Université de Genève

Metaheuristics for optimization

TP 6: Genetic Algorithms and Function Minimization

Author: Joao Filipe Costa da Quinta

E-mail: Joao.Costa@etu.unige.ch

December 13, 2021

UNIVERSITÉ DE GENÈVE

FACULTÉ DES SCIENCES
Département d'informatique

1 Introduction

During this TP we will be working with an algorithm that tries to simulate genetic evolution. In biology each the entire population adapts to an environment through the generations by crossing different individuals, and mutations. This is what we will try to do in this algorithm.

2 Function minimisation

The goal is to use genetic algorithm to minimize a given function:

$$f(x,y) = -\left|\frac{1}{2}x\sin\left(\sqrt{|x|}\right)\right| - \left|y\sin\left(30\sqrt{\left|\frac{x}{y}\right|}\right)\right|$$

This function takes 2 parameters, x and y, both these parameters will be $\in [10, 1000] \cap R$.

3 Implementation of the algorithm

Both x and y values will be coded in binary form, and then both values will be concatenated. That is the 'genetic' code of an individual. For example, $x = 0101010101$ and $y = 1010101010$, this means that this individuals genetic code would be 01010101011010101010.
There are three important steps in this algorithm, selection, crossing and mutation. Let's see how each step is done, all these steps will be done at each generation.

3.1 Selection

The goal is to choose the best individuals to 'live on', and the worst ones we let them 'die'. There are many ways of doing this. The way we choose to do this in this TP is the k-Tournament, this is a really easy step to selection method to understand. From the whole population at generation t, we chose 5 individuals at random (all have the same probability of being chosen), from the individuals, we keep the one with the best fitness. We do this N times, so that the number of individuals is constant through the generations.

3.2 Crossing

Just like in nature, we will make it so that an individual is a mix of individuals of the previous generation. This step is easily done thanks to how our individuals genetics code.
Lets assume we have 2 individuals:

$$individual_1 = 01010101011010101010$$
$$individual_2 = 11111111110000000000$$

Their corresponding children would be:

$$child_1 = 01010111111010100000$$
$$child_2 = 11111101010000001010$$

However, this is only done with a given probability, P_c. So there is a probability P_c that two individuals are crossed, and a probability $1 - P_c$ that two individuals stay the same.

3.3 Mutation

This is the easiest step, basically for every bit in the population, there is a P_m that it changes from 0 to 1, or 1 to 0. However usually this probability is very small.

3.4 Algorithm

(1) Generate random population of size N

(2) For $generation\_max$ do the following steps

(3) Select Individuals, as show in section 3.1

(4) Compute crossing, as show in section 3.2

(5) Do mutations, as show in section 3.3

(6) After doing this for $generation\_max$, return best fit individual.

4 Results

The general algorithm works, but I think there might be some errors, I say this because my results aren't what I expected.

For our tests: N=100, $P_c = 0.6$, $P_m = \{0.1, 0.01\}$, $max\_generation = \{10,100,1000\}$

Mean results for 10 executions

max_generation	with crossing		without crossing	
-	P_m = 0.1	P_m = 0.01	P_m = 0.1	P_m = 0.01
10	-1282	-1275	-1283	-1274
100	-1284	-1283	-1285	-1285
1000	-1284	-1261	-1283	-1285

My best result was with $max\_generation = 1000$, $P_m = 0.01$.

The reason why I think there are some mistakes in my code, is because the parameters don't seem to influence much the minimum.