1 Genetic Algorithms

The genetic algorithm (GA) was first introduced by John Henry Holland, even though he did not call it the genetic algorithm at the time. This section gives an overview of the workings and most important operations of the genetic algorithm. If not otherwise stated, everything in this section is based on Holland's book "Adaptation in natural and artificial systems" as well as Goldberg's book "Genetic algorithms in search, optimization and machine learning" [J. Holland, 1975] [D. Goldberg 1989].

1.1 Simple Genetic Algorithms (SGAs)

Genetic algorithms are probabilistic search algorithms inspired by evolution. By performing selection based on survival of the fittest and genetic operations, genetic algorithms are able to solve problems that are too complex to be modeled mathematically. By utilizing evolution as search strategy, genetic algorithms are able to find sophisticated solutions to problems where the structure of the solution is complex, maybe even unknown.

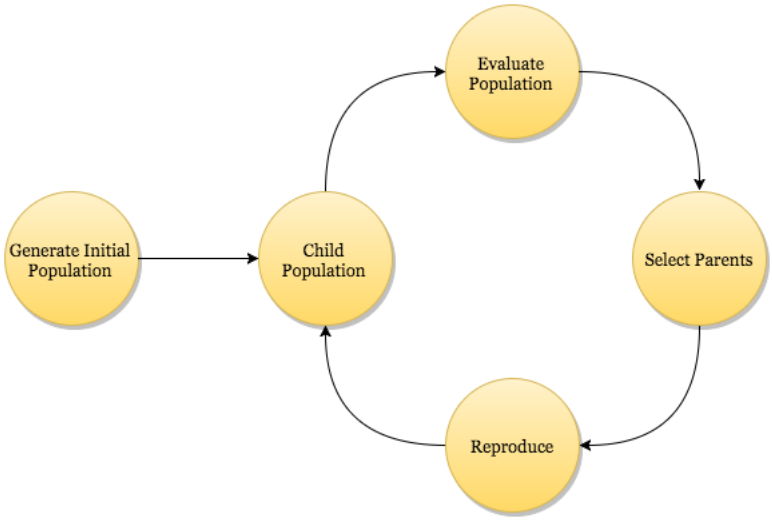

Figure 1: Overview of the phases of the genetic algorithm. First an initial population is generated and initialized as the first child population. Second, each individual in the child population is evaluated. Third, based on their evaluation, individuals are selected to become parents for the next generation. Forth, a new generation is generated by reproduction, and fifth, the newly generated population becomes the child population.

Genetic algorithms starts out by randomly generating a set of solutions to a problem, the set of solutions constitutes the population of the first generation of solutions. Next, each individual in the initial population is evaluated based on some predetermined objective function. Based on their objective function values, the fittest individuals are selected for reproduction, meaning that they

become the parents of the next generation. By utilizing different genetic operations on the parent solutions a new child population is generated, hopefully with higher average fitness than the previous population since it is based on the fittest individuals from the previous population, and the process repeats itself until the population has found an desirable solution - a population of high-fitness solutions. The different phases of the genetic algorithm is displayed in figure 1.

Two key properties are crucial for the genetic algorithm to be useful, (a) there has to be a way to measure the fitness of the individuals, (b) there need to be a way to represent the individuals so that genetic operations can be performed on them. The section below discuss how individuals usually are represented for genetic algorithms.

1.1.1 Representation

In genetics, they call an organisms hereditary information for its genotype, and its observable properties its phenotype. For example, the hereditary information in your genes (genotype) are responsible for your eye color (phenotype).

The genetic search algorithm works on genotypes represented as bit strings. Goldberg explained this with a simple example. Let's say the objective function that we want to find an optimal solution for is x^2 for $x \in \{0, 31\}$. Then we can generate genotypes for the random solutions using bit strings of size 5, each representing a phenotype value between 0 and 31. Figure 2 displays the genotype and phenotype for four randomly generated individuals for two generations. Here, the phenotypes are just the genotypes on decimal form, but in other problems the phenotype could be everything from eye color to a wind farm.

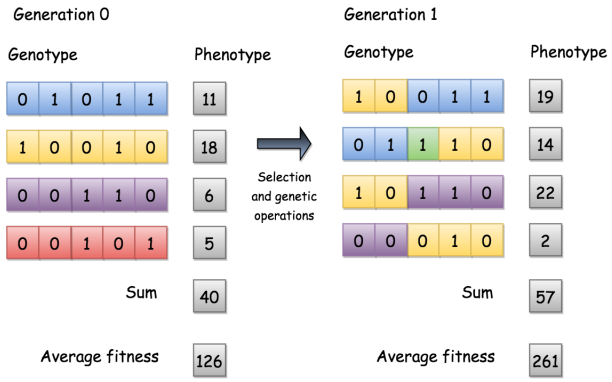

Figure 2: Genotypes and phenotypes for four individuals in two generations. On the left side is the randomly generated first generation, and on the right side is the next generation after selection, mutation (red square) and crossover operations. The fitness function used is x^2 and as shown the average fitness of the populations increases from 126 to 261 in only one generation.

The different colors of each individual in figure 2 is just there so that it is easy to understand the genetic operations. The operations will be explained in the section below, for now just accept

that some fitness-based selection and genetic operations are performed on the individuals to the left to produce the individuals to the right. As you can see the average phenotype values increases from generation 0 to generation 1, and the average fitness, which is just the average of the squared phenotype values, increases form 126 yo 261 in only one generation.

1.1.2 Operations

The sections above explain that the genetic algorithm uses selection of the fittest and genetic operations, but not exactly how this work. This section will explain different selection strategies and the most common genetic operations - mutation and crossover.

1.1.2.1 Selection

Selection is the process of selecting which individuals from a given population that will be the parents of the next generation. It can be done in a number of ways, and some of the most popular strategies will be presented here.

The simplest form of selection is called *elitist selection*, meaning that the n best individuals from the population are selected as parents for the next generation. This strategy is extremely simple, but unfortunately not very good because choosing only the best individuals each generation often leads to fast convergence of an non-optimal solution. It is important to prioritize exploration, at least in the beginning of the search, otherwise, parts of the search space that could have lead to the optimal solution is cut off too soon. To maintain diversity in the group long enough to find high-fitness solutions *controlled elitist selection* is preferred.

A very popular controlled elitist selection strategy is *tournament selection* [Razali et al., 2011]. In tournament selection, groups of n individuals are randomly drawn from the population and the best (fittest) individual from the group is chosen as the tournament winner, and is therefore selected. Figure 3 illustrates how tournament selection works. In the example, n is equal to 3, therefore the three individuals with fitness 9, 4 and 6 are randomly drawn from the population. The individual with fitness 9 wins the tournament and is chosen for reproduction.

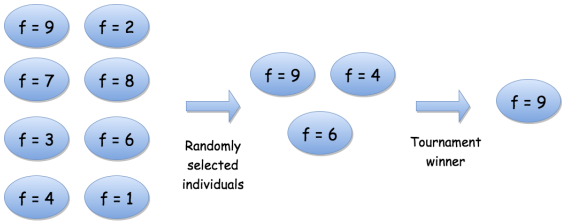

Figure 3: Tournament selection. A group of three individuals are randomly drawn from the pool of all individuals. The best individual in the group, the one with fitness 9, is selected for reproduction.[Razali et al., 2011]

By varying the value of n you can control how much exploration your algorithm should do. If n is equal to the population size, this is elitist selection, if n is equal to 1 however the search is

completely random. This means that low values of n leads to more exploration of the search space, and higher values of n leads to faster convergence. These properties makes it desirable to vary the value of n during the genetic search so that exploration is prioritized at the beginning of the search, while exploitation is prioritized at the end.

1.1.2.2 Mutation

In biology, mutation is defined as a permanent alteration in the DNA sequence that makes up a gene. Mutations vary in size, sometimes just a small base pair of the gene is altered, while other times the mutation can alter large parts of a chromosome. There are many different forms of mutations. Sometimes it can simply mean changing the value of a nucleotide to its complement (nucleotieds are the building blocks of DNA and consists of the values A, C, G and T), other times it can mean insertion of extra nucleotides into the DNA, deleting some of the nucleotides from the DNA, or invertion of nucleotides [P. Compeau, 2014].

In genetic computation the mutation process is simplified. Since the search space only consist of strings of bits with constant size, a mutation in genetic computing simply consists of flipping of bits, this is called single point mutation. In Figure 4 we have a bit string of size 8, where the 6th element is mutated from the value 1 to the value 0. Mutation is usually implemented by having a given probability of each value in the genotype being flipped.

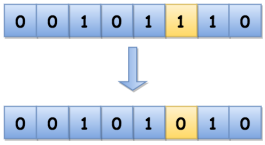

Figure 4: Mutation of a single bit. The bit in position 6 at the upper bit string has the value 1 before the mutation, while after mutation the value if flipped into 0.

Mutation is important because without mutation a population can converge to a population of individuals where each genotype has the same value at a given position. Since every individual has the same value in their genotype, reproduction will never be able to make a new individual that doesn't also have the same value at the same position. With mutation however, there is always a probability of the value being flipped, mutation is therefore crucial for maintaining diversity in the population.

Even though mutation is important, the probability of mutation needs to be kept low. If the mutation rate is very high, the genotype of a new individual will almost be a random bit string. Remember that a new individual is made by reproduction between two individuals with high fitness in the previous population, if mutation heavily changes the new individual, it will not inherit the good features of its parents and the whole point of evolutionary search will be gone.

1.1.2.3 Crossover

As in nature, we want individuals of the evolutionary search to inherit features from individuals in the previous population. In genetic algorithms this is performed by applying the crossover

operation. Crossover is a simple method where a new individual gets some parts of its genotype from one parent, and the other parts from its other parent. Figure 5 illustrates the crossover operation. Before crossover is performed one needs to decide at which position it will be performed. If we are operating with genotypes of length 8 as in figure 5 there will be 7 possible crossover positions, one between each of the 8 values of the genotype. In figure 5 position 4 is picked as the crossover position, therefore the first of the newly produced individuals will have a genotype consisting of the first 4 values of the first individual of the old population and the 4 last values of the second individual of the old population, while the second newly generated individual will get the opposite genotype.

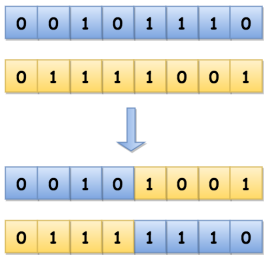

Figure 5: Single point crossover at position 4.

The crossover in figure 5 is called a single point crossover, because crossover is only performed at one position. It is also possible to have multiple crossover positions as shown in figure 6, where a double point crossover is performed with crossover positions 2 and 5.

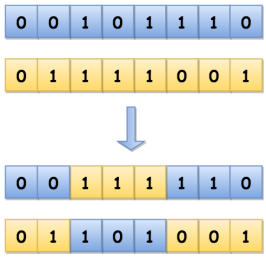

Figure 6: Double point crossover at position 2 and 5.

Crossover probability is usually kept high because one is hoping that combining to good solutions from the previous population will produce even better solutions for the next population. However, it is desirable to keep some solutions from the previous population as they are, because they might be better then combination of two of them.

Goldberg performs a very interesting analysis of the crossover operation by studying the probability of surviving a crossover operation for different schemata. A schema is a string taken from the library {0, 1, *}. A shema of size 8 could therefore for example look like this *0011*****, where *0* and *1* represents letters with values 0 and 1, while * represents letters that we don't care about. This means that both strings *00110000* and *00111111* are covered by the given schema. If desirable

features are represented by schemata it is easy to see that some features are much more likely to survive a single point crossover operation than others. Lets study the shemata $s_1 = 11******$ and $s_2 = 1******1$. Both has size 8, meaning that there are 7 possible crossover positions. However, only one of the crossover positions will destroy s_1 while all 7 positions will break up s_2. This means that the genetic search would converge faster if s_1 was the desirable feature, then if s_2 was.