Microbial GA report

**Introduction**

The field of genetic algorithms (GAs) was borne out of research in the 50s and 60s on evolutionary computing, which took inspiration from natural evolutionary processes to apply them to practical computation tasks, in particular evolving solutions to computational problems, such as airfoil design (Mitchell, 1998, p2). Genetic algorithms took a slightly different approach to the evolutionary computation techniques in existence at the time; instead of aiming to develop algorithms to solve specific problems, the creator of GAs, John Holland, sought to study the mechanics of natural adaptation and utilise these in computer systems. His genetic algorithms employ a kind of ‘natural selection’ and genetics-inspired operators to alter a population of ‘chromosomes’, i.e a string of genes (with values of 1 or 0), into a new population, utilising the operations of crossover and mutation. The incorporation of more than just mutation in evolutionary computation was ground-breaking, and set the topic of genetic algorithms going; today the varying techniques have amalgamated somewhat, but the foundational idea of using selection, crossover and mutation (to varying degrees) persists in the field. Genetic algorithms are a useful tool now in a vast array of fields, from dynamic control systems (Michalewicz, 1992), to scheduling problems (Wall, 1996), to UAV path planning more recently (Roberge, 2012), borne out of the algorithms’ simplicity, and relative lack of required knowledge of the internals of the problem.

This report aims to study a minimalist variant of the genetic algorithm, the microbial GA (Harvey, 1996), and analyse how varying the internal parameters of the algorithm can affect its performance on a simple problem. The microbial GA is an attempt by Harvey to create as bare-bones a genetic algorithm as possible, whilst still retaining the key operations of selection, crossover and mutation, and using a population of chromosomes. To form such a minimal GA, Harvey employs the techniques of a steady state GA (Whitley, 1989); in traditional GAs, a whole new population is formed out of the previous generation, and then this becomes the new current generation, however in steady state GAs, one offspring is produced at a time, and replaces another member of the population. This negates the need to form a whole new population each generation, and is combined with tournament selection (Harvey, 1992), to choose which ‘parent’ is fitter at each comparison. After *N* tournament selections, this is the equivalent to one generation passing in a traditional GA. Many traditional GAs use roulette wheel selection, wherein the probability of selection is based on an individual’s fitness divided by the total fitness of the population, akin to a roulette wheel with different size sections for each individual; tournament selection is far simpler and just compares fitness of two individuals, the greater fitness ‘winning’.

The broad aim of this paper is thus to use the microbial genetic algorithm to analyse a simple optimisation problem; namely, the problem of dividing friends between two parties to minimise conflict between those who dislike each other. Imagine you have a group of 20 close friends, and want to have a birthday celebration. However, some of your friends like each other, and others dislike each other, so you decide to throw two parties to optimise interactions and prevent arguments.

The microbial GA is thus tasked with evolving a genotype specifying which of the two parties each friend will attend, so as to maximise some fitness function based upon the number of positive interactions. The genotype *G* is specified by each individual taking an allele of +1 or -1, indicating they are attending party 1 or 2, and the fitness is evaluated using an ‘interaction matrix’ between friends, which effectively says who likes/dislikes who. So, if two people like each other, they have a weight of +1, and if they dislike each other, a weight of -1 (liking/disliking is assumed to be symmetrical for simplicity). This can form a diagonal matrix of weights *W* between every person, as the example below shows. For a group of 4 friends, an example genotype could be -1 1 -1 1, and its interaction matrix could be:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *W* | P1 | P2 | P3 | P4 |
| P1 | 0 | 1 | -1 | -1 |
| P2 | 1 | 0 | 1 | 1 |
| P3 | -1 | 1 | 0 | -1 |
| P4 | -1 | 1 | -1 | 0 |

Table 1 - Interaction matrix W

The fitness function minimising the number of conflicting individuals is then

with higher *F* indicating a better party allocation. So, if two people who like each other go to the same party, their contribution will be positive, 1 x 1 x 1 = 1, or 1 x -1 x -1 = 1, and vice versa.

The key objective to be studied in this report is the effect of the crossover operator on performance of the microbial GA, namely, is crossover beneficial for the performance of a genetic algorithm, both on speed of convergence and max fitness? I hypothesise that crossover will increase max fitness performance, but the crossover GA will take more generations to reach its maximum. The report will analyse the effects of crossover on the microbial GA, for a variety of different parameter values on the above problem. The rest of the report will be structured as follows: the next section will outline the methods and tools used for the analysis; the third section will report the findings, and the final section will offer a discussion and analysis of the results.

**Method**

The chief requirement to enable this study is an implementation of the microbial GA. I utilised the MATLAB software package to write this, which offers a suite of useful tools for analysing numerical functions. We received a template of a basic roulette wheel-based GA, which I adapted to include the features of the microbial GA, namely tournament selection and use of a single population, and tailored it to the party task, by implementing the fitness function *F*, creating a population of random genotypes, and building the interaction matrix *W*. There are a number of parameters of the GA: NoIndividuals specifies the number of genotypes in the population; NoGenes is the length of each genotype (20 for the main problem); NoGenerations is the number of iterations of the GA that will be done; mutationRate is the probability that any gene along a genotype will be mutated during the mutation step; Pc is the crossover probability, the probability that any gene along a parent genotype will be passed to the offspring genotype.

Once the basic implementation was built, I tested its functionality on a genotype length of 4, i.e splitting 4 people into 2 parties, and verified it was working correctly by running it a number of times. With a working GA, I then proceeded to perform the experiments. Firstly, I ran the GA several times on a genotype length of 20, to observe the expected performance, and then also experimented with the parameters of the algorithm.

With a grasp of the effects of varying the parameters, I built functionality to average the fitness achieved over many runs of the algorithm, and to find the average number of generations taken to reach a maximum fitness score, which would be used to analyse the performance. I built upon this functionality by specifying a range of different population sizes for further analysis. Finally, I generated suitable plots from these varied parameters, to show the effect of crossover on convergence speed, and overall fitness achieved. The results are presented below.

**Results**

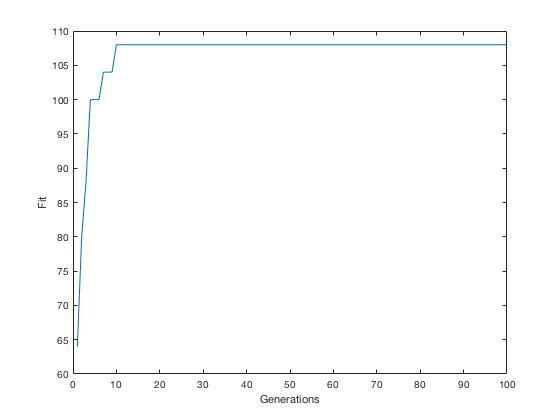
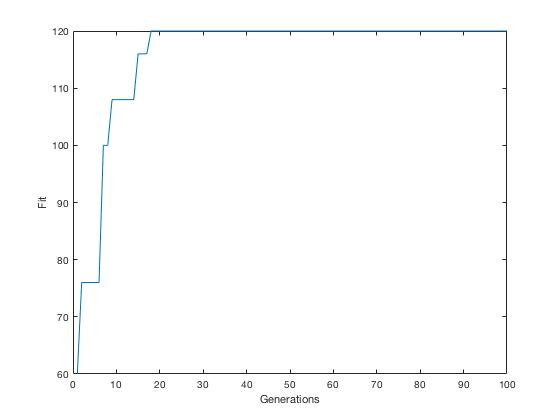
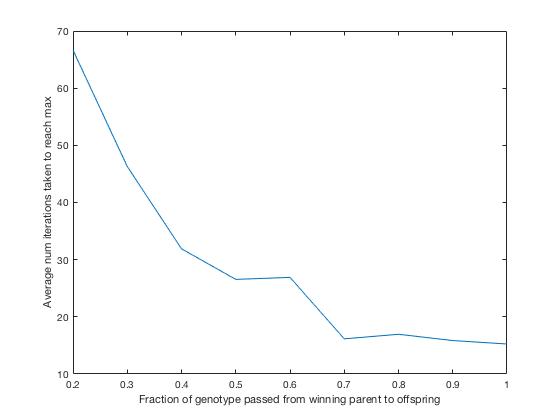


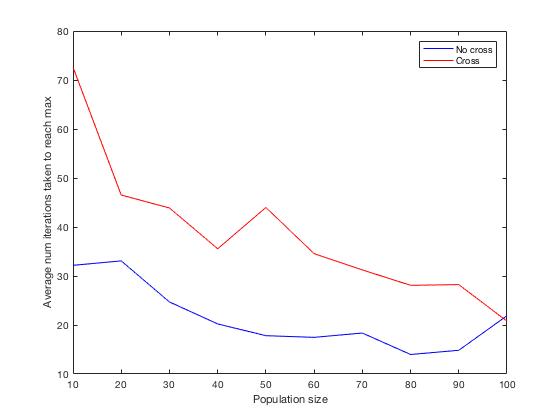
Figure a) GA with crossover 1b) GA without crossover

Typical fitness curves are given in figure 1 for the GA with and without crossover (i.e 50% of winning parent’s genotype passed down, versus 100% respectively) run for 100 generations, with a population size of 50. These figures appear to agree with my initial hypothesis; the crossover GA (figure 1a) achieves a higher fitness of 120, taking 18 generations to reach it, while for no crossover, a maximum of 108 is achieved, but in only 10 generations. To investigate this, table 2 presents the average number of iterations over 50 runs taken by each version of the GA, and their average maximum fitness. Clearly, the GA without crossover arrives at maximums more quickly, but its hastiness is at a slight cost to the maximum fitness it achieves versus the crossover GA. The difference in fitness performance is less than expected however, at least for this simple problem. As the above suggests, the number of iterations required for the GA to reach a maximum decreases as the fraction of genetic material passed from winning parent increases; figure 2 shows this. The decrease in iterations required appears to be an almost exponential decay, with the curve levelling out somewhat as the fraction of genotype passed down increases towards 100% (i.e no crossover).

Next, the effect of changing the population size was investigated. Taking a range of population sizes from 100 down to 10, figure 3 shows how this affected speed of convergence to a maximum, with and without crossover again (N.B each population size’s results were averaged over 20 runs). We can see that for the

Figure 2 - average iterations taken to reach a maximum for a range of crossover probabilities



range of population sizes, the GA with crossover takes longer to converge upon a maximum, steadily decreasing as the population size increase, while the non-crossover GA in fact converges in less than 40 generations for all population sizes. Table 3 gives the average number of iterations taken, and average maximum fitness achieved by each; interestingly, the average fitness scores are roughly the same as when just a population size of 50 was used.

|  |  |  |
| --- | --- | --- |
|  | Average iterations taken | Average fitness achieved |
| With crossover | 26.84 | 118.88 |
| Without crossover | 18.28 | 116.40 |

Table 2 - performance of the GA with and without crossover

Figure - convergence speed of GA for range of population sizes

|  |  |  |
| --- | --- | --- |
|  | Average iterations taken | Average fitness achieved |
| With crossover | 38.55 | 131.24 |
| Without crossover | 21.46 | 116.22 |

Table - performance of the GA for a range of population sizes

**Discussion**

The experiments compared the speed of convergence to a maximum of the GA with and without crossover for a variety of parameters, and on the whole, it was shown that the non-crossover GA converged fastest, and achieved a lower maximum fitness, with a few exceptions. This result makes sense, and agrees with my initial hypothesis; a discussion of why this is the case follows.

Without crossover involved, all the genetic material of the winning parent in a given tournament is passed to the offspring. Thus, the population’s overall genetic material tends slightly more towards the fitness peak that the winner was situated on, than it would if some of the loser’s genetic material remained. Including some of the loser’s genetic material would not necessarily make the offspring immediately fitter than it would have been if the winner was just copied in, but there may have been some good genetic material somewhere along the loser’s genotype, that could later mix with other genetic material in the right way to produce a fitter offspring than the original winner would have been.

Using just the winner’s genetic material, the GA will accelerate up towards nearby maxima, be they the global or local; whereas, using both the winner and the loser’s genotypes, there is the chance that the loser’s genetic material can later mix in a beneficial way with other individuals, or in some cases, produce a fitter offspring immediately from crossover. Conversely, as figure 2 shows, passing too small an amount of genetic material from the winning parent to the child will reduce the GA’s convergence speed drastically, where a value below 50% starts acting like a simple mutation of a few of the genes, rather than intentional passing of genetic material between relatives.

Thus, it would be expected that the crossover GA would outperform the non-crossover GA on maximum fitness achieved, and table 2 partially supports this, however the difference in scores is rather small. This implies that perhaps, for this task, the fitness landscape has many maxima, with many similar fitness values, and so even though the non-crossover GA is converging more quickly to maxima, the maxima it finds are not much different to the ones found with crossover. The results of the experiment on a range of population sizes further support this. The maximum fitness scores shown in table 3 are quite similar, so the effects of crossover on this task are fairly minimal, which implies that the fitness landscape has many maxima, many of them with similar fitness values. A smaller population size means smaller initial diversity, which means that the population is starting its search from fewer points in the landscape. That the scores were similar at the start with and without crossover versus at the end (eg for crossover GA, 132.64 at population size = 10, and 132.32 at population size = 70), shows that there are multiple maxima in this landscape, each with quite large basins of attraction, such that even a smaller population size can converge towards many of them.

That being said, it appears there is a trade-off to be made on this problem. The GA with crossover achieves an average score of 10 points more than the non-crossover GA, but it takes longer to converge for the full range of population sizes (see figure 3), so it would depend on whether one wants a quicker solution to a problem, or an overall better solution to the problem.

**References**

Harvey, Inman. Evolutionary Robotics and SAGA the case for Hill Crawling and Tournament Selection, 1992. Brighton, UK.

Harvey, Inman. The Microbial Genetic Algorithm, 2002. Brighton, UK.

Michalewicz, Zbigniew. A Modified Genetic Algorithm for Optimised Control Problems, 1992. Missouri, USA.

Mitchell, Melanie. An Introduction to Genetic Algorithms, 1998. Page 2. Massachusetts, USA.

Wall, Matthew. A Genetic Algorithm for Resource-Constrained Scheduling, 1996. Massachusetts, USA.

Whitley, Darrell. The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best, 1989. Colorado, USA.