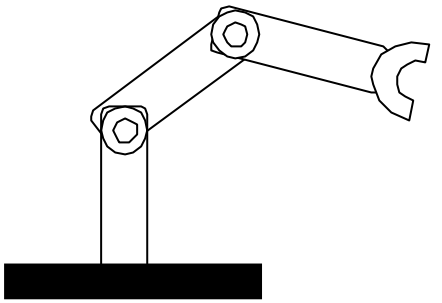
Lecture 7: Evolving neural network controllers for mobile robots

Purpose of this lecture

- In this lecture, we will study the GA based learning of neural network robot controllers
- After this lecture, you should be able to understand
  - The basic idea of neural network based robot control
  - The basic properties of a mobile robot
  - The basic considerations for evolving the neural network controller

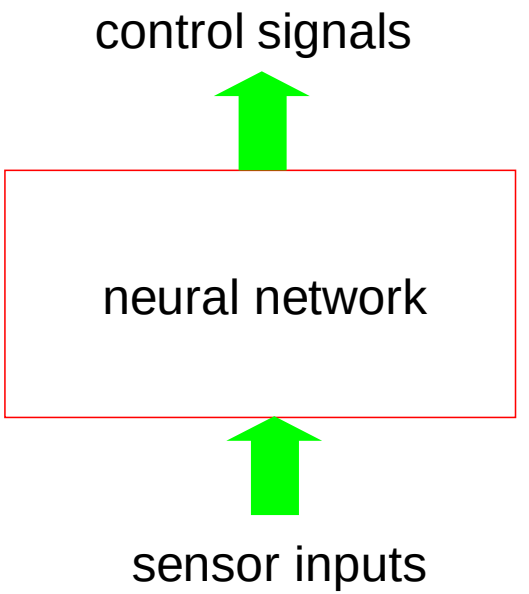
Applications of neural network controller

- Neural networks can be used to control a robot in different ways
  - Control each link of a robot arm
  - Control the motors of mobile robots
  - Control a robot so that it can approach to a target most quickly



Basic idea of neural network controller

- Information obtained from different sensors are used as the inputs
  - Usually some kind of pre-processing is necessary
  - Feature extraction/selection is also important for reducing the computation amount
- The outputs are used as the control signals
  - A decoder might be useful

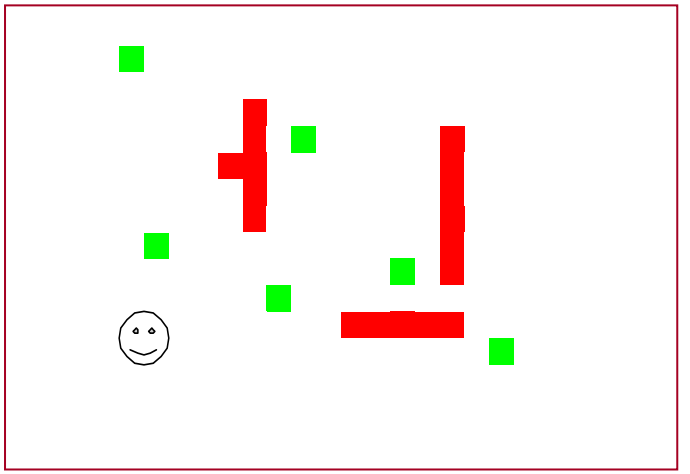


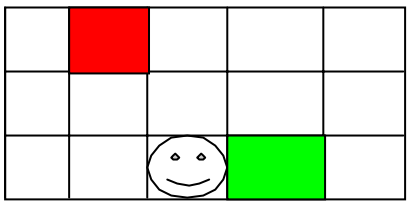
Models for neural network controllers

- Multi-layer neural network (MLP)
  - An MLP is usually powerful enough
- Time-delay neural network (TDNN)
  - Each neuron is replaced by an FIR/IIR filter
  - Historical information can be used
  - The robot can be more efficient
- Recurrent neural network (RNN)
  - A neural network with feedbacks
  - Can be approximated by an automaton
  - Useful for goal approaching robots
  - Difficult to learn

A simple example: virtual robot and environment

- Shown in the right is a virtual environment containing
  - A robot,
  - Some foots, and
  - Some obstacles
- This kind of virtual environment is often used to investigate the efficiency/efficacy of evolutionary learning

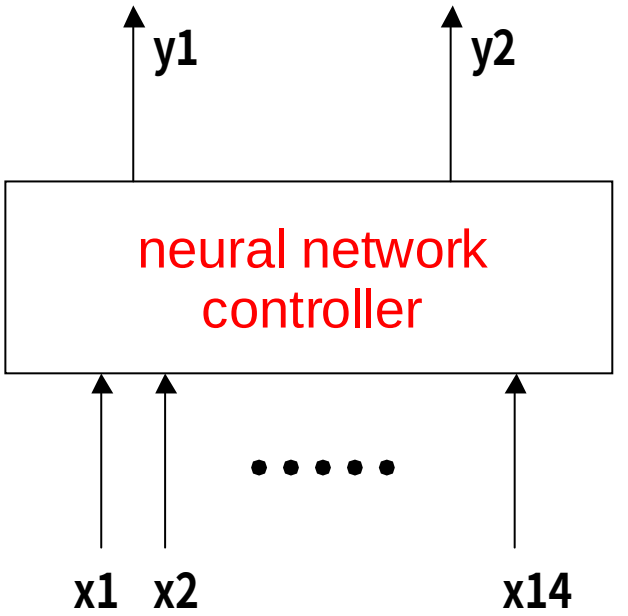




Field of vision of the robot

The neural network controller model

- The inputs are the values in the 14 cells
- The two outputs control the robot
  - Move forward
  - Move backward
  - Turn left
  - Turn right



Evolutionary learning of the controller

- Usually, it is difficult to assign teacher signals to all possible inputs
- Even in this simple example, there are 3^{14} possible inputs
- Evolutionary learning of the controller is necessary
- Genotype can be defined as usual

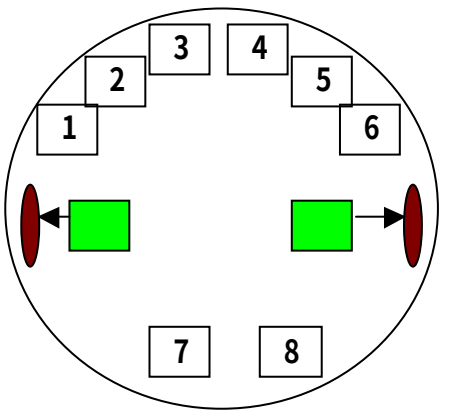
- Fitness can be defined as follows:

$$fitness = a \cdot N_f - b \cdot N_o$$

where N_f is the number of foods the robot gets during a given number of steps, N_o is the number of obstacles the robot runs into on the way, a and b are reward and punish factors, respectively.

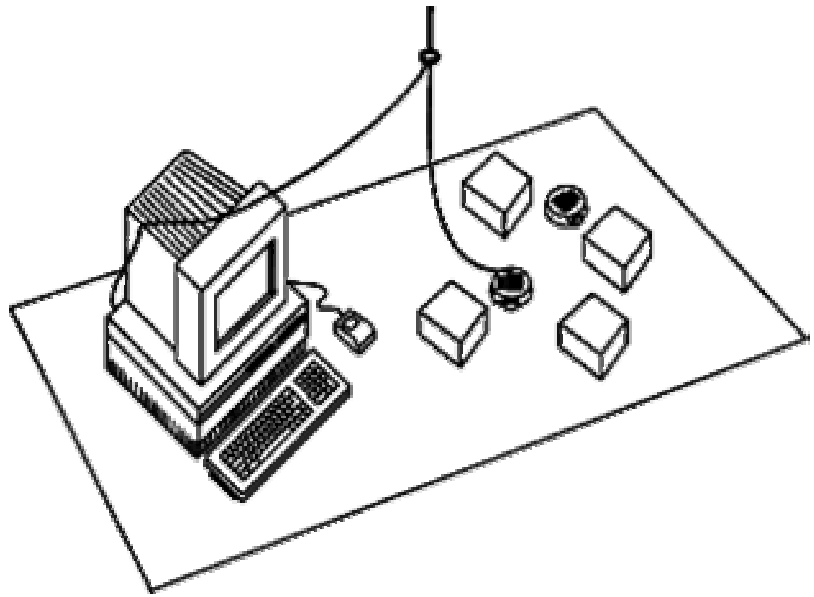
Khepera: a real robot

- Khepera was originally designed as a research and teaching tool for a Swiss Research Priority Program at EPFL in Lausanne
- It allows real world testing of algorithms developed in simulation
  - for trajectory planning
  - obstacle avoidance
  - pre-processing of sensory information



Experiment setup for Khepera

- Khepera can be remote controlled by a host computer through a standard serial port
- The sensor inputs can be transferred to the computer, the computer can make proper decisions and then send instruction based on the inputs
- The decision can be made by using a neural network



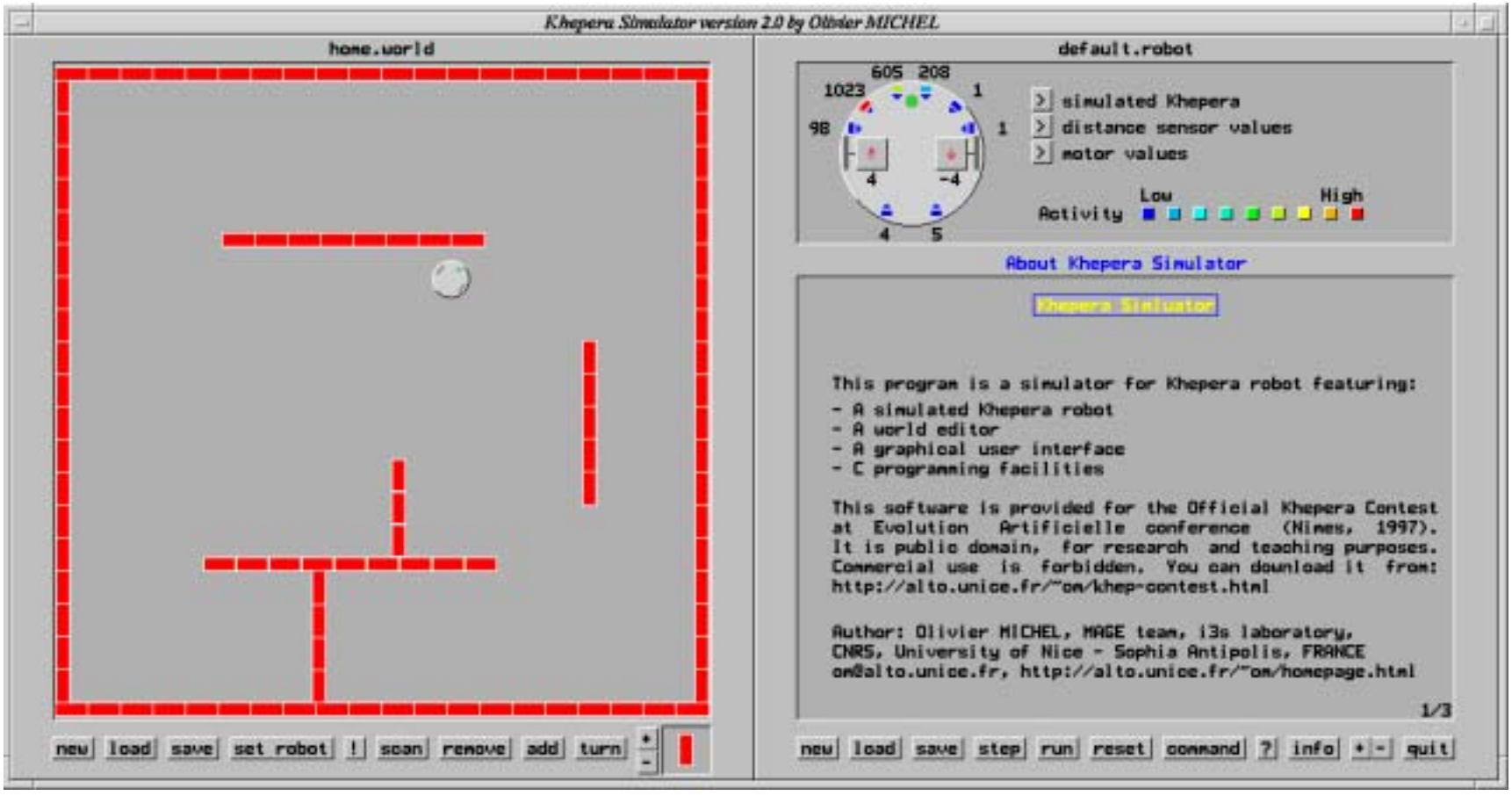
http://www.k-team.com

Problems in using real robots

- Suppose that the population size is 100
- 100 neural networks must be evaluated in each generation
- One controller must be evaluated by running the robot for a large number of steps, if possible, even in different environments
- Thus, real-time evolution of real robots is impossible

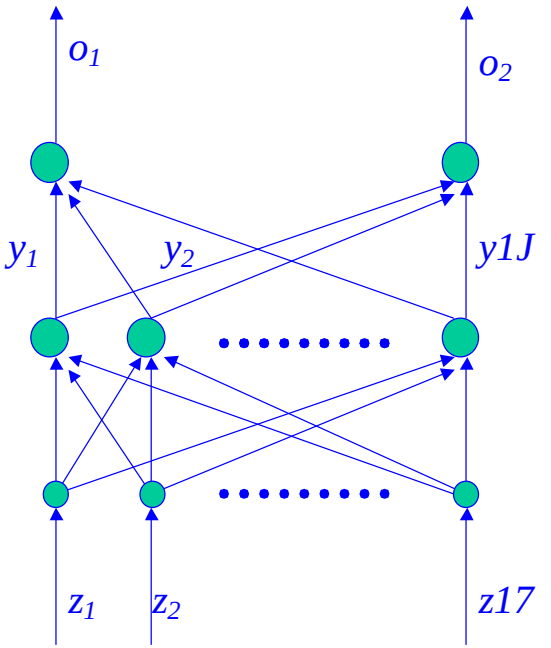
- A possible solution is
  - to use/develop some simulation tools,
  - evolve the robots using the simulator, and
  - test the results using real robots
- For Khepera robot, many free software are available

A 2-D simulation environment



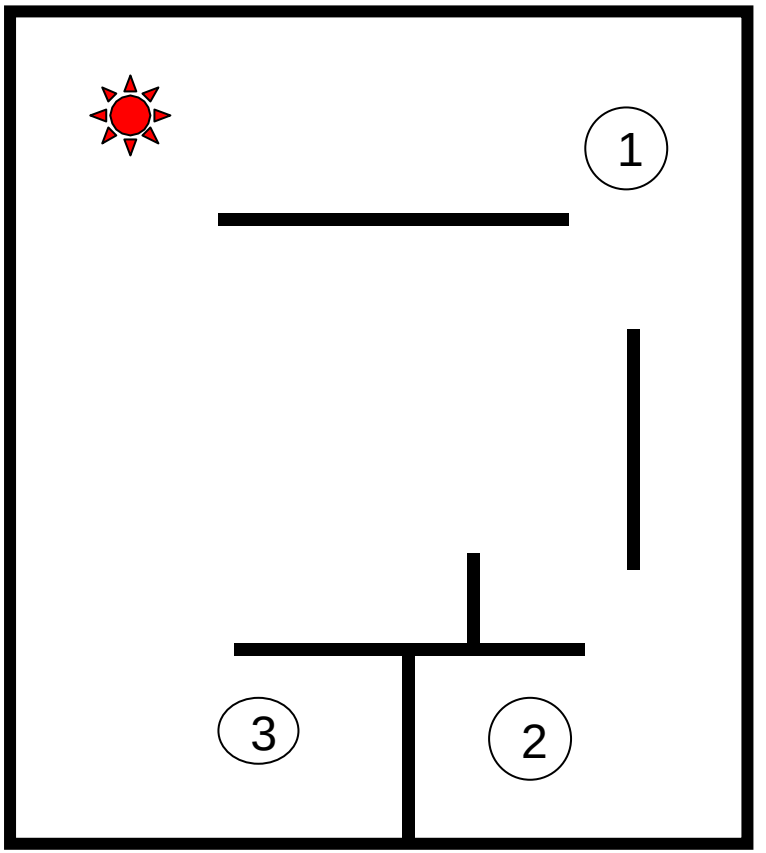
A neural network controller for the base Khepera

- 17 inputs
  - 8 infrared sensors + 8 light sensors + 1 extra input corresponding to the threshold
- Two outputs
  - (1,1) → v1=v2=10
  - (1,0) → v1=0,v2=10
  - (0,1) → v1=10,v2=0
  - (0,0) → v1=v2=-10
- J Hidden neurons
  - J is determined by trials and errors



Basic idea for evolving the neural network controller

- Goal
  - Approach to the light source from any point
- Evaluation
  - To avid lucky guy, we should test the controller several times using different start points
  - For stable evolution, three start points can be selected properly and fixed



Definition of fitness

1) Fitness for the i-th start point

$$\text{fitness}_i = \frac{1}{1+\text{distance}_i} - a_i N_i$$

where distance_i is the distance between the robot and the goal when the robot stops, and a_i is a penalty factor. If the robot reaches to the goal, $\text{distance}_i = 0$, and N_i is the number of steps the robot actually moved. Otherwise, N_i is the pre-specified maximum number of steps for testing the robot.

2) The overall fitness

Since the complixity of task for different start points are different, we can define the overall fitness using the weighted average

$$\text{fitness} = \sum_{i=1}^{3} w_i \, \text{fitness}_i \quad \text{with} \sum_{i=1}^{3} w_i = 1$$

As an example, we can define $w_1 = 0.1$, $w_2 = 0.45$ and $w_3 = 0.35$

Other applications

- The same idea introduced in this lecture can be applied directly to
  - Evolving neural network based automatic driving systems
  - Evolving game robots for
    - Remote controlled mini-car racing
    - Chasing and avoiding robots
  - Evolving intelligent agents

Questions

- What is the inputs of a neural network robot controller ?
- What is the outputs of the neural controller ?
- How to define the genotype of a neural controller ?
- How to define the fitness of a neural controller when it is used to approach a given goal
- How to define the fitness of a neural controller when it is used to follow some moving object ?
- What kind of sensors are necessary for a robot to follow a, say, football ?

Homework

- Write a program for evolving neural network controller, with the same virtual environment and robot defined in the example in p. 6.
- The number of hidden neurons can be given and then fixed during evolution.
- You can also find the number of hidden neurons through evolution
- number of foods: number of obstacles: number of spaces =20:30:50
- The foods and obstacles are put into the environment at random places
- Try to give some idea to avoid lucky guy