AIAB spring assignment

Cadidate: 246504

May 2023

Abstract

This report explores the effect that the Crossover Probability and Deme size of a Microbial Genetic Algorithm has on the efficiency of said algorithm. Our aim was to find a optimal values for these parameters. We tested this over multiple iterations as well as varying numbers of tournaments/generations, and found that there was a clear optimal value for crossover probability but not for deme size.

1 Introduction

Genetic algorithms (GAs) are a form of optimisation, which are composed of a set of processes that evolve a population of candidate solutions to a given problem. Within this report we will go over a specific type: Microbial GAs and the effect that their Crossover probability and Deme size has on their efficiency.

2 Background

Genetic algorithms are the most prominent example of 'Evolutionary computation' [1], which is the idea of evolving a population of candidate solutions to a problem, using operators designed and inspired from natural genetic variation and selection [2]. They were initially developed to solve engineering problems, using real valued parameters to optimise things like airfoils.

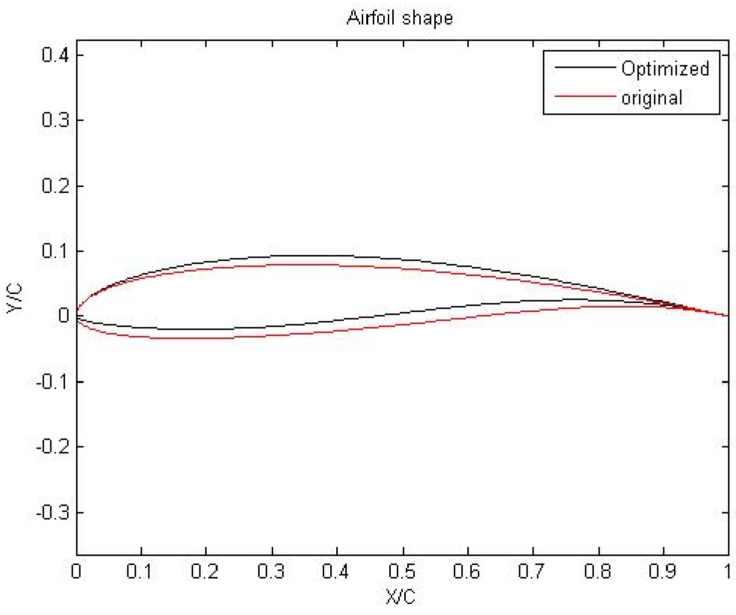


Figure 1: Figure shows an airfoil shape before and after being optimised by GA

The geometries of the airfoil are generated from an inverse method of velocity distribution [3]. Various combinations of these geometries are what the produced candidate solutions will be. The 'genotype' of each

solution is how it is encoded by the genetic algorithm e.g. a binary string which represents combinations of geometries. From this genotype, we can derive a phenotype which is the functional representation of our solution i.e. the shape of the airfoil.
In our experiment, we have chosen to test our GA on the Travelling Salesperson Problem (TSP): a widely recognised problem for benchmarking optimization methods.

2.1 Travelling Salesperson Problem

The TSP asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" [4]. To represent this problem, we randomly generated a list of unique sets of coordinates to represent the location of cities. We experimented with different numbers of cities, and found that 23 was an adequate medium between allowing for variation whilst saving computational power.

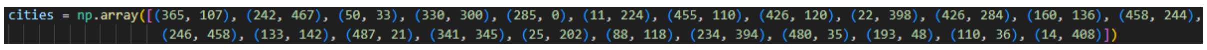
2.2 Microbial GAs

There are multiple types of Genetic Algorithms that exist, but we have chosen to focus on Microbial. Rather than simulating sexual reproduction of vertebrate organisms i.e. effective genotypes are combined to have 'children', instead Microbial GAs simulate the asexual gene transferal between eukaryotes. This allows us to keep the best individuals currently present within the population as they are not replaced by potentially worse children, which allows for a faster rate of growth. Furthermore, our Microbial GA is spatial; meaning genotypes can only transfer genetic material to genotypes within a region (Deme) of a given size. This is to simulate the geographical location of real-world microbes, but has an algorithmic advantage as well: Implementing these Demes helps to limit convergence of the GA [5] as genotypes tend not to be influenced by the others that are further away. As for the other parameter we are testing, Crossover probability is how likely an individual gene is to be transferred from a superior genotype to an inferior.

3 Methods

3.1 Problem Specification

As we mentioned previously, we randomly generated an array C of 23 tuples (x, y) to represent the cities.



```
cities = np.array([(365, 107), (242, 467), (50, 33), (330, 300), (285, 0), (11, 224), (455, 110), (426, 120), (22, 398), (426, 284), (160, 136), (458, 244),
                   (246, 458), (133, 142), (487, 21), (341, 345), (25, 202), (88, 118), (234, 394), (480, 35), (193, 48), (110, 36), (14, 408)])
```

Figure 2: Shows the array of coordinates

The next step was to decide how to encode our genotype. We decided this to be a list of 23 unique numbers ranging from 0-22. Each number would represent the index of one of a city, and the list would signify which order the salesperson visits them. So that the starting city is always the same, every genotype starts with a 0. When the GA is instantiated, a population of 100 of these genotypes is randomly generated. This population is then iterated through to determine which has the best fitness value, and this value is then stored.

3.2 Calculating fitness

To be able to determine how effective a genotype is, we need to define a fitness function which returns a number to represent the genotype's effectiveness. First, we calculated the total distance travelled, by calculating the sum of the Pythagorean distance between every adjacent city in a genotype g.
Let $C_i = (x_i, y_i)$.

$$TotalDistance = \sqrt{(x_{g_0} - x_{g_{22}})^2 + (y_{g_0} - y_{g_{22}})^2} + \sum_{i=1}^{22} \sqrt{(x_{g_i} - x_{g_{i-1}})^2 + (y_{g_i} - y_{g_{i-1}})^2} \tag{1}$$

Then, the fitness is calculated as follows:

$$Const = 1000000 \tag{2}$$

$$Fitness = \frac{Const}{TotalDistance} \tag{3}$$

The value of the constant $Const$ was purely to make the differences in fitness easier for us to read and plot, and can be any positive number without changing the hierarchy of the genotypes the GA produces.

3.3 Evolve Function

This is the main method within the GA. It will run the other methods within the GA a given number of times determined by the 'tournaments' parameter which specifies how many tournaments will be run.

3.3.1 Tournaments

In order to determine a superior genotype that will transfer its genes to an inferior genotype, two genotypes must be picked and evaluated against each other. The first genotype will be selected entirely at random, and the second genotype will then be selected within the Deme of the first. Then a winner of the two will be determined based on their respective fitness.

3.3.2 Crossover

After a winner has been determined, the next step is gene crossover. For every gene in the winning genotype, it has a given likelihood to replace the gene at the same index in the losing genotype. If the gene is crossed over, the gene is moved from its current index in the losing genotype to the same index as it is in the winning genotype; representing the new order in which the Salesperson will visit the cities.

3.3.3 Mutation

Finally, the losing genotype must be mutated. This is to employ slight variation that is completely independent of the winning or losing genotype; allowing for faster improvement. The mutation rate can vary depending on the GA, but In our GA, we randomly selected two indexes. Then we took the gene at the first index and moved it to the second index. In subsequent iterations of this GA, we could potentially test the effectiveness of increasing the number of genes that are mutated.

3.3.4 Evaluate new genotype

Once this newly produced genotype is put back into the population, its fitness is reevaluated and compared to the current best fitness that has been previously stored. If the new genotype is better, its fitness will replace the old best fitness.
Every time a tournament happens and a new genotype is added to the population, the best fitness in the population at that stage is recorded. This was so that once a GA had finished running, we could plot the change/increase in the best fitness over the course of the GA's runtime.

3.4 Evaluating the data

In order to find the best value for the crossover probability, we initially made multiple GAs with values ranging from 0.0 to 1.0 (incremented by 0.1). We calculated the average best fitness for every value and plotted them against the number of tournaments.
After determining the best value for crossover probability, we then made more GAs with Deme sizes ranging from 10 to 100, and repeated the process of average fitness.

4 Results

Below, you can see our initial plot.

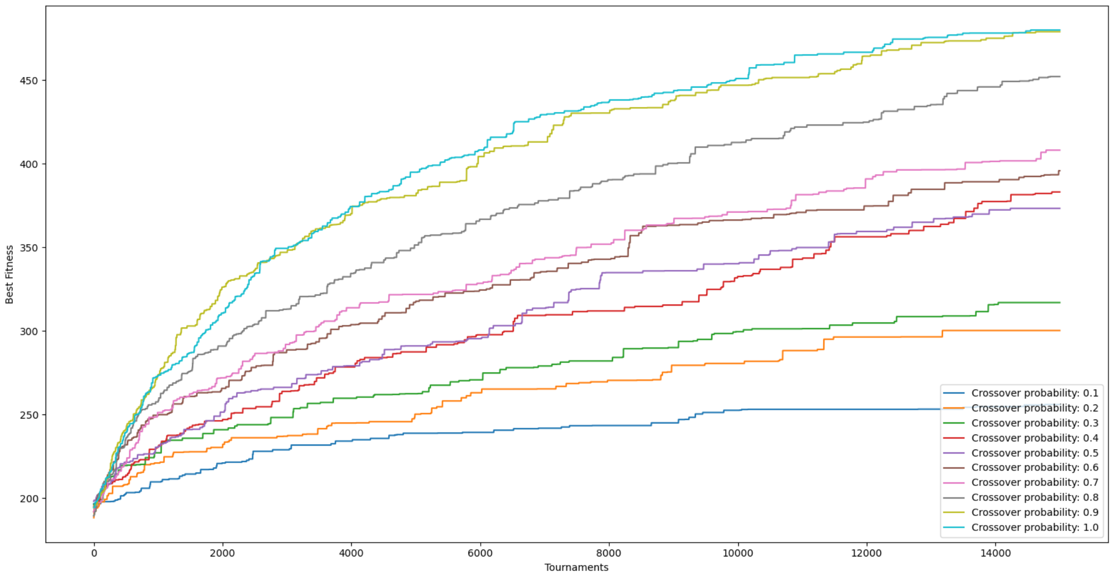


Figure 3: The average best fitness against number of tournaments for each value of CP

As you can see, figure 3 shows that the higher the crossover probability, the better. We ran this multiple times and the graphed looked roughly the same each time. This is likely due to the fact that the each gene's efficiency/desirability is dependent on the genes before it (the distance travelled is entirely dependant on the order), therefore copying over the entire sequence of genes is necessary to retain the superiority of the winning genotype in each tournament. It is clear that full replicating the winning genotype is the best option in this case, therefore allowing the mutation to be what ultimately progresses the genotype past its predecessor.
We then ran more GAs, this time with a locked crossover probability of 1.0, and varying Deme sizes as mentioned earlier.

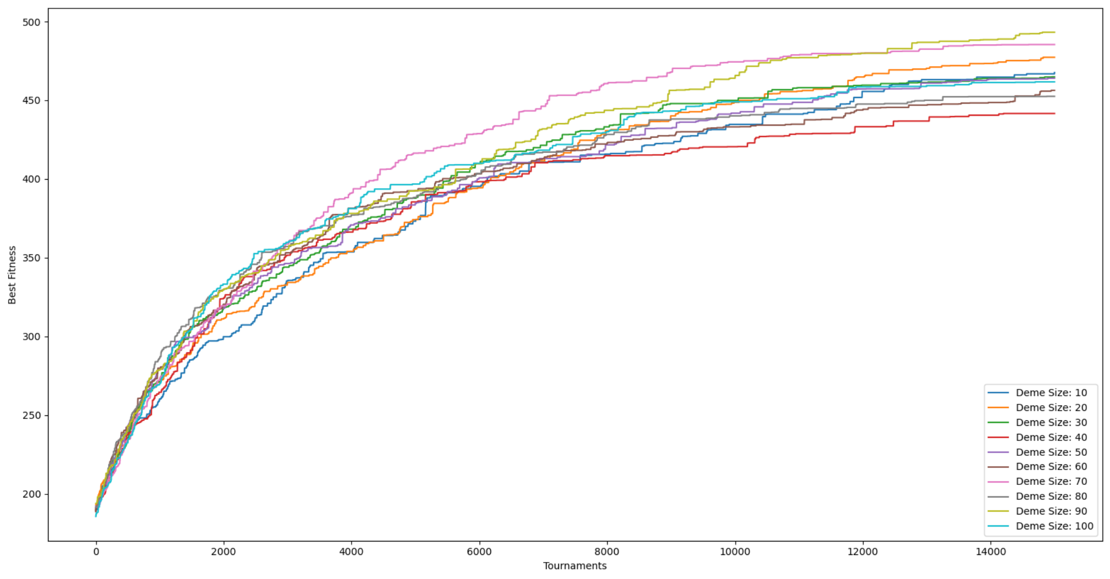


Figure 4: The average best fitness against number of tournaments for each Deme Size

Unlike the crossover probability, the Deme size seems to have much less of an effect on the outcome of the GA. This paired with the fact that maximum crossover being optimal suggests that the TSP (or this iteration of it) is inherently resistant to convergence. To make sure it is not an issue with the dataset we used, we decided to generate a new set of coordinates for the cities and test the Deme size on a new dataset.

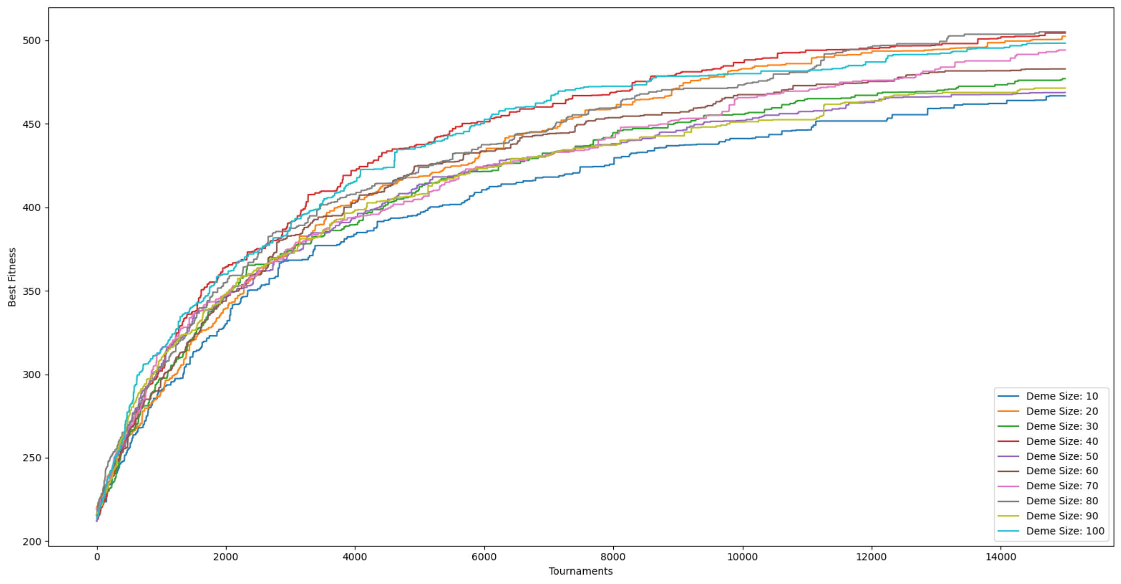


Figure 5: The average best fitness against number of tournaments for each Deme Size

Clearly, by demonstration in Figure 4 and Figure 5, there was no anomaly with occurring due to our data set. Furthermore, this implies that the TSP is a problem less afflicted by convergence issues than another optimization benchmark problem like the Knapsack problem [6].

5 Conclusion

When applied to the Travel Salesperson problem, the Crossover probability of our Microbial Genetic algorithm clearly has a benefitial effect on it's efficiency. Conversely, the Deme size seems to have a much less consistent effect on the outcome of the GA. Both of these findings could therefore imply that convergence is not as present within our GA compared to others or when tackling other optimization problems.

References

[1] Thomas Back and H-P Schwefel. Evolutionary computation: An overview. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 20–29. IEEE, 1996.

[2] Melanie Mitchell. *A brief history of evolutionary computation*, pages 2–3. MIT, 1998.

[3] Ben Gardner and Michael Selig. Airfoil design using a genetic algorithm and an inverse method. In *41st Aerospace Sciences Meeting and Exhibit*, page 43, 2003.

[4] Wikipedia contributors. Travelling salesman problem, 2023. Online; accessed 15-May-2023.

[5] RR Sharapov and AV Lapshin. Convergence of genetic algorithms. *Pattern recognition and image analysis*, 16:392–397, 2006.

[6] Wikipedia contributors. Knapsack problem, 2023. Online; accessed 23-May-2023.