PART I

Exercises for the EHW course: evolution of a PAL

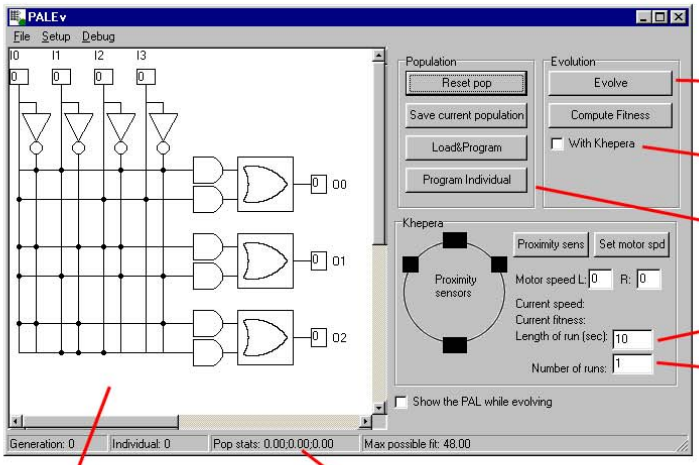
In this exercice you will evolve the configuration of a PAL to implement small logic circuits such as adders and logic gates. You should try to modify the parameter of the PAL and of the genetic algorithm to see what impact this has on the evolvability of the circuits. The objective is to get the feeling of what type of circuits can be evolved and understand why some are easier to evolve than other.

The program that you will use works in two modes: the "truth table" and the "Khepera" mode:

- Truth table mode: in this mode you specify a truth table and the program will evolve the PAL such as to match this truth table. The fitness function is the number of bits that match the entries of the truth table.

- Khepera mode: the truth table isn't used, instead the Khepera robot is controlled and the fitness function aims to avoid obstacles. This mode is used in the mini projects.

In the exercises only the truth table mode is used.



Launches the evolutionary process. In "truth table mode" the fitness function counts the number of bits that match the entries in the truth table. In "Khepera mode" the Khepera robot is controlled and the fitness function aims to avoid obstacles while going forward.

When checked goes in "Khepera mode", otherwise in "truth table mode".

The population box allows you to save populations to disk and restore them later. You can also program a specific individual in the PAL or recreate a new population (reset).

Length of a run in seconds (only in "Khepera mode")

Number of times an individual is tried. In "truth table mode" leave this to 1 as it slows down evolution unnecessarily.

Current PAL configuration. You can click on the inputs to toggle them.

Maximum; average; minimum fitness of the population

Exercise 1. Create a 2-input AND circuit.

Setup the PAL to have 2 inputs, 1 AND line and 1 output. Change the GA parameters to 0% mutation and 0% crossover (i.e. genetic operators are deactivated). By deactivating the genetic operators you simply evaluate n (n=population size) random configurations. Set the population size to 100. Setup the truth table to do an AND:

In 0	In 1	Out 0
0	0	0
0	1	0
1	0	0
1	1	1

Truth table for the 2-input AND

Start the evolution several times and check wether a circuit is found with the maximum fitness. What can you say about the usefulness of the genetic operators in this case?

Exercise 2. Create an n-input AND circuit, varying n

Keep the genetic operators disabled. Set the number of AND lines to 1 and try to create a AND circuit with 3, 4, ... inputs. You should also notice that it is more and more difficult to find a working circuit in the random population when n increases. This is caused by the *increasing size of the search space*!

Exercise 3. Evolve a "constant 0" and a "constant 1" circuit

Activate the genetic operators (~1% mutation, ~70% crossover). Setup the PAL with 4 inputs and 4 outputs. Set the truth table to all "0" and evolve the PAL several times, changing the number of AND lines between 1 and 20. What happens with the average number of generations needed to find a solution? Compare what you obtain with a truth table set to all "1". Are the results similar? Why are some configurations easier to evolve than other, considering that the size of the search space is the same in all the cases?

Exercise 4. Evolve a 1x1 bit adder

Setup the PAL with 2 inputs and 2 outputs. Setup the truth table to do the sum of In0 and In1 with the result on 2 bits: Out[0..1] = In0 + In1 (you can use the file add1_1.tbl).

In 0	In 1	Out 0	Out 1
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Evolve the system with the required minimum number of AND lines and then progressively increase that number. What is the influence on evolvability and why?

Exercise 5. Evolution of a 2x2 bit adder

Try to evolve a 2x2 bit adder (2 inputs on 2 bits, output on 3 bits: Out[0..2] = In[0..1] + In[2..3]). Find parameters such as to evolve the adder in as few generations as possible. You can load the truth table from add2_2.tbl.