بنام خدا

پیاده سازی تابع Ackley با پایتون

```
#----------------------------------فراخوانی کتابخانه ها----------------------------------
-----
import random
import numpy
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import math
from math import e
#----------------------------------------توابع تست----------------------------------------
----


def F4(x):
    dim=len(x);
    o=-20*numpy.exp(-.2*numpy.sqrt(numpy.sum(x**2)/dim))-
numpy.exp(numpy.sum(numpy.cos(2*math.pi*x))/dim)+20+numpy.exp
(1);
    return o,4;
#-------------------------پارامترهای توابع----------------------------------
def getFunctionDetails(a):
```

```
    # [name, lb, ub, dim]
    param = {  1 : ["F31",-5,5,2],
               2 : ["F32",-5,5,2] ,
               3 : ["F33",-30,30,2],
               4 : ["F4",-30,30,2]
            }
    return param.get(a, "nothing")


#----------------------------------ساخت جمعیت بعد عمل آمیزش-------------------------
-------------
def crossoverPopulaton(population, scores, popSize,
crossoverProbability, keep):
    newPopulation = numpy.empty_like(population)
    newPopulation[0:keep] = population[0:keep]
    for i  in range(keep, popSize, 2):
        parent1, parent2 = pairSelection(population, scores, popSize)
        crossoverLength = min(len(parent1), len(parent2))
        parentsCrossoverProbability = random.uniform(0.0, 1.0)
        if parentsCrossoverProbability < crossoverProbability:
            offspring1, offspring2 = crossover(crossoverLength, parent1,
parent2)
        else:
            offspring1 = parent1.copy()
```

```
            offspring2 = parent2.copy()
        newPopulation[i] = numpy.copy(offspring1)
        newPopulation[i + 1] = numpy.copy(offspring2)
    return newPopulation
#------------------------------ساخت جمعیت بعد از عملگر جهش----------------------------
-----------
def mutatePopulaton(population, popSize, mutationProbability, keep,
lb, ub):
    for i  in range(keep, popSize):
        offspringMutationProbability = random.uniform(0.0, 1.0)
        if offspringMutationProbability < mutationProbability:
            mutation(population[i], len(population[i]), lb, ub)
#-----------------------------------------------------------------------------------------
---
def elitism(population, scores, bestIndividual, bestScore):
    worstFitnessId = selectWorstIndividual(scores)
    if scores[worstFitnessId] > bestScore:
        population[worstFitnessId] = numpy.copy(bestIndividual)
        scores[worstFitnessId] = numpy.copy(bestScore)
#-------------------------برگرداندن فرد با بیشترین برازندگی-------------------------------
----
def selectWorstIndividual(scores):
    maxFitnessId = numpy.where(scores == numpy.max(scores))
```

```
    maxFitnessId = maxFitnessId[0][0]
    return maxFitnessId
#----------------------------------جفت کردن والد ها-------------------------------------
-----
def pairSelection(population, scores, popSize):
    parent1Id = rouletteWheelSelectionId(scores, popSize)
    parent1 = population[parent1Id].copy()
    parent2Id = rouletteWheelSelectionId(scores, popSize)
    parent2 = population[parent2Id].copy()
    return parent1, parent2
#----------------------------------چرخ رولت برای انتخاب-----------------------------------
------
def rouletteWheelSelectionId(scores, popSize):
    reverse = max(scores) + min(scores)
    reverseScores = reverse - scores.copy()
    sumScores = sum(reverseScores)
    pick    = random.uniform(0, sumScores)
    current = 0
    for individualId in range(popSize):
        current += reverseScores[individualId]
        if current > pick:
            return individualId
```

```
#----------------------------------عمل آمیزش----------------------------------------
-----
def crossover(individualLength, parent1, parent2):
    crossover_point = random.randint(0, individualLength - 1)
    offspring1 =
numpy.concatenate([parent1[0:crossover_point],parent2[crossover_po
int:]])
    offspring2 =
numpy.concatenate([parent2[0:crossover_point],parent1[crossover_po
int:]])
    return offspring1, offspring2
#----------------------------------عمل جهش----------------------------------------
-----
def mutation(offspring, individualLength, lb, ub):
    mutationIndex = random.randint(0, individualLength - 1)
    mutationValue = random.uniform(lb[mutationIndex],
ub[mutationIndex])
    offspring[mutationIndex] = mutationValue
#------------------------------حذف فرد با برازندگی کمتر-----------------------------
---
def clearDups(Population, lb, ub):
    newPopulation = numpy.unique(Population, axis=0)
    oldLen = len(Population)
    newLen = len(newPopulation)
```

```
    if newLen < oldLen:
        nDuplicates = oldLen - newLen
        newPopulation = numpy.append(newPopulation,
numpy.random.uniform(0,1,(nDuplicates,len(Population[0]))) *
(numpy.array(ub) - numpy.array(lb)) + numpy.array(lb), axis=0)

    return newPopulation
#----------------------------محاسبه برازندگی جمعیت----------------------------------------
-----
def calculateCost( population, popSize, lb, ub):
    scores = numpy.full(popSize, numpy.inf)
    for i in range(0,popSize):
        population[i] = numpy.clip(population[i], lb, ub)
        if (NF==1):
            scores[i] ,show= F31(population[i,:])
        elif (NF==2):
            scores[i] ,show= F32(population[i,:])
        elif (NF==3):
            scores[i] ,show= F33(population[i,:])
        elif (NF==4):
            scores[i] ,show= F4(population[i,:])
    return scores,show
#---------------------------------مرتب سازی جمعیت-------------------------------------
------
```

```
def sortPopulation(population, scores):
    sortedIndices = scores.argsort()
    population = population[sortedIndices]
    scores = scores[sortedIndices]
    return population, scores
#----------------------------------ورودی های الگوریتم-----------------------------------
---
NF=4
iters=500
lb=-10
ub=10
dim=2
popSize=8
cp = 1
mp = 0.03
keep = 2;
#----------------------------ساختن فضاهای خالی برای عملیات-----------------------------
------
if not isinstance(lb, list):
   lb = [lb] * dim
if not isinstance(ub, list):
   ub = [ub] * dim
bestIndividual=numpy.zeros(dim)
```

```
scores=numpy.random.uniform(0.0, 1.0, popSize)
bestScore=float("inf")
ga = numpy.zeros((popSize, dim))
#----------ساختن جمعیت سپس اجرای الگوریتم----------
for i in range(dim):
   ga[:, i]=numpy.random.uniform(0,1,popSize) * (ub[i] - lb[i]) + lb[i]
convergence_curve=numpy.zeros(iters)
for l in range(iters):
  ga = crossoverPopulaton(ga, scores, popSize, cp, keep)
  mutatePopulaton(ga, popSize, mp, keep, lb, ub)
  ga = clearDups(ga, lb, ub)
  scores,show = calculateCost(ga, popSize, lb, ub)
  bestScore = min(scores)
  ga, scores = sortPopulation(ga, scores)
  convergence_curve[l]=bestScore
  if (l%1==0):
    print([str(bestScore)+'Behtarin Barazandegist Dar Ejraye'+
str(l+1)+'om']);
#----------رسم نمودار بهینه شده----------
plt.plot(convergence_curve)
plt.xlabel('Tedade Ejra')
```

```
plt.ylabel('Min')
plt.title('Nemodar Min/Ejra')
#----------رسم توابع به صورت کانتور----------
if (show==4):
    def drawing3D(x1, x2):
        return ((-20*numpy.exp(-.2*numpy.sqrt((((x1)**2)+((x2)**2))/2))-
numpy.exp(((numpy.cos(2*math.pi*(x1)))+(numpy.cos(2*math.pi*(x2))
)))/2)+20+numpy.exp(1))
    x1 = numpy.linspace(-30,30,50)
    x2 = numpy.linspace(-30,30,50)
    X, Y = numpy.meshgrid(x1, x2)
    Z = drawing3D(X, Y)
    fig = plt.figure()
    ax1 = plt.axes(projection='3d')
    ax1.contour3D(X, Y, Z,1000)
    ax1.set_xlabel('x1')
    ax1.set_ylabel('x2')
    ax1.set_zlabel('f(x1,x2)');
    ax1.view_init(30, 60)
    fig
elif (show==31):
    def drawing3D(x1, x2):
```

```
        return ((0.5)+(((numpy.sin(numpy.sqrt(((x1)**2)+((x2)**2))))**2)-(0.5))/(1+0.1*(((x1)**2)+((x2)**2))))
    x1 = numpy.linspace(-5,5,50)
    x2 = numpy.linspace(-5,5,50)
    X, Y = numpy.meshgrid(x1, x2)
    Z = drawing3D(X, Y)
    fig = plt.figure()
    ax1 = plt.axes(projection='3d')
    ax1.contour3D(X, Y, Z,1000)
    ax1.set_xlabel('x1')
    ax1.set_ylabel('x2')
    ax1.set_zlabel('f(x1,x2)');
    ax1.view_init(30, 60)
    fig
elif (show==32):
    def drawing3D(x1, x2):
        return (-e**((-0.2)*(numpy.sqrt(((x1)**2)+((x2)**2)))+(3*numpy.cos(2*(x1))+3*numpy.sin(2*(x2)))))
    x1 = numpy.linspace(-5,5,50)
    x2 = numpy.linspace(-5,5,50)
    X, Y = numpy.meshgrid(x1, x2)
    Z = drawing3D(X, Y)
```

```
    fig = plt.figure()
    ax1 = plt.axes(projection='3d')
    ax1.contour3D(X, Y, Z,1000)
    ax1.set_xlabel('x1')
    ax1.set_ylabel('x2')
    ax1.set_zlabel('f(x1,x2)');
    ax1.view_init(30, 60)
    fig
elif (show==33):
    def drawing3D(x1, x2):
        return
(((3)*(((numpy.cos(2*(x1))))+(numpy.sin(2*(x2)))))+numpy.exp(((-
0.2)*(numpy.sqrt(((x1)**2)+((x2)**2))))))
    x1 = numpy.linspace(-30,30,50)
    x2 = numpy.linspace(-30,30,50)
    X, Y = numpy.meshgrid(x1, x2)
    Z = drawing3D(X, Y)
    fig = plt.figure()
    ax1 = plt.axes(projection='3d')
    ax1.contour3D(X, Y, Z,1000)
    ax1.set_xlabel('x1')
    ax1.set_ylabel('x2')
    ax1.set_zlabel('f(x1,x2)');
```

```
    ax1.view_init(30, 60)
    fig
plt.show()
```

نتایج: