

## Genetic Algorithm

Genetic algorithm, firstly introduced by Holland<sup>1</sup>, is an iterative procedure that consists of a constant-size population of individuals, each one represented by a finite string of symbols, known as the genome, encoding a possible solution in a given problem space. This space, referred to as the search space, comprises all possible solutions to the problem at hand. Generally speaking, the genetic algorithm is applied to spaces which are too large to be exhaustively searched. The symbol alphabet used is often binary, though other representations have also been used, including character-based encodings, real-valued encodings, and -- most notably -- tree representations.

The standard genetic algorithm proceeds as follows: an initial population of individuals is generated at random or heuristically. Every evolutionary step, known as a generation, the individuals in the current population are decoded and evaluated according to some predefined quality criterion, referred to as the fitness, or fitness function. To form a new population (the next generation), individuals are selected according to their fitness. Many selection procedures are currently in use, one of the simplest being Holland's original fitness-proportionate selection, where individuals are selected with a probability proportional to their relative fitness. This ensures that the expected number of times an individual is chosen is approximately proportional to its relative performance in the population. Thus, high-fitness ("good") individuals stand a better chance of "reproducing", while low-fitness ones are more likely to disappear.

Selection alone cannot introduce any new individuals into the population, i.e., it cannot find new points in the search space. These are generated by genetically-inspired operators, of which the most well known are crossover and mutation. Crossover is performed with probability  $p_{\text{cross}}$  (the "crossover probability" or "crossover rate") between two selected individuals, called parents, by exchanging parts of their genomes (i.e., encodings) to form two new individuals, called offspring; in its simplest form, substrings are exchanged after a randomly selected crossover point. This operator tends to enable the evolutionary process to move toward "promising" regions of the search space. The mutation operator is introduced to prevent premature convergence to local optima by randomly sampling new points in the search space. It is carried out by flipping bits at random, with some (small) probability  $p_{\text{mut}}$ . Genetic algorithms are stochastic iterative processes that are not guaranteed to converge; the termination condition may be specified as some fixed, maximal number of generations or as the attainment of an acceptable fitness level. Below we present the standard genetic algorithm in pseudo-code format.

```
begin GA
  g:=0 { generation counter }
  Initialize population P(g)
  Evaluate population P(g) { i.e., compute fitness values }
  while not done do
    g:=g+1
    Select P(g) from P(g-1)
    Crossover P(g)
    Mutate P(g)
    Evaluate P(g)
  end while
end GA
```

Let us consider the following simple example demonstrating the genetic algorithm's workings. The population consists of 4 individuals, which are binary-encoded strings (genomes) of length 8. The fitness value equals the number of ones in the bit string, with  $p_{\text{cross}}=0.7$ , and  $p_{\text{mut}}=0.001$ . More typical values of the population size and the genome length are in the range 50-1000. Also note that fitness computation in this case is extremely simple since no complex decoding nor evaluation is necessary. The initial (randomly generated) population might look like this:

Label	Genome	Fitness
A	00000110	2
B	11101110	6
C	00100000	1
D	00110100	3

Using fitness-proportionate selection we must choose 4 individuals (two sets of parents), with probabilities proportional to their relative fitness values. In our example, suppose that the two parent pairs are {B,D} and {B,C} (note that A did not get selected as our procedure is probabilistic). Once a pair of parents is selected, crossover is effected between them with probability  $p_{\text{cross}}$ , resulting in two offspring. If no crossover is effected (with probability  $1-p_{\text{cross}}$ ), then the offspring are exact copies of each parent. Suppose, in our example, that crossover takes place between parents B and D at the (randomly chosen) first bit position, forming offspring E=10110100 and F=01101110, while no crossover is effected between parents B and C, forming offspring that are exact copies of B and C. Next, each offspring is subject to mutation with probability  $p_{\text{mut}}$  per bit. For example, suppose offspring E is mutated at the sixth position to form E'=10110000, offspring B is mutated at the first bit position to form B'=01101110, and offspring F and C are not mutated at all. The next generation population, created by the above operators of selection, crossover, and mutation is therefore:

Label	Genome	Fitness
E'	10110000	3
F	01101110	5
C	00100000	1
B'	01101110	5

Note that in the new population, although the best individual with fitness 6 has been lost, the average fitness has increased. Iterating this procedure, the genetic algorithm will eventually find a perfect string, i.e., with maximal fitness value of 8.

---

<sup>1</sup> J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, Michigan, 1975.