Bio-Inspired Artificial Intelligence

Prof. Giovanni Iacca
giovanni.iacca@unitn.it

Week 2 Lab Exercises

Introduction

Goal. The goal of this lab is to continue the investigations of Evolutionary Algorithms (EAs) we started last week. In particular, you will observe the effects of crossover, selection pressure, and population size in artificial evolution, and reflect to what extent these observations also apply to biological evolution.

Getting started. Download the file `02.Exercises.zip` from Moodle and unzip it. This lab continues the use of the *inspyred* framework for the Python programming language seen in the previous lab. If you did not participate in the previous lab, you may want to look that over first and then start this lab's exercises.

Each exercise has a corresponding `.py` file. To solve the exercises, you will have to open, edit, and run these `.py` files.

Note once again that, unless otherwise specified, in this week's exercises we will use real-valued genotypes and that the aim of the algorithms will be to *minimize* the fitness function $f(\mathbf{x})$, i.e. lower values correspond to a better fitness!

Exercise 1

In this exercise we will analyze the effect of crossover in the EA. An offspring individual is formed from two parent individuals $\mathbf{x}_1$ and $\mathbf{x}_2$ by randomly taking the value for each entry x_i either from $\mathbf{x}_1$ or $\mathbf{x}_2$. The EA has a parameter defining the fraction of offspring that is created using crossover at each generation (the remaining individuals are created via asexual reproduction).

To start the experiments, from a command prompt run[1]:

```
$>python exercise_1.py
```

This script executes 30 runs using mutation only (as in the previous exercises), and 30 runs using crossover only. The boxplots compare the best fitness values obtained in the two cases.

- Do you see any difference between the two results? Why?

[1]For all exercises in this lab you may follow the name of the `.py` file with an integer value, which will serve as the seed for the pseudo-random number generator. This will allow you to reproduce your results. Also, please note that in this document `$>` represents your command prompt, do not re-type these symbols.

Exercise 2

In this exercise we will focus on the effect of changing the fraction of offspring created using crossover. Run `exercise_2.py` to compare the best fitnesses obtained by varying this fraction (while using a fixed mutation probability of 0.5, i.e. each loci of each genome will have a 50% chance of being mutated).

- Is there an optimal crossover fraction for this fitness function? Why?

Exercise 3

We will now investigate the effect of the selection pressure. In the previous exercises, we were using tournament selection with a tournament size of 2. Run the `exercise_3.py` file to compare the best fitness values and the distances from the global optimum obtained using tournament sizes 2 and 10.

- Which tournament size gives better results for the fitness function sphere and why?
- Which tournament size is better for the fitness function Rastrigin[2] (you can change the problem by changing the parameter `problem_class` in the script) and why?

Exercise 4

In this exercise you will run the EA on many test functions commonly used to benchmark optimization algorithms. Run the EA on some of the test functions[3] shown in the comments of `exercise_4.py` (especially the multimodal functions) and adapt the mutation magnitude, crossover rate, selection pressure, and population size so as to get the best results. If you run the code as provided it will initialize and bound the values of your population vectors to suitable ranges. You may comment/uncomment certain lines to alter this behavior. See the comments in `exercise_4.py` for further details.

You may first try the 1D or 2D case, which has the advantage that the fitness landscape can be visualized. However, keep in mind that sometimes the resolution of the plot is not sufficient to accurately represent a function.

- Do you see a different algorithmic behavior when you test the EA on different benchmark functions? Why?
- What is the effect of changing the number of variables on each tested function?

Instructions and questions

Concisely note down your observations from the previous exercises (follow the bullet points) and think about the following questions.

[2] `http://pythonhosted.org/inspyred/reference.html#inspyred.benchmarks.Rastrigin`

[3] See `https://pythonhosted.org/inspyred/reference.html#single-objective-benchmarks` for a list of single-objective benchmark problems.

- Why is it useful to introduce crossover in EA? Can you think of any cases when mutation only can work effectively, without crossover? What about using crossover only, without mutation?
- What's the effect of changing the fraction of offspring created by crossover?
- Are there optimal parameters for an EA?
- What are the advantages and disadvantages of low/high selection pressure?