Instituto Politécnico Nacional

Escuela Superior de Cómputo

Evolutionary Computing

3CM1

3 Genetic Algorithm (knapsack, coin change, jugs)

Santiago Nieves Edgar Augusto

Teacher: Jorge Luis Rosas Trigueros

Date: 30/08/2016

Deadline: 30/08/2016

Theoretical framework:

Genetic algorithms search the solution space of a function through the use of simulated evolution ie the survival of the test strategy In general the ttest individuals of any population tend to reproduce and survive to the next generation thus improving successive generations, genetic algorithm is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms. Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as mutation, crossover and selection.

In a genetic algorithm a population of candidate solutions to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible.

The algorithm requires:

- A genetic representation of the solution domain.
- A fitness function to evaluate the solution domain.

Genetic algorithm has some parts:

- Heredity: There must be a process in place by which children receive the properties of their parents. If creatures live long enough to reproduce, then their traits are passed down to their children in the next generation of creatures.
- Variation: There must be a variety of traits present in the population or a means with which to introduce variation. For example, let’s say there is a population of beetles in which all the beetles are exactly the same: same color, same size, same wingspan, same everything. Without any variety in the population, the children will always be identical to the parents and to each other. New combinations of traits can never occur and nothing can evolve.
- Selection: There must be a mechanism by which some members of a population have the opportunity to be parents and pass down their genetic information and some do not. This is typically referred to as “survival of the fittest.” For example, let’s say a population of gazelles is chased by lions every day. The faster gazelles are more likely to escape the

lions and are therefore more likely to live longer and have a chance to reproduce and pass their genes down to their children. The term fittest, however, can be a bit misleading.

Material and equipment:

Software:

Sublime Text.

Python 2.7.6.

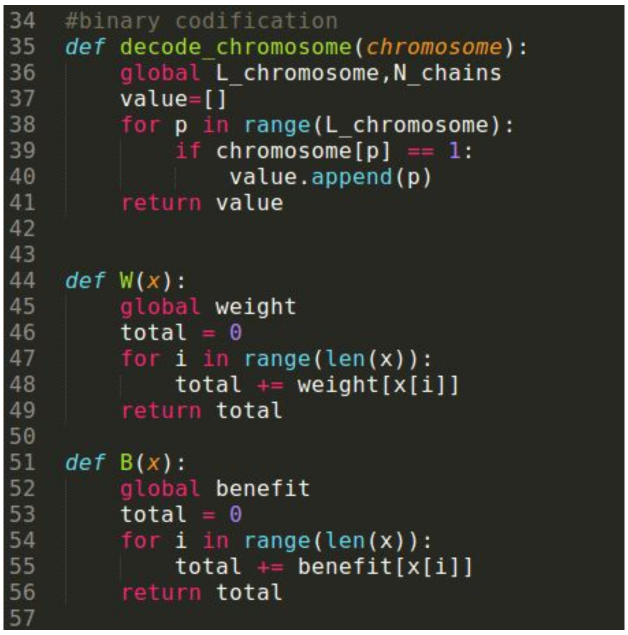
Ubuntu 16.04 LTS.

Hardware:

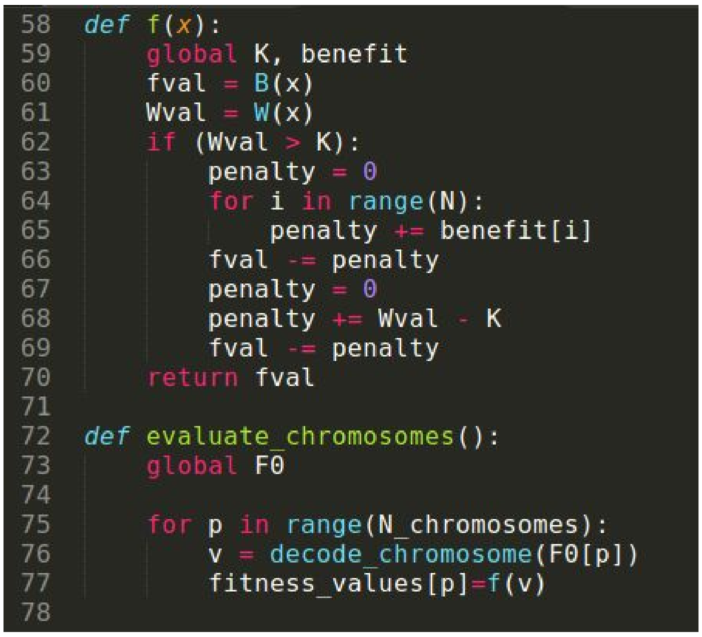
Procesador: Intel® Core™ i3 CPU M 380 @ 2.1GHz × 4.

Body:

Given the problem knapsack, given N items, each with its own value V i and weight W i , con 1 <= i <= N, and a maximum knapsack size S, compute the maximum value of the items that one can carry, if he can either ignore or take a particular item but in this problem you need to implement that problem with algorithm genetic.



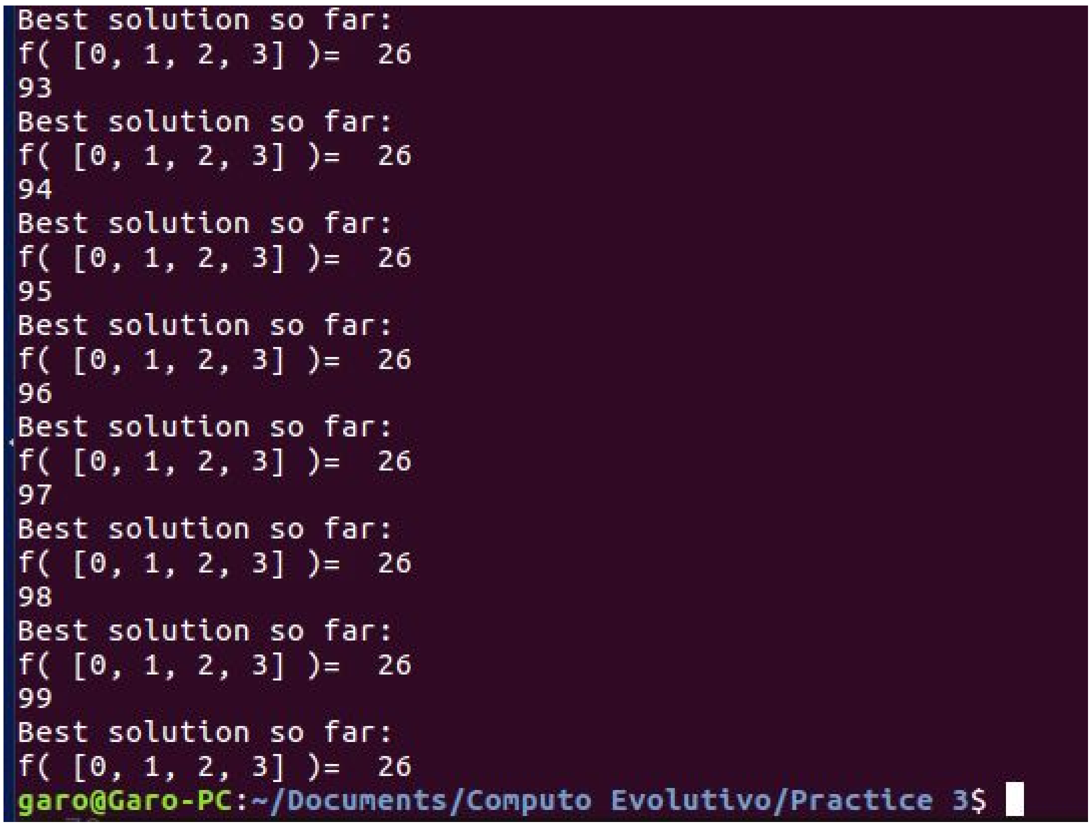
```python
#binary codification
def decode_chromosome(chromosome):
    global L_chromosome,N_chains
    value=[]
    for p in range(L_chromosome):
        if chromosome[p] == 1:
            value.append(p)
    return value


def W(x):
    global weight
    total = 0
    for i in range(len(x)):
        total += weight[x[i]]
    return total

def B(x):
    global benefit
    total = 0
    for i in range(len(x)):
        total += benefit[x[i]]
    return total

```



```python
def f(x):
    global K, benefit
    fval = B(x)
    Wval = W(x)
    if (Wval > K):
        penalty = 0
        for i in range(N):
            penalty += benefit[i]
        fval -= penalty
        penalty = 0
        penalty += Wval - K
        fval -= penalty
    return fval

def evaluate_chromosomes():
    global F0

    for p in range(N_chromosomes):
        v = decode_chromosome(F0[p])
        fitness_values[p]=f(v)

```

Figure 1. Code of the knapsack problem.

Figure 2. Code of the knapsack problem.

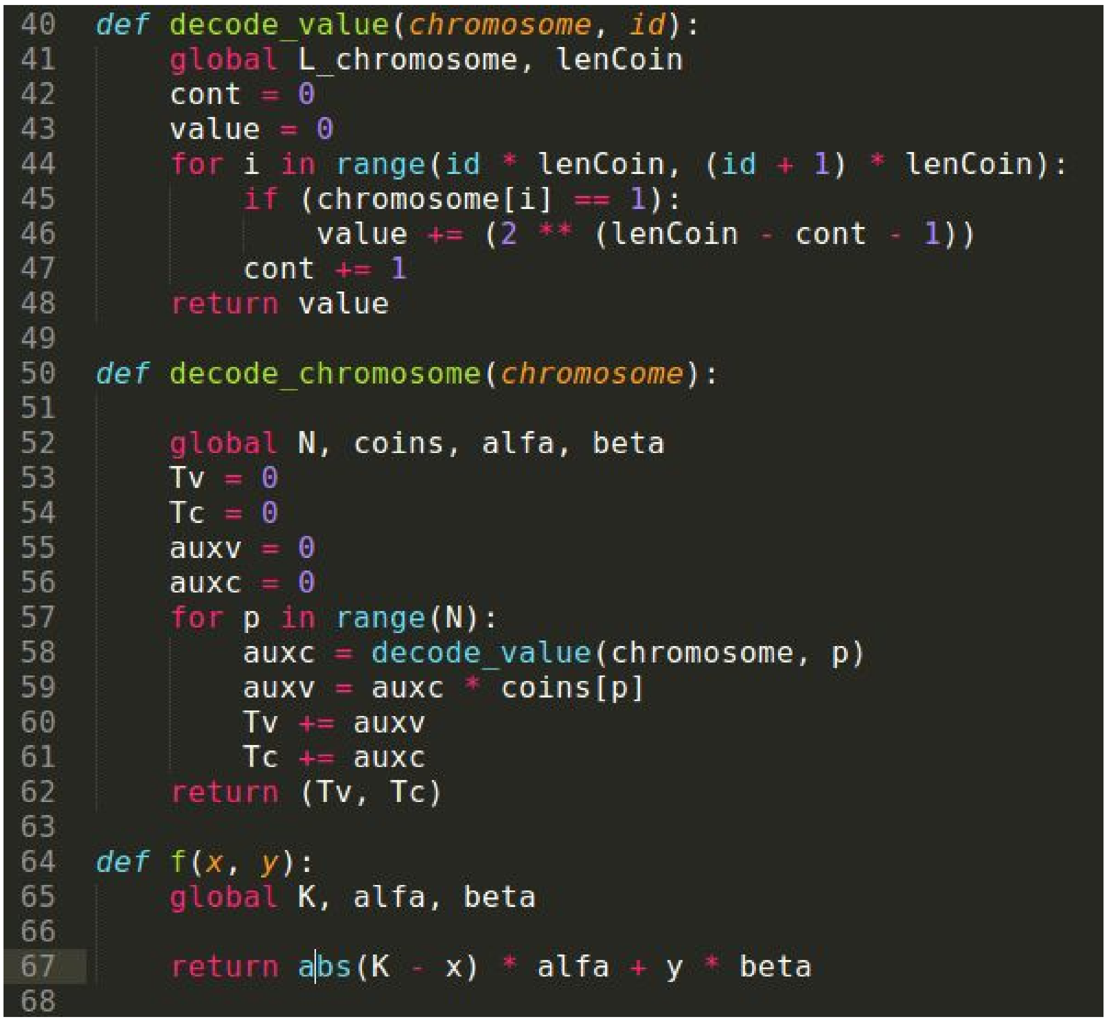


```
Best solution so far:
f( [0, 1, 2, 3] )=  26
93
Best solution so far:
f( [0, 1, 2, 3] )=  26
94
Best solution so far:
f( [0, 1, 2, 3] )=  26
95
Best solution so far:
f( [0, 1, 2, 3] )=  26
96
Best solution so far:
f( [0, 1, 2, 3] )=  26
97
Best solution so far:
f( [0, 1, 2, 3] )=  26
98
Best solution so far:
f( [0, 1, 2, 3] )=  26
99
Best solution so far:
f( [0, 1, 2, 3] )=  26
garo@Garo-PC:~/Documents/Computo Evolutivo/Practice 3$
```

Figure 3. To test the problem with one example.

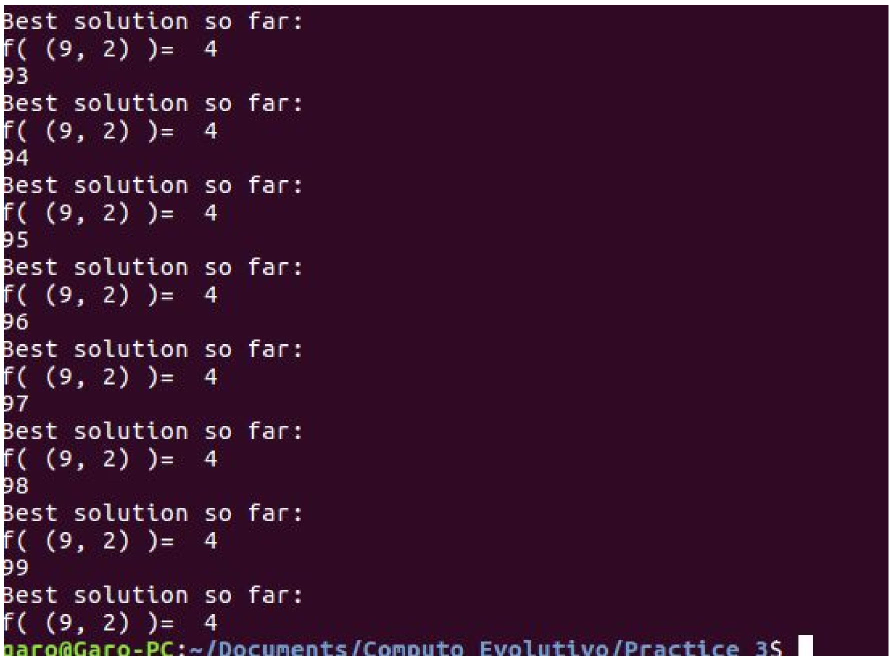
Given the problem coin change, given a value N, if we want to make change for N cents, and we have infinite supply of each of S = { S1, S2, .. , Sm} valued coins, What is the minimum number of coins to have N? The order of coins doesn't matter.

For example, for N = 4 and S = {1,2,3}, there are 2 solutions: {2,2},{1,3}. So output should be 2. For N = 10 and S = {2, 5, 3, 6}, solutions: {5,5}. So the output should be 2. The same situation in this problem, you need to implement this problem but now with algorithm genetic.



```python
def decode_value(chromosome, id):
    global L_chromosome, lenCoin
    cont = 0
    value = 0
    for i in range(id * lenCoin, (id + 1) * lenCoin):
        if (chromosome[i] == 1):
            value += (2 ** (lenCoin - cont - 1))
        cont += 1
    return value

def decode_chromosome(chromosome):

    global N, coins, alfa, beta
    Tv = 0
    Tc = 0
    auxv = 0
    auxc = 0
    for p in range(N):
        auxc = decode_value(chromosome, p)
        auxv = auxc * coins[p]
        Tv += auxv
        Tc += auxc
    return (Tv, Tc)

def f(x, y):
    global K, alfa, beta

    return abs(K - x) * alfa + y * beta

```

Figure 4. Program for coin change the functions most important.



```
Best solution so far:
f( (9, 2) )=  4
93
Best solution so far:
f( (9, 2) )=  4
94
Best solution so far:
f( (9, 2) )=  4
95
Best solution so far:
f( (9, 2) )=  4
96
Best solution so far:
f( (9, 2) )=  4
97
Best solution so far:
f( (9, 2) )=  4
98
Best solution so far:
f( (9, 2) )=  4
99
Best solution so far:
f( (9, 2) )=  4
garo@Garo-PC:~/Documents/Computo Evolutivo/Practice 3$
```

Figure 5. Test for the program coin change with genetic.

Given the problem jugs, given 3 jags the first with 3 liters, the second with 5 liters and the third with 8 liters you need to say the minimum operations to have a jug that it needs to have 4 liters, the operations are: you can fill or empty a jug, you can pass the content of one jug to another jug until the jug will be empty or the another jag will be full.

Conclusions:

In this practice I can learn about Genetic Algorithm, when the teacher teached about it I didn't catch the idea but when I did this practice I could step the topic up, in the beginning of the practice I didn't understand the idea of this topic but now we need to solve the first problem that we saw in the beginning of the course but now with Genetic Algorithm in my case this new topic in this problem is some different but I can practice this topic better with these kind of problems that they are different that the previous problem with Genetic Algorithm.

Bibliography

A Genetic Algorithm for Function Optimization A Matlab Implementation Christopher R Houck North Carolina State University and Jeery A Joines North Carolina State University and Michael G Kay North Carolina State University Age

https://en.wikipedia.org/wiki/Genetic_algorithm