way the experimenter can vary what is being evaluated (the Units of Evaluation) and what is being selected (the Units of Selection). In the Microbial GA, individual antibodies are simultaneously both the Unit of Evaluation and Unit of Selection, so it is unsurprising that it fails to solve the multi-antigen task and ends up converging to match a single antigen every run.

In the Group GA, groups of antibodies are both the Units of Evaluation and Units of Selection (i.e. they are identical) because fitness is calculated at the group level and the tournament winning group overwrites the losing group. In the Binomics GA, the

Unit of Evaluation is the group of antibodies, but the Unit of Selection is now different, namely the individual antibody, because fitness is passed from the group to the individual to determine selection at the individual level. The Binomics GA outperformed the Microbial GA and was able to niche to match different antigens, but took a lot longer as compared to the Group GA. We believe that the reason why the Group GA outperforms the Binomics GA on this task is related to the difference between what is being evaluated and what is being selected.

The Group GA differs from the genetically based niching methods described earlier because fitness is evaluated at a group level (like in symbiotic GAs) which means that the Group GA can be used to solve tasks where fitness is meaningless at the individual level. It also has the advantage of niching emergently without having to know the exact number of niches ahead of time or pre-setting any parameter such as a niche radius. One limitation of the Group GA is that even though the exact number of niches does not need to be known ahead of time, a specific group size does need to be set. The RGGA and EGGA eliminate this need by varying the group size during evolution. In the RGGA this is done by randomly varying the group size every tournament and in the EGGA the group size is evolved. We have shown that both these GAs are able to solve the 4 antigen task and in the case of the EGGA an appropriate group size is chosen.

We have shown that the EGGA is able to evolve an appropriate group size that allows evolution to solve the 4 antigen task. As far as we know this is a novel method of genetically encoding and evolving the number of groups or niches. Significantly, we can see that the EGGA is able evolve a viable group size even though as we discussed earlier, from a pure fitness maximisation standpoint bigger groups are always better. This means that over time it modifies the group size into the sweet spot between groups that are too small and hence have low fitnesses and those that are too large to allow niching to occur. This initial study indicates a need to investigate whether this is a property of this specific task or if the EGGA can also be successfully used to evolve

group and niche sizes for other tasks.

Another advantage of the Group GA is that it could be used to evolve groups where the individuals in the groups have different genetic codes. For example, the Group GA would still be able to evolve a niched population of antibodies if some of the antigens had a real numbered genetic code instead of a binary code. There is no recombination (at the individual level) in the Group GA, the losing group is just replaced by a mutated copy of the winning group and this may allow further freedom in the types of genetic encoding. The only modification to the Group GA required to allow it to evolve groups containing individuals with different genetic codes is implementing different types of mutation for each different genetic encoding.

By changing one's view on what is the group and what is the individual, the Group GA can be used to solve many different problems. As discussed earlier, both SANE and the Binomics GA take a different view of what is the group and what is the individual when evolving ANNs. In standard GAs the full network is viewed as the individual, but both SANE and the Binomics GA view the individual neurons as the individuals and the fully constructed network as the group. In this same way, one could view a robot as a group of body parts that all work together in a symbiotic manner. The Group GA could be used to evolve an entire robot's morphology where each part could be specified using a different genetic code. Whereas conventionally in Evolutionary Robotics one would prespecify just how many limbs and brains there are [29], the use of the Group GA could relax the need for such constraints. To evolve a full robot, two random groups of robot parts would be chosen from the population and evaluated as a full robot. Initially, the robots constructed from the groups will likely have very low fitness e.g. they may contain 4 arms, no legs and no neural controllers - but over time the Group GA should modify the population so that it contains the appropriate parts to construct fit robots.

The Group GA suite of algorithms have the potential to be a useful set of tools that use emergent niching to solve problems where the optimal division of labour is

unknown. The RGGA and EGGA do not require a specific group size to be set ahead of time and the EGGA has the potential to evolve the optimal group size with no *a priori* assumptions. Going forward, there is a need to test these GAs on a wide variety of tasks, including evolving neural networks and planning and scheduling tasks, which may benefit from being solved cooperatively in order to find out when it performs well and under what circumstances it performs poorly.

Bibliography

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

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

3. 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.

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

5. 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, Dunedin, New Zealand.

6. Eisen, J. (2007). Environmental Shotgun Sequencing: Its Potential and Challenges for Studying the Hidden World of Microbes. *PLoS Biol.*, 5(3):e82.

7. 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.

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

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

10. Harvey, I. (2011). The Microbial Genetic Algorithm. In Kampis, G., Karsai, E., and Szathmary, E., editors, *ECAL 2009, Part II. LNCS 5778*, pages 126–133, Heidelberg. Springer.

11. 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, Odense, Denmark. MIT Press.

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

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

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

15. 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. Koonin, E. V. and Wolf, Y. I. (2008). Genomics of bacteria and archaea: the

emerging dynamic view of the prokaryotic world. *Nucleic acids research*, 36(21):6688–719.

17. Lehmann, L., Keller, L., West, S., and Roze, D. (2007). Group selection and kin selection: two concepts but one process. *Proceedings of the National Academy of Sciences of the United States of America*, 104(16):6736–9.

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

19. 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.

20. 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*, Cambridge, MA. MIT Press.

21. Moriarty, D. E. and Miikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, 22:11–32.

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

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

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

25. 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.

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

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

28. 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, *ECAL 11, Eur. Conf. on Artificial Life*, pages 318–325, Paris. MIT Press.

29. Vaughan, E., Di Paolo, E., and Harvey, I. (2004). The evolution of control and adaptation in a 3D powered passive dynamic walker. In Pollack, J., Bedau, M., Husbands, P., Ikegama, T., and Watson, R., editors, *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems, ALIFE'9*, pages 139–145, Cambridge, MA. MIT Press.

30. Venter, J. C., Remington, K., Heidelberg, J. F., Halpern, A. L., Rusch, D., Eisen, J. A., Wu, D., Paulsen, I., Nelson, K. E., Nelson, W., Fouts, D. E., Levy, S., Knap, A. H., Lomas, M. W., Nealson, K., White, O., Peterson, J., Hoffman, J., Parsons, R., Baden-Tillson, H., Pfannkoch, C., Rogers, Y.-H., and Smith, H. O. (2004). Environmental genome shotgun sequencing of the Sargasso Sea. *Science*, 304(5667):66–74.

31. 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):i.

32. 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.