Evolutionary Computation:

Genetic algorithms

- Introduction: can evolution be intelligent?
- Genetic algorithms
- Case study

Can evolution be intelligent?

- Intelligence can be defined as the capability of a system to adapt its behavior to ever-changing environment.

- Evolutionary computation simulates evolution on a computer. The result of such a simulation is a series of optimization algorithms, usually based on a simple set of rules.

- Optimization iteratively improves the quality of solutions until an optimal, or at least feasible, solution is found.

- The evolutionary approach is based on computational models of natural selection and genetics.

- We call them evolutionary **computation**, an umbrella term that combines g**enetic algorithms**, **evolution strategies** and **genetic programming**.

Simulation of natural evolution

- All methods of evolutionary computation simulate natural evolution by creating a population of individuals, evaluating their fitness, generating a new population through genetic operations, and repeating this process a number of times.
- We will start with **Genetic Algorithms** (GAs) as most of the other evolutionary algorithms can be viewed as variations of genetic algorithms.

Genetic Algorithms

- In the early 1970s, John Holland introduced the concept of genetic algorithms.
- His aim was to make computers do what nature does. Holland was concerned with algorithms that manipulate strings of binary digits.
- Each artificial “chromosomes” consists of a number of “genes”, and each gene is represented by 0 or 1:

1	0	1	1	0	1	0	0	0	0	0	1	0	1	0	1

- Two mechanisms link a GA to the problem it is solving: **encoding** and **evaluation**.

- The GA uses a measure of fitness of individual chromosomes to carry out reproduction. As reproduction takes place, the crossover operator exchanges parts of two single chromosomes, and the mutation operator changes the gene value in some randomly chosen location of the chromosome.

Basic genetic algorithms

Step 1: Represent the problem variable domain as a chromosome of a fixed length, choose the size of a chromosome population N, the crossover probability p_c and the mutation probability p_m.

Step 2: Define a fitness function to measure the performance, or fitness, of an individual chromosome in the problem domain. The fitness function establishes the basis for selecting chromosomes that will be mated during reproduction.

Step 3: Randomly generate an initial population of chromosomes of size *N*:

$x_1, x_2, \ldots, x_N$

Step 4: Calculate the fitness of each individual chromosome:

$f(x_1), f(x_2), \ldots, f(x_N)$

Step 5: Select a pair of chromosomes for mating from the current population. Parent chromosomes are selected with a probability related to their fitness.

Step 6: Create a pair of offspring chromosomes by applying the genetic operators – **crossover** and **mutation**.

Step 7: Place the created offspring chromosomes in the new population.

Step 8: Repeat *Step 5* until the size of the new chromosome population becomes equal to the size of the initial population, *N*.

Step 9: Replace the initial (parent) chromosome population with the new (offspring) population.

Step 10: Go to *Step 4*, and repeat the process until the termination criterion is satisfied.

Genetic algorithms

- GA represents an iterative process. Each iteration is called a **generation**. A typical number of generations for a simple GA can range from 50 to over 500. The entire set of generations is called a **run**.

- A common practice is to terminate a GA after a specified number of generations and then examine the best chromosomes in the population. If no satisfactory solution is found, the GA is restarted.