*# coding=utf-8*

*# 目标式子: y = 10 \* sin(5x) + 7 \* cos(4x),0 <= x <= 10，计算其最大值*

*# 首先是初始化，包括具体要计算的式子、种群数量、染色体长度、交配概率、变异概率等。并且要对基因序列进行初始化*

**import** random

**import** math

**import** numpy

**import** matplotlib.pyplot **as** plt

*#染色体编码*

**def** geneEncoding(pop\_size, chrom\_length):

pop = [[]]

**for** i **in** range(pop\_size):

temp = []

**for** j **in** range(chrom\_length):

temp.append(random.randint(0, 1))

pop.append(temp)

**return** pop[1:]

*# 找出适应函数值中最大值，和对应的个体*

**def** best(pop, fitvalue):

px = len(pop)

bestindividual = []

bestfit = fitvalue[0]

**for** i **in** range(1,px):

**if**(fitvalue[i] > bestfit):

bestfit = fitvalue[i]

bestindividual = pop[i]

**return** [bestindividual, bestfit]

*# 转化为适应值，目标函数值越大越好，负值淘汰。*

**def** calfitvalue(objvalue):

fitvalue = []

temp = 0.0

Cmin = 0;

**for** i **in** range(len(objvalue)):

**if**(objvalue[i] + Cmin > 0):

temp = Cmin + objvalue[i]

**else**:

temp = 0.0

fitvalue.append(temp)

**return** fitvalue

*# 将种群的二进制基因转化为十进制（0,1023）*

**def** decodechrom(pop):

temp = [];

**for** i **in** range(len(pop)):

t = 0;

**for** j **in** range(10):

t += pop[i][j] \* (math.pow(2, j))

temp.append(t)

**return** temp

*# 计算目标函数值*

**def** calobjvalue(pop):

temp1 = [];

objvalue = [];

temp1 = decodechrom(pop)

**for** i **in** range(len(temp1)):

x = temp1[i] \* 10 / 1023 *#（0,1023）转化为 （0,10）*

*#objvalue.append(-numpy.sqrt(x) + 10\*math.cos(2\*x) + 30)*

objvalue.append(10 \* math.sin(5 \* x) + 7 \* math.cos(4 \* x))

**return** objvalue *#目标函数值objvalue[m] 与个体基因 pop[m] 对应*

*# 个体间交叉，实现基因交换*

**def** crossover(pop, pc):

poplen = len(pop)

**for** i **in** range(poplen - 1):

**if**(random.random() < pc):

cpoint = random.randint(0,len(pop[0]))

temp1 = []

temp2 = []

temp1.extend(pop[i][0 : cpoint])

temp1.extend(pop[i+1][cpoint : len(pop[i])])

temp2.extend(pop[i+1][0 : cpoint])

temp2.extend(pop[i][cpoint : len(pop[i])])

pop[i] = temp1

pop[i+1] = temp2

*# 基因突变*

**def** mutation(pop, pm):

px = len(pop)

py = len(pop[0])

**for** i **in** range(px):

**if**(random.random() < pm):

mpoint = random.randint(0,py-1)

**if**(pop[i][mpoint] == 1):

pop[i][mpoint] = 0

**else**:

pop[i][mpoint] = 1

*#求和*

**def** sum(fitvalue):

total = 0

**for** i **in** range(len(fitvalue)):

total += fitvalue[i]

**return** total

*#适应度求和*

**def** cumsum(fitvalue):

**for** i **in** range(len(fitvalue)):

t = 0;

j = 0;

**while**(j <= i):

t += fitvalue[j]

j = j + 1

fitvalue[i] = t;

*# 自然选择（轮盘赌算法）*

**def** selection(pop, fitvalue):

newfitvalue = []

totalfit = sum(fitvalue)

**for** i **in** range(len(fitvalue)):

newfitvalue.append(fitvalue[i] / totalfit)

cumsum(newfitvalue)

ms = [];

poplen = len(pop)

**for** i **in** range(poplen):

ms.append(random.random()) *#random float list ms*

ms.sort()

fitin = 0

newin = 0

newpop = pop

**while** newin < poplen:

**if**(ms[newin] < newfitvalue[fitin]):

newpop[newin] = pop[fitin]

newin = newin + 1

**else**:

fitin = fitin + 1

pop = newpop

*# 计算2 进制序列代表的数值, 将二进制转化为十进制 x∈[0,10]*

**def** b2d(b, max\_value, chrom\_length):

t = 0

**for** j **in** range(len(b)):

t += b[j] \* (math.pow(2, j))

t = t \* max\_value / (math.pow(2, chrom\_length) - 1)

**return** t

*# 以下是主程序*

pop\_size = 100 *# 种群数量*

max\_value = 10 *# 基因中允许出现的最大值,根据已知确定*

gen\_size = 300 *# 遗传代数，源代码中用种群数量pop\_size代替*

chrom\_length = 10 *# 染色体长度*

pc = 0.6 *# 交配概率*

pm = 0.01 *# 变异概率*

results = [[]] *# 存储每一代的最优解，N 个二元组*

fit\_value = [] *# 个体适应度*

fit\_mean = [] *# 平均适应度*

*# 如果采用geneEncoding函数产生初始群体，则需要多执行几次，有时会出现除以0的情况*

pop = geneEncoding(pop\_size, chrom\_length)

print(**"pop的初始种群(100个)："**)

**for** m **in** pop:

print(m)

**for** i **in** range(gen\_size): *# 遗传代数=pop\_size种群数量*

obj\_value = calobjvalue(pop) *# 个体评价*

fit\_value = calfitvalue(obj\_value) *# 淘汰*

best\_individual, best\_fit = best(pop, fit\_value) *# 第一个存储最优的解, 第二个存储最优基因*

results.append([best\_fit, b2d(best\_individual, max\_value, chrom\_length)])

selection(pop, fit\_value) *# 新种群复制*

crossover(pop, pc) *# 交配*

mutation(pop, pm) *# 变异*

results = results[1:] *# results是一个二维数组，未改变二维*

results.sort()

print(**"for循环逐项输出results"**)

**for** k **in** results:

print(k)

*#图形展示结果*

X = []

Y = []

**for** i **in** range(gen\_size):

X.append(i)

t = results[i][0] *# 只要第一列*

Y.append(t)

print(**"打印函数最大值和对应的y和x值:"**)

print(results[-1]) *#-1表示从数组最后一个元素*

plt.plot(X, Y)

plt.show()