Evolutionary Computation

Notes for the PL 3

Ernesto Costa
Nuno Lourenço
João Macedo

Class PL 3

Experiments with Simple Evolutionary Algorithms (SEA)

3.1 Introduction

We discussed in class the structure of an evolutionary algorithm, that can be easily defined in pseudocode (see algorithm 1).

Algoritmo 1: Simple Evolutionary Algorithm

```
   Input: NumGener, Problem
   Output: Best
1  Population ← RandomPopulation(Problem);
2  Population ← EvalPopulation(Population);
3  foreach gener_i ∈ NumGener do
4  |  Mates ← SelectParents(Problem);
5  |  Offspring ← Variation(Mates);
6  |  Offspring ← EvalPopulation(Offspring);
7  |  Population ← SelectSurvivors(Population, Offspring);
8  return Best(Population);
```

The main issues when applying this model to a concrete problem are **how to**:

- Choose a **representation** for the candidate solutions
- Define a **fitness** function

- Choose the **variation** operators appropriate to the representation and the problem
- Choose the **selection** mechanisms

We are going to explore a problem whose natural representation is a **binary string**. The idea is to consolidate the basic concepts involved in designing an evolutionary algorithm, while at the same time start with the experience of the design of experiments, and the use of visual tools. The problem that we are going to use is the **Numbers of João Brandão**, already discussed in class. A brief review follows. Suppose that we have a set with all integers numbers from 1 to n. We want to find out a subset of maximal size, respecting the constraint that no number in the subset is equal to the average of two other numbers also belonging to the same subset. The code that you are going to need is available at **UCStudent**.

3.2 Exercises

Problema 3.1 F

Use the code provided to solve the Numbers of João Brandão problem. Use the code with different values of the parameters (e.g., number of generations,population size, chromosome size, mutation and crossover probabilities), observe the best individual obtained and **plot** the evolution of the best element of the population over the generations and of the average fitness of the entire population, also over the generations. Analyze the results and draw conclusions.

Problema 3.2 M

Complete the general code code so you can **run the algorithm several times** for the same set of parameters, and return the values of the best over all and the average best over generations. Visualize the results and draw your conclusions.

Problema 3.3 F

Now let's do a another exercise. Adapt the given code so you can store in an **external text file** the fitness of the best individual at the end of the run, one value by line. Then write the code necessary to read and visualize the data.

Problema 3.4 M

The main code provided includes different types of crossover operators. The main goal of this exercise is to discuss which of three crossover operators (one point, two-points and uniform) is the best to solve the JBN problem. Design an experiment that can aid you do decide on this issue, based on the obtained results for each case. Use visualization.