Evolutionary Computation

Notes for the PL 5

Ernesto Costa
João Macedo

Class PL 5

Representations

5.1 General Goal

During the lectures we presented a class of algorithms collectively known as evolutionary algorithms (EA). We stressed the fact that from a problem solving perspective, the representation and the variation operators are key aspects. Other issues, like population initialization, selection of parents and of survivals, and different parameters (e.g., variation's operators probabilities) are equally important aspects.

Our goal for today's Lab is to do some experiments with EAs in the case of a representation based on **floats** and on **permutations** we just discussed. The idea is to consolidate the general aspects of an EA (e.g., running, collecting data, visualization, interpretation of the results), while at the same time see the specific aspects of the algorithms when we change the representation (e.g., particular variation operators, population initialization).

In the next sections we will briefly describe the problems, and state what you are suppose to do. In **UCStudent** we provide some code and data that you are going to need. Be aware that the code is provided without any warranty, yet we tried to do our best. So be cautious and analise carefully the code. If you find any problem, please report to the professors.

5.2 Benchmark Problems: the case of floats

In order to play with the different algorithms, we introduce some standard testbed problems. The ones presented here may be complemented by you.

For each case, we give a brief description and display the function in the case of two variables. In your experiments you should test the algorithms with a greater number of dimensions (e.g., 10).

Quartic Also known as De Jong's F4. Continuous, convex, unimodal (see figure 5.1).

$$Min(F_4) = F_4(0, \ldots, 0) = 0$$

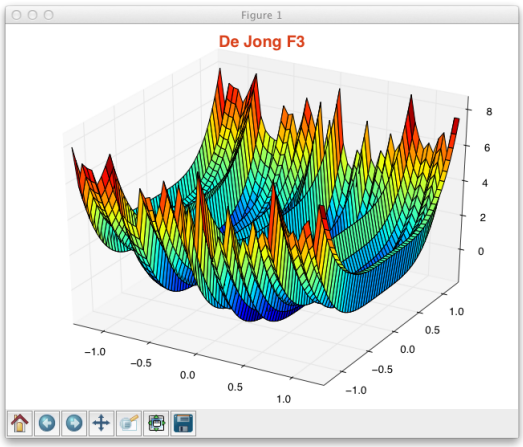

Figura 5.1: **De Jong F4**

It is defined by the equation 5.1.

$$f(\langle x_1, \ldots, x_n \rangle) = (\sum_{i=1}^{n} i \times x_i^4) + \mathcal{N}(0,1) \qquad x_i \in [-1.28, 1.28] \tag{5.1}$$

with $\mathcal{N}(0,1)$ a gaussian noise of mean 0 and standard deviation 1.

Rastrigin Highly multimodal with many local minima (see figure 5.2 for a 2D version).

$$Min(Rastrigin) = Rastrigin(0, \ldots, 0) = 0$$

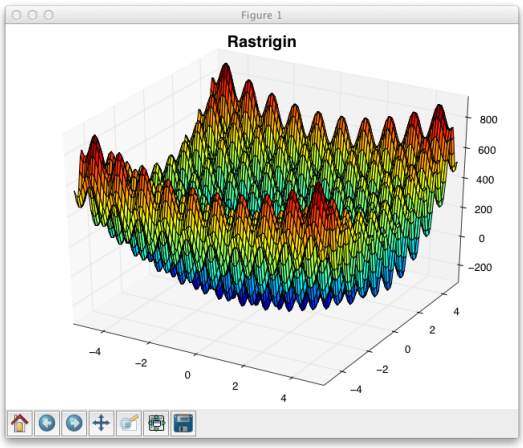

Figura 5.2: **Rastrigin**

It is defined by the equation 5.2.

$$f(\langle x_1, \ldots, x_n \rangle) = A \times n + (\sum_{i=1}^{n} x_i^2 - A \times cos(2\pi x_i) \qquad x_i \in [-5.12, 5.12] \quad (5.2)$$

with A typically equal to 10.

Schwefel Multimodal with many local minima (see figure 5.3 for a 2D version).

$$Min(Schwefel) = Schwefel(420.9687, \ldots, 420.9687) = n \times 418.9829$$

It is defined by the equation 5.3.

$$f(\langle x_1, \ldots, x_n \rangle) = \sum_{i=1}^{n} -x_i \times sin(\sqrt{|x_i|}) \qquad x_i \in [-500, 500] \qquad (5.3)$$

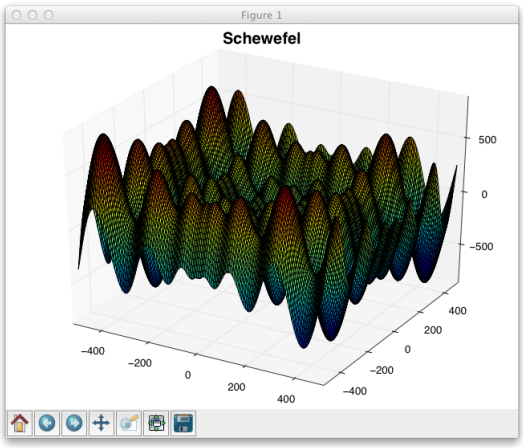

Figura 5.3: **Schwefel**

5.3 Benchmark Problem: the case of permutations

Here we present again the TSP as it was discussed in class.

TSP The Traveling Salesman Problem involves a salesman that has to visit a certain number of cities starting from a particular one, visiting the others only one time and returning to the initial one. This is called a tour. We want the tour of minimum cost (e.g., minimum number of kilometers.).

You are going to need a file with the description of a concrete example. We provide a specific example in **UCStudent**. Notice that these files have a specific **format** as it is illustrated in the figure 5.4.

For the purpose of the exercise, the only relevant part are the 2D coordinates of each city. So there is some code whose main goal is to extract from this type of files the coordinates of the cities. Then, the list of coordinates is kept into a dictionary, where the key is an integer that identifies the city and the value is a pair with the coordinates.

```
NAME: berlin52
TYPE: TSP
COMMENT: 52 locations in Berlin (Groetschel)
DIMENSION: 52
EDGE_WEIGHT_TYPE: EUC_2D
NODE_COORD_SECTION
1 565.0 575.0
2 25.0 185.0
3 345.0 750.0
4 945.0 685.0
...
EOF
```

Figura 5.4: Example of the format of a tsp file. You may have more than one COMMENT line.

5.4 Let's do it

Problema 5.1

For each class of problems (multimodal optimization and TSP), the idea is to experiment the algorithms with different parameters (population size, number of generations, probabilities of the variation operators) and see which one give the best results. You should analyze the performance of the result (best over generations, average best), but also pay attention the the number of evaluations needed to attain the best final result.

Problema 5.2

For the optimization of functions whose individuals are represented by a vector of floats, we presented in class different variation operators, but not all were implemented. So, now your task is to complete that job. Implement the **uniform mutation**. This operator mutates an individual according with a certain probability. For an individual $\mathbf{x} = \langle x_1, \ldots, x_k, \ldots, x_n \rangle$, the operator selects a random component, say k, and produces a new individual $\mathbf{x} = \langle x_1, \ldots, x'_k, \ldots, x_n \rangle$, where x'_k is a random value in the range $[inf\_k, sup\_k]$, obtained from an uniform probability distribution. Implement also the **heuristic crossover** operator. This operator takes two individuals, say x_1 and x_2 and produces a new one x_3 according with the rule

$$x_3 = a \times (x_2 - x_1) + x_2, \text{ with } a \in [0, 1]$$

and x_2 not worse that x_1. Notice that only **one** offspring is produced.

Do some empirical comparison tests, and see if you can draw any conclusions.

Problema 5.3

For the TSP whose individuals are represented by a permutation of integers, we presented in class different variation operators, but not all were implemented. So, now your task is to complete that job. Implement the **scramble mutation** operator and the **cycle crossover** operator. The mutation operator select randomly two points in the chromosome and scrambles all the components between these two points. The cycle crossover builds offspring in such a way that each component (and its positions) comes from one of the parents. It is a kind of uniform crossover but for permutations. To achieve the result we have to compute the cycles defined by the two individuals and then build two offspring from those cycles, as it was explained in class.

Do some empirical comparison tests, and see if you can draw any conclusions.