Universidade Federal de Santa Catarina

Centro Tecnológico

Departamento de Automação e Sistemas

Exercício 1: Algoritmo Genético

Inteligência Artificial

Juliano Ricardo da Silva
Prof. Dr. Eric Antonelo

Florianópolis - SC

Fevereiro de 2022

Conteúdo

1 Tarefas

1.a Questão 1

O algoritmo genético programado pode ser conferido abaixo:

```python
#Algoritmo genetico

L = 4 * 8 #size of chromossome in bits

import struct
import random
import math
from statistics import mean
from statistics import median
import matplotlib.pyplot as plot
import numpy as np

def floatToBits(f):
        s = struct.pack('>f', f)
        return struct.unpack('>L', s)[0]

def bitsToFloat(b):
        s = struct.pack('>L', b)
        return struct.unpack('>f', s)[0]

#exemplo 1.23 -> '0010111100'

def get_bits(x):
        x = floatToBits(x)
        N = 4 * 8
        bits = ''
        for bit in range(N):
                b = x & (2**bit)
                bits += '1' if b > 0 else '0'
        return bits

#exemplo '00010111100' -> 1.23

def get_float(bits):
        x = 0
        assert(len(bits) == L)
        for i, bit in enumerate(bits):
                bit = int(bit)  #0 or 1
                x += bit * (2**i)
        return bitsToFloat(x)
```

```
#size of population
global n

def pop_len():
        p = int(input("digit the number of population:  "))
        if(p%2 != 0):
                print("odd number")
                p = p - 1
                print("changed to even")
                return p
        else:
                return p

#generates a list of people (chromossomes)
def population():
        for i in range(g):
                p = random.SystemRandom().uniform(0, math.pi)
                if p == math.pi:
                        p = round(p)
                else:
                        print(p)
                person = get_bits(p)
                print(person)
                people.append(person)

#fitness calculation for each chromossome
def calc_fitness():
        for t in range(len(people)):
                people[t] = get_float(people[t])
                fitness   = people[t] + abs(math.sin(32*people[t]))
                fitnessList.append(fitness)

#probab weights for selection
def calc_weights():
        for w in range(len(fitnessList)):
                weights = fitnessList[w]/ (sum(fitnessList)/len(fitnessList)
                   )
                weightsList.append(weights)
```

```
#couple's selection
def roullette_selection():
        s = random.choices(people, weightsList, k = 2)
        for i in range(2):
                c = get_bits(s[i])
                couple.append(c)

#single point crossover
def crossover():
        pc = random.randint(1,10)/10
        if   0.1 <= pc <= 0.7:
                print('cruzamento resultou em: \n')
                d1 = dad[0:16]+mom[16:32]
                d2 = mom[0:16]+dad[16:32]
        else:
                print("copia identica \n")
                d1 = dad[:]
                d2 = mom[:]
        descendants.append(d1)
        descendants.append(d2)
        print(descendants)

#mutation
def mutation():
    pm = random.randint(1,10)/10
    if  0.0001 <= pm <= 0.1:
        sd = descendants[random.randint(0,1)]
        if sd == descendants[0]:
            new_population.append(descendants[1])
        else:
            new_population.append(descendants[0])
            ap = random.randint(0,32)
            if sd[ap] == '0':
                m = '1'
            else:
                m = '0'
                md = sd[:ap] + m + sd[ap+1:]
                new_population.append(md)
                print("selected descendant:",sd)
                print("position:",ap)
                print("the mutated chromossome is: \n",md)
    else:
        new_population.append(descendants[0])
        new_population.append(descendants[1])
        print("doesn't occurred a mutation")
```

```python

#fitness calculation for each chromossome
def calc_new_fitness():
        for t in range(len(chrome)):
                chrome[t] = get_float(chrome[t])
                fitness   = chrome[t] + abs(math.sin(32*chrome[t]))
                new_fitnessList.append(fitness)
                print("List of fitness: ", new_fitnessList)
        return new_fitnessList

chrome = []
n = pop_len()
g = n
hist = []
new_fitnessList = []
avg_fit = []
novalista =  []
alist = []
nl = int(2*g)

while True:
    while(not(len(chrome) == g)):
        print("********************************************")
        print("*****************NEW ITERATION**************")
        print("********************************************")
        people = []
        population()   #population
        print("Population of chromossomes:",people)
        fitnessList = []
        calc_fitness() #calculates the fitness for each chromossome
        print("List of fitness: ", fitnessList)
        weightsList = []
        calc_weights() #calculates the probab weights for each chromossome
        print("Prob weights:",weightsList)
        couple = []
        roullette_selection() #selects a couple of chromossomes
        print("The couple selected: ", couple)
        dad = couple[0]
        mom = couple[1]
        descendants = []
        crossover() #chromossomes crossover to generate descendants
        new_population = []
        mutation()
        for i in new_population:
            chrome.append(i)
```

```
            print("New pop is:  ", chrome) #[-g:]
        calc_new_fitness()
        h = int(2)
        alist = list(np.average(np.reshape(new_fitnessList, (-1, h)), axis=1))
        print("Average fitness list: ", alist)

        break

print("done!")
print("population length: ", len(chrome))


x_values = list(range(0,g,2))
y_values = [f for f in alist]
plot.plot(x_values, y_values)
plot.show()
```

1.b Questão 2

As figuras abaixo mostram diferentes resultados obtidos para diferentes tamanhos de população:

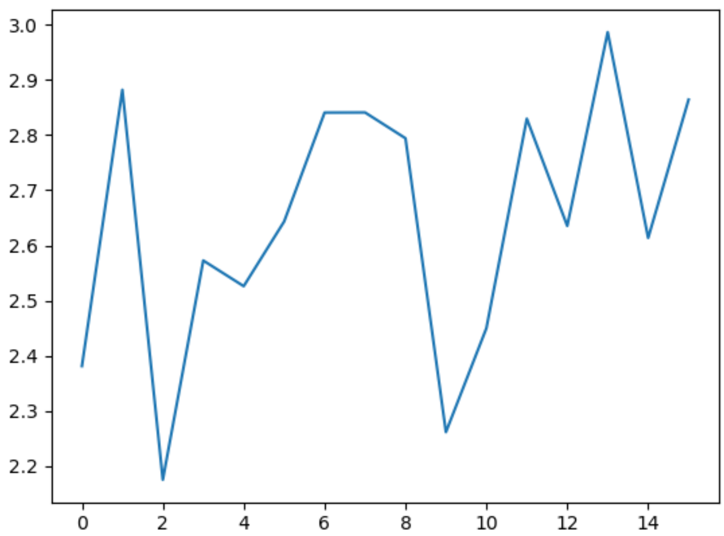

Figura 1: Para uma população n = 15.

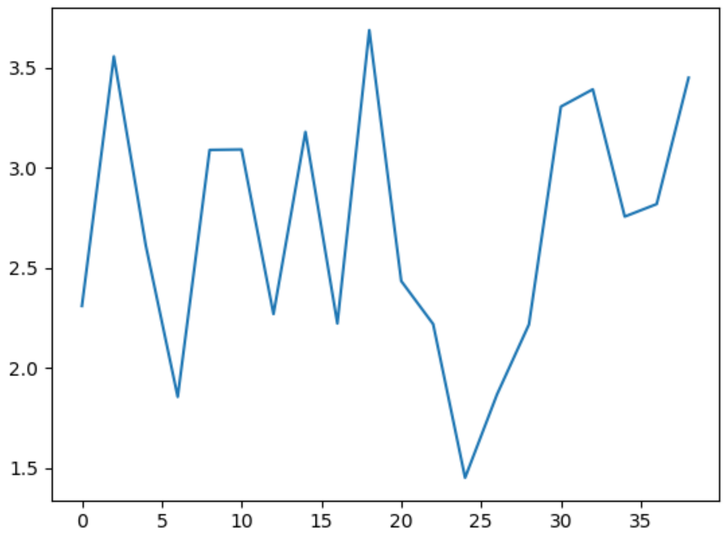

Figura 2: Para uma população n = 40.

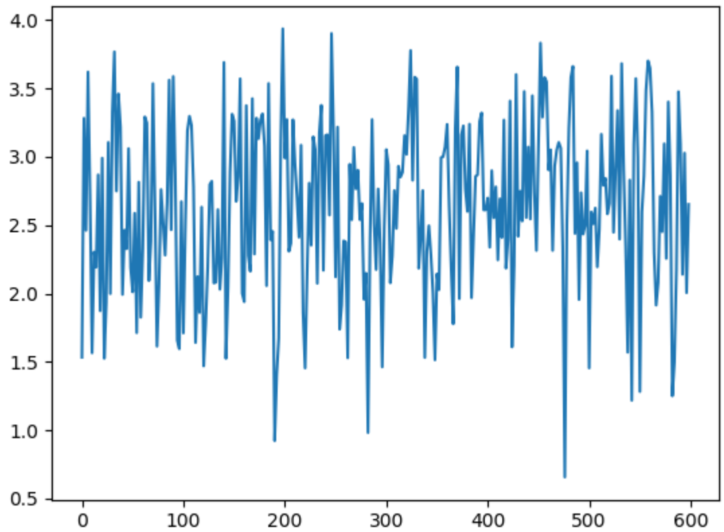

Figura 3: Para uma população n = 600.

Modificando a taxa de mutação e cruzamento:

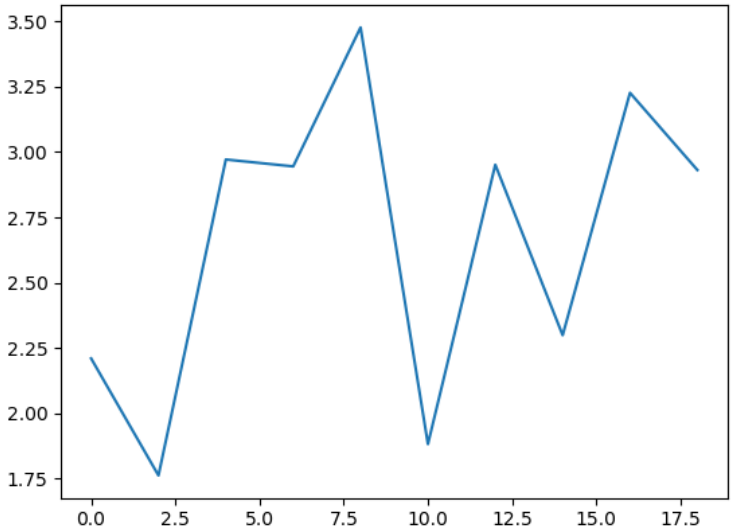

Figura 4: Para uma população n = 20, taxa de cruzamento e mutação maiores.

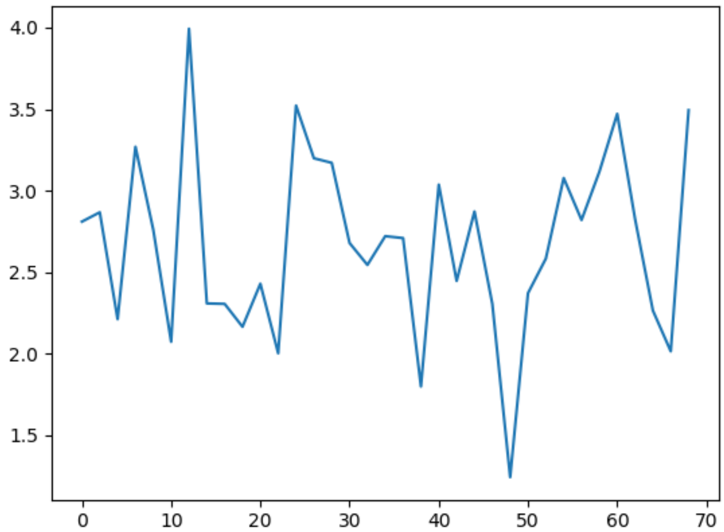

Figura 5: Para uma população n = 70, taxa de cruzamento e mutação menores.

1.c Questão 3

Os gráficos de aptidão média são mostrados abaixo:

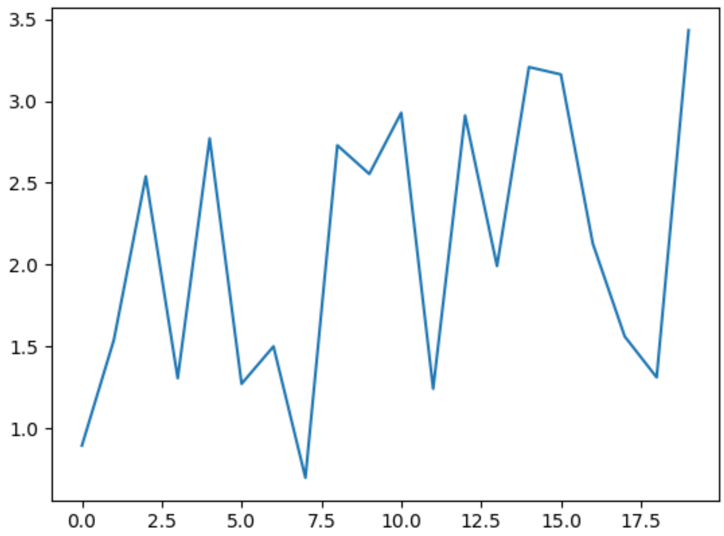

Figura 6: Gráfico de aptidão média para população de n = 20.

1.d Questão 4

Finalmente, a comparação das aptidões:

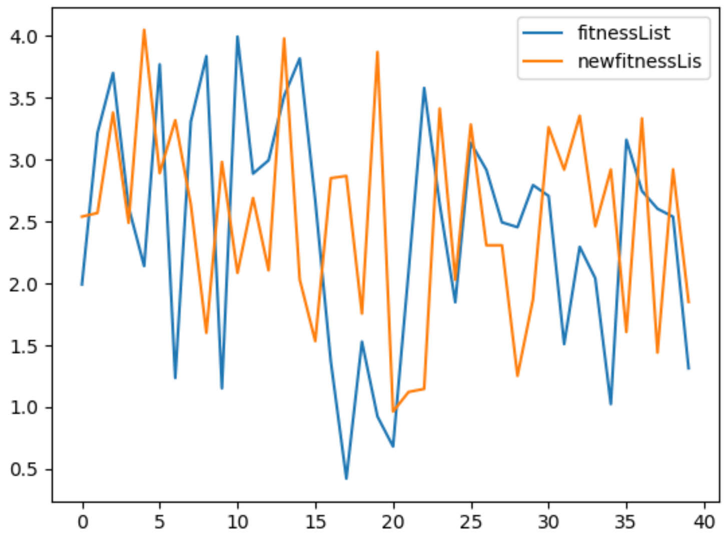

Figura 7: Comparando a aptidão obtida com a população inicial com a obtida após o algoritmo genético, n = 40.

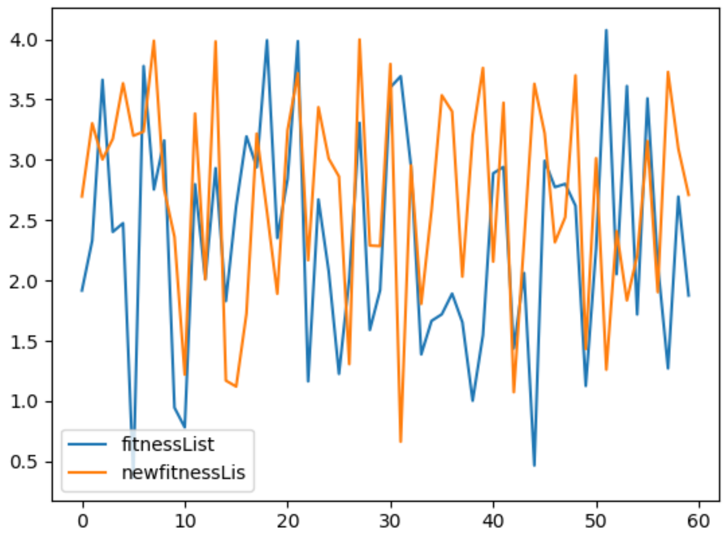

Figura 8: Comparando a aptidão obtida com a população inicial com a obtida após o algoritmo genético, n = 60.

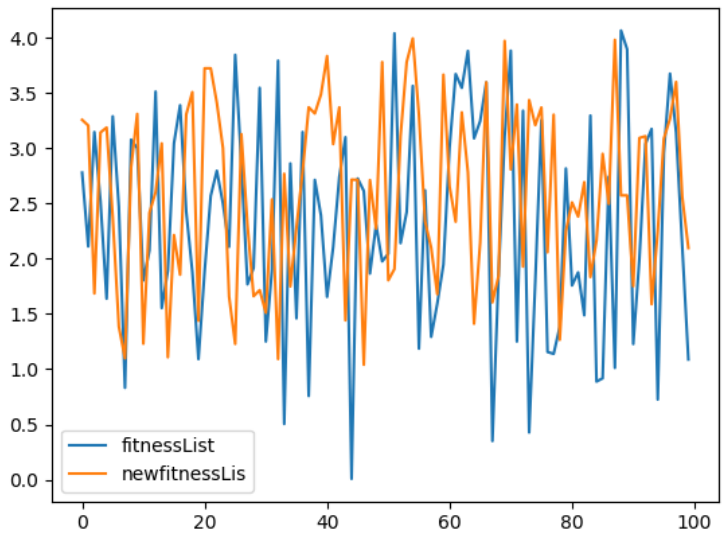

Figura 9: Comparando a aptidão obtida com a população inicial com a obtida após o algoritmo genético, n = 100.

Isso mostra que, gradativamente, há um aumento no valor médio da aptidão após as operações genéticas ocorridas ao decorrer das gerações.