Roshan Shetty 525 DML

Assignment

Aim : Genetic Algorithm

Code :

```
import numpy as np
import pandas as pd
import os
import random

#initialize population
best = -100000
populations =([[random.randint(0,1) for x in range(6)] for i in range(4)])
print(type(populations))
parents=[]
new_populations = []
print(populations)

#fitness score calculation .....
def fitness_score() :
    global populations,best
    fit_value = []
    fit_score=[]
    for i in range(4) :
        chromosome_value=0

        for j in range(5,0,-1) :
            chromosome_value += populations[i][j]*(2**(5-j))
        chromosome_value = -1*chromosome_value if populations[i][0]==1 else chromosome_value
        print(chromosome_value)
        fit_value.append(-(chromosome_value**2) + 5 )
    print(fit_value)
    fit_value, populations = zip(*sorted(zip(fit_value, populations) , reverse = True))
    best= fit_value[0]

#print(type(populations))
#selecting parents....
def selectparent():
    global parents
    #global populations , parents
    parents=populations[0:2]
    print(type(parents))
    print(parents)

#single-point crossover .....

def crossover() :
    global parents
    cross_point = random.randint(0,5)
```

```
    parents=parents + tuple([(parents[0][0:cross_point +1] +parents[1][cross_point+1:6])])
    parents =parents+ tuple([(parents[1][0:cross_point +1] +parents[0][cross_point+1:6])])
    print(parents)

def mutation() :
    global populations, parents
    mute = random.randint(0,49)
    if mute == 20 :
        x=random.randint(0,3)
        y = random.randint(0,5)
        parents[x][y] = 1-parents[x][y]
    populations = parents
    print(populations)

#fitness_score()
#selectparent()
#crossover()
#mutation()
for i in range(5) :
    fitness_score()
    selectparent()
    crossover()
    mutation()
print("best score :")
print(best)
print("sequence.....")
print(populations[0])
```

Roshan Shetty 525 DML

Output:

```
└──╼ $python main.py
<class 'list'>
[[1, 1, 1, 0, 1, 1], [0, 0, 0, 1, 1, 0], [1, 0, 1, 1, 0, 0], [1, 1, 0, 1, 1, 0]]
-27
6
-12
-22
[-724, -31, -139, -479]
<class 'tuple'>
([0, 0, 0, 1, 1, 0], [1, 0, 1, 1, 0, 0])
([0, 0, 0, 1, 1, 0], [1, 0, 1, 1, 0, 0], [0, 0, 0, 1, 0, 0], [1, 0, 1, 1, 1, 0])
([0, 0, 0, 1, 1, 0], [1, 0, 1, 1, 0, 0], [0, 0, 0, 1, 0, 0], [1, 0, 1, 1, 1, 0])
6
-12
4
-14
[-31, -139, -11, -191]
<class 'tuple'>
([0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 1, 0])
([0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 1, 0], [0, 0, 0, 1, 1, 0], [0, 0, 0, 1, 0, 0])
([0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 1, 0], [0, 0, 0, 1, 1, 0], [0, 0, 0, 1, 0, 0])
4
6
6
4
[-11, -31, -31, -11]
<class 'tuple'>
([0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0])
([0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0])
([0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0], [0, 1, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0])
4
4
20
4
[-11, -11, -395, -11]
<class 'tuple'>
([0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0])
([0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0])
([0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0])
4
4
4
4
[-11, -11, -11, -11]
<class 'tuple'>
([0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0])
([0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0])
([0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0])
best score :
-11
sequence........
[0, 0, 0, 1, 0, 0]
```