

# Lecture notes, Machine learning and robotics

Kristian Nymoen

May 2, 2011

## 1 Evolutionary algorithms

### 1.1 inspiraton from biology

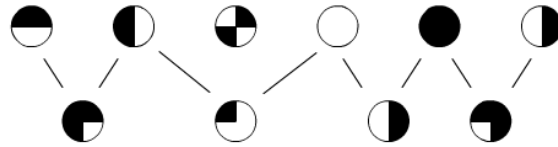


Figure 1: Evolution in biology - some individuals are more fit than others and will be able to reproduce. Properties of the parents can be found in the offspring. In this way, species in nature evolve into more fit species and adapt to changing conditions such as climate changes etc.

By implementing the same principle in computer algorithms, we can find near-optimal solutions to difficult problems.

### 1.2 Overview of evolutionary algorithms

Pseudocode for a typical genetic algorithm:

```
BEGIN
  INITIALIZE population with random solutions
  EVALUATE each candidate
  REPEAT UNTIL (TERMINATION CONDITION is satisfied) DO
    1 SELECT parents
    2 RECOMBINE pairs of parents
    3 MUTATE the resulting offspring
    4 EVALUATE new candidates
    5 SELECT individuals for the next generation
  OD
END
```

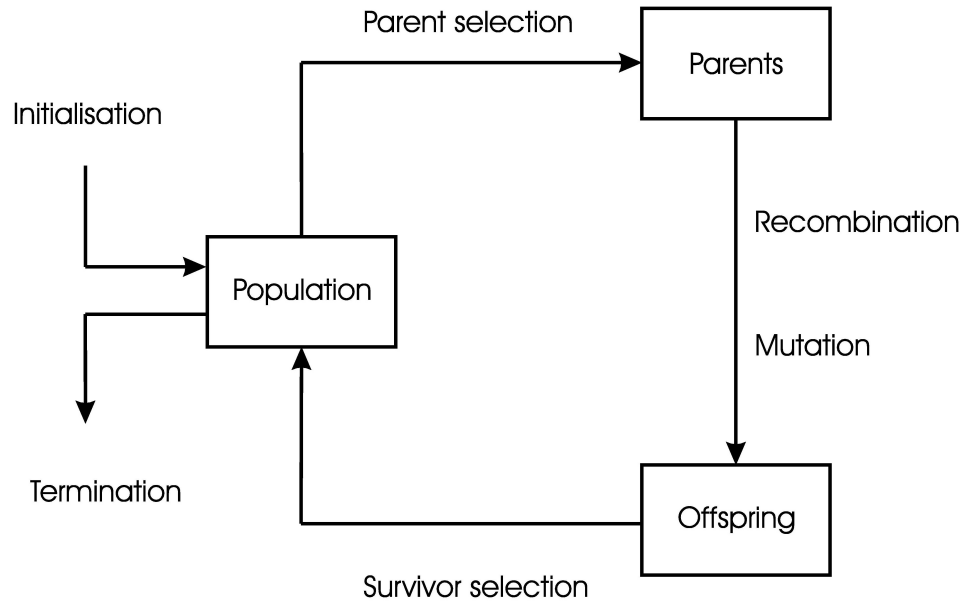


Figure 2: Overview of an evolutionary algorithm

### 1.3 Operators in an evolutionary algorithm

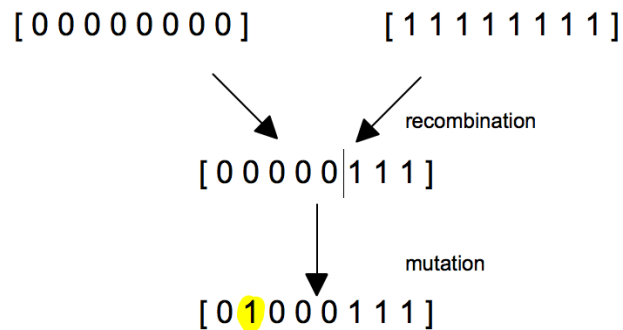


Figure 3: Recombination and mutation. The most common operators in an evolutionary algorithm

**Recombination** Combines two “parents” to create an “offspring”.  
 The offspring has properties from both parents.

**Mutation** A (usually small) change in the chromosome.  
 For instance by flipping a bit from 0 to 1.

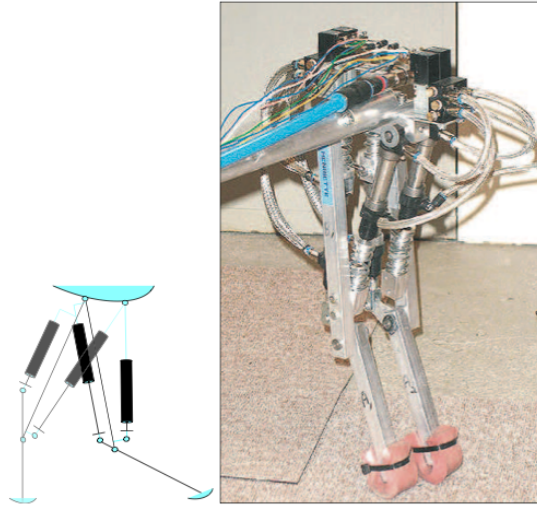


Figure 4: The “Henriette” robot

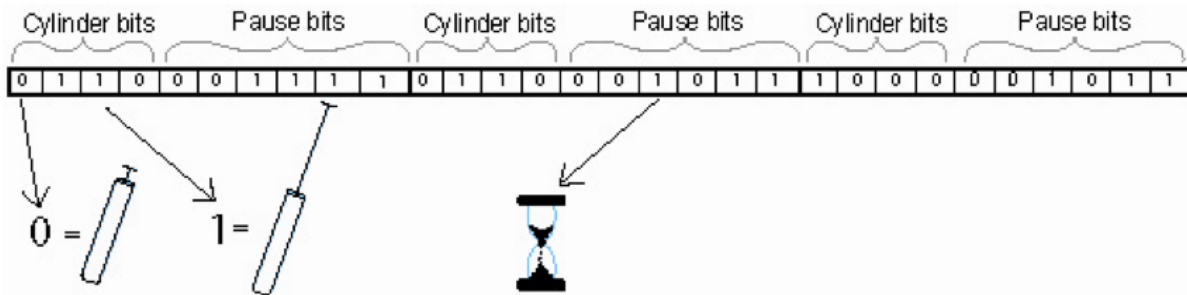


Figure 5: The “Henriette” chromosome

### 1.4 An example of evolutionary algorithms in robotics

In the chicken robot “Henriette” the walking pattern was coded into bit strings. The walking chromosome consisted of a sequence of 3 different “states”. Each state consisted of a leg configuration and a pause length. An evolutionary algorithm was used to evolve the leg configurations and the pause length.

For each leg configuration, 4 bits denote the position of 4 actuators, 6 bits denote the length of the pause. - Resulting in a chromosome consisting of 30 bits, as shown in Figure 5.

Another example is the mars rover which is able to adapt to changing conditions such as hardware malfunctions. - If a wheel falls off, the robot will adapt to this condition by trying out different configurations, and eventually it will be able to continue moving.

## 2 Reinforcement learning - a simple example

Can (among other things) be used for *mapping* in robotics. An autonomous robot must be able to construct a *map* or floor plan and to localize itself in it.

In a simple 2 x 3 room, a robot tries out different paths to create a map of its surroundings. At first, it does not know anything except for the “legal” moves. Figure 6 shows the 6 different states (positions) that the robot can be in, and a so-called *Q-table* with the scores associated with each action that can be taken in each state.

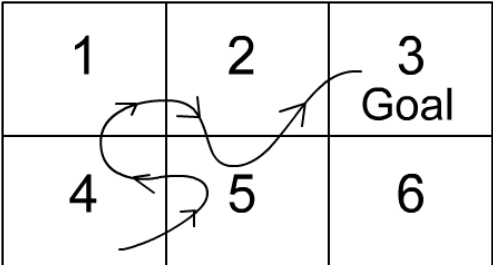
1	2	3 Goal
4	5	6

Initial Q-table				
State	Move up	Move down	Move left	Move right
1)	-	0	-	0
2)	-	0	0	0
3)	-	-	-	-
4)	0	-	-	0
5)	0	-	0	0
6)	0	-	0	-

Figure 6: The 2x3 room and the initial Q-table

We begin the training by putting the robot in position 4. It moves along the trajectory shown in Figure 7. Reaching the goal gives an award of 100, so the final move: *move right from state 2* is awarded a score of 100.

1	2	3 Goal
4	5	6



Episode 1				
State	Move up	Move down	Move left	Move right
1)	-	0	-	0
2)	-	0	0	100
3)	-	-	-	-
4)	0	-	-	0
5)	0	-	0	0
6)	0	-	0	-

Figure 7: First training trajectory, and resulting score table

We conduct one more training session, placing the robot in position 1. It moves along the trajectory shown in Figure 8. When reaching square 2, the highest possible score in the next move

is 100, so this move should be awarded, but not as high as reaching the final goal. We use a scaling factor of 0.9 compared to the highest possible score from state 2, giving a score of 90 for the move: *move right from state 1*. The robot continues its random walk, and eventually reaches the goal by *moving up from state 6*, yielding a score of 100.

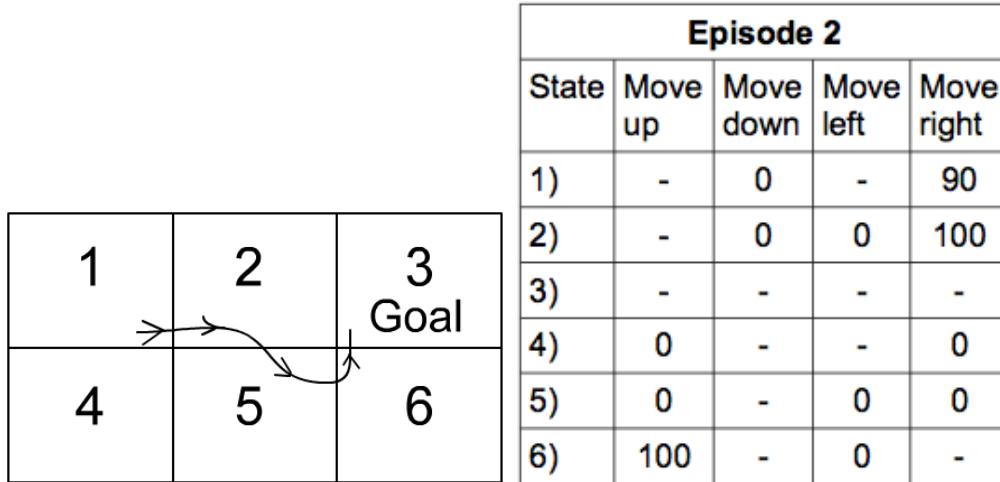


Figure 8: Second training trajectory, and resulting score table

If this is continued, the Q-table will eventually converge towards the table shown in figure 9. This presents a complete list of all possible moves for the robot in any possible state. By using this table the robot can always find the fastest possible way to the goal.

Complete Q-table				
State	Move up	Move down	Move left	Move right
1)	-	72.9	-	90
2)	-	81	81	100
3)	-	-	-	-
4)	81	-	-	81
5)	90	-	72.9	90
6)	100	-	81	-

Figure 9: Completed Q-table