# A Report on Genetic Algorithms and how they are affected by population size and population density

Adriaan Louw (53031377)

September 15, 2018

# Contents



# 1 Abstract

# 2 Introduction

This report will detail how Genetic Algorithms (GA) work and how they are used. This includes how to determine a good fitness function. Additionally will attempt answers 2 questions. Firstly how the choice of the initial population in the GA affects the effectiveness of the GA. Secondly whether maintaining diverse population is beneficial to a GA.

Genetic Algorithms are based on Darwinian evolution. This form of evolution was first described by Charles Darwin in 1859 in his seminal book *On the Origin of Species* (Darwin, 2004). In it he describes how life on earth developed through natural selection. Those species best adapted to their environment are more likely to survive and reproduce while those who are poorly adapted are less likely to survive and reproduce. So over time the traits of the better adapted (also called fit) individuals are passed more readily than those from less fit individuals. The whole population becomes better suited to their environment. Genetic Algorithms use these ideas (and others) to determine (evolve) a solution to a problem. In GA's the potential solution to a problem form the population. These individuals

are "evolved" to create new solutions based on the old solutions from the previous generation. The fitness of individuals in Genetic Algorithms are based on how well they solve the problem. The first description of this computing model was by Holland (1975). In it he rigorously defined the mechanics of the first Genetic Algorithm. This computing model can be applied to a varying range of problems.

# 3 Methodology

# 4 Analysis

## 4.1 How do Genetic Algorithms work?

As previously described a GA tries to use Darwinian ideas to generate a answer set to a problem.

### 4.1.1 Representation

The first problem in GA's is how to represent the problem space i.e. hypotheses on a computer.

A common way of representing these hypothesis for concept learning are by using bit strings (Mitchell, 1997). As an example assume assume we have a hypothesis space with boolean variables A and B. A particular hypothesis could be

$$(A = true \vee false) \wedge B = true \tag{1}$$

We can express this particular hypothesis as the binary string 1101 where the first 2 digits correspond to the values of A and the second 2 digits correspond to the values of B. Note that in the case of the first 2 digits, bother are set to 1 to indicate that A can be true or false. In the case of the second 2 digits only the last digit is set to 1. Corresponding to the fact that B can only be true.

We can see from this example that the number of bits required for each variable is exactly the same as the number of permutations a variable has. In other words if say a variable called has 10 options then it would take 10 bits to represent it.

These can be used to store rules. Assuming we have a rule that states when A=true and B=true or false then C = false, we can represent the condition as 4 bits "0111" as above and the result as "10". Then concatenate these strings to form our rule "011110".

### 4.1.2 Operators

After representing the hypothesis some operators need to be applied to the current population in order for a new generation to be created.

The crossover operator is used to create 2 new strings from 2 parents (Mitchell, 1997). Bits from each parent is selected and then combined to create the offspring. The idea is to imitate how chromosomes in living organisims exchange genetic information. This is done in GA's through the use of a crossover mask.

In single-point crossover (Mitchell, 1997), bits 1 to i of the first parent are combined with the i+1 to n bits of the other parent. Where i is the crossover point and n is the last bit. An example of this would be given 2 8-bit strings, we use a crossover mask like 11110000. The first 4 bits of the one string will be concatenated to the last 4 bits of the second string. To add to the randomness, the crossover point is chosen at random each time. Crossover is not usually applied to every member of the population (Beasley, Bull, & Martin, 1993). A probability is chosen, usually between 0.6 and 1.0, to decide whether the crossover will be applied. If a crossover is not applied, then the parent will join the offspring population unaltered.

Two-point crossover defines, like the name suggests, 2 crossover points. These 2 points cannot be at the start or end of the string. The idea is to replace a middle section of the one parent with a middle section of another (Mitchell, 1997). For example, given a crossover mask 00111100, the first 2 bits and the last 2 bits of one parent is combined with the middle 4 bits of the other parent. Just as in single crossover, the crossover points are randomly selected each time a crossover is to be applied.

In uniform crossover, the bits of the crossover string is chosen at random (Mitchell, 1997). For instance we can have a 8-bit crossover mask like 01110101. Each bit is chosen independently from the other bits. Also like the other crossover examples the crossover mask is regenerated for each use.

The mutation operator attempts to replicate the random mutations tat happen in the DNA of living organisms. In the simplest case a a random bit in member of the population is flipped (Mitchell, 1997). In a 4-bit entity like 0101, applying mutation might cause the string to become 0111.

### 4.1.3 Selection and fitness function

Now that we have created a new generation of candidates we need to be able to determine which of these candidates solve the problem best. Only those candidates can be used to create the next generation. The function to determine how well a candidate solves the problem is called the fitness function. The fitness function is highly problem/domain specific. The designer of the GA needs to decide which features of each candidate is most important in determining its ability to solve the problem (Beasley et al., 1993).

Based on input from the fitness function, the selection process then selects those candidates that are to continue to the next generation. Here follows a short description of some of the more common selection schemes.

#### 4.1.3.1 Roulette wheel selection

In this method the probability that an individual is selected is the ratio of its fitness to that of all the other candidates (Mitchell, 1997). Given fitness function $f$ and candidate $c_i$ out of a possible $n$ candidates the the probability that $c_i$ will be selected will be given by

$$c_i = \frac{f(c_i)}{\sum_{j=1}^{n} f(c_j)} \tag{2}$$

The roulette wheel is a very popular selection method because of its simplicity to implement and to understand (Lipowski & Lipowska, 2011). This method was introduced by De Jong (1975).

#### 4.1.3.2 Ranking Selection

Ranking selection was proposed by Baker (1985). The candidates are first sorted from best to worst according to their fitness (via the fitness function). Then the probability that a candidate will be selected will be proportional to its rank. With the fitter candidates having a better chance to be selected than the less fit ones.

#### 4.1.3.3 Tournament Selection

According to (Mitchell, 1997), in tournament selection, 2 candidates are selected at random. The fitter of the 2 candidates is the "winner". Given some predefined probability $p$, the probability that the winner will be selected is $p$ and the probability that the "loser" is selected is $1-p$. This process is followed until enough candidates are selected for the next generation.

## 4.2 How does the choice of initial population affect Genetic Algorithms?

The size of the initial population has a "direct effect" on various aspects of a GA (Gupta & Others, 2014). Firstly with larger populations the GA requires more memory. Thus on memory intensive domains this will detrimental. Secondly the convergence speed and search speed of the algorithm will also be affected by the population size. There is a balance that needs to be found in the size of the population.

When the initial population is large the effectiveness of the algorithm increases. One reason for this is that there is a greater chance that the necessary "building blocks" are present in the population (Harik, Cantu-Paz, Goldberg, & Miller, 1999). The "building blocks" are not likely to be created via mutation. But increasing the population size only help up to a point unit it becomes computationally too expensive to run each generation (Gupta & Others, 2014). This can be due to either the selection process and/or the crossover and mutation process becoming too computationally expensive. Leading to fewer generations can be computed in the same amount of time.

Conversely, a small population will have fewer opportunities for crossover and mutation thus potentially offsetting the computational benefit of having a small population. But a smaller population also tends to become homogeneous more rapidly meaning that any potential variation that was in the population, that could have contributed more to a potential solution, could be lost (De Jong & Spears, 1990). In the extreme case of using only 1 individual for the initial population means that the convergence rate will be low even with a high mutation rate (Whitley, 2001).

### 4.3 How does the diversity of the population affect Genetic Algorithms?

## 5 Conclusion

# References


Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. In *Proceedings of an international conference on genetic algorithms and their applications* (pp. 101–111).

Beasley, D., Bull, D. R., & Martin, R. R. (1993). *An Overview of Genetic Algorithms : Part 1, Fundamentals* (Vol. 15; Tech. Rep. No. 2). Retrieved from `http://orca.cf.ac.uk/64436/http://orca.cf.ac.uk/policies.htmlforusagepolicies.`

Darwin, C. (2004). *On the origin of species, 1859*. Routledge.

De Jong, K. A., & Spears, W. M. (1990). An analysis of the interacting roles of population size and crossover in genetic algorithms. In *International conference on parallel problem solving from nature* (pp. 38–47). Retrieved from `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.51.630&rep=rep1&type=pdf`

De Jong, K. A. (1975). Analysis of the behavior of a class of genetic adaptive systems.

Gupta, R., & Others. (2014, jun). Effect of varying the size of initial parent pool in genetic algorithm. In *Contemporary computing and informatics (ic3i), 2014 international conference on* (pp. 785–788). IEEE. Retrieved from `http://ieeexplore.ieee.org/document/5235649/` doi: 10.1109/FCC.2009.81

Harik, G., Cantu-Paz, E., Goldberg, D. E., & Miller, B. L. (1999). The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary computation*, *7*(3), 231–253. doi: 10.1162/evco.1999.7.3.231

Holland, J. (1975). Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. *USA: University of Michigan*.

Lipowski, A., & Lipowska, D. (2011). *Roulette-wheel selection via stochastic acceptance* (Tech. Rep.). Retrieved from `https://arxiv.org/pdf/1109.3627.pdf`

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Whitley, D. (2001). *An overview of evolutionary algorithms: practical issues and common pitfalls* (Tech. Rep.). Retrieved from `www.elsevier.com/locate/infsof`