遗传算法：

import math  
import random

def b2d(b): # 将二进制转化为十进制 x∈[0,10]  
 t = 0  
 for j in range(len(b)):  
 t += b[j] \* (math.pow(2, j))  
 t = t \* 10 / 1023  
 return t  
  
popsize = 50 # 种群的大小  
# 用遗传算法求函数最大值：  
# f(x)=10\*sin(5x)+7\*cos(4x) x∈[0,10]  
chromlength = 10 # 基因片段的长度  
pc = 0.6 # 两个个体交叉的概率  
pm = 0.001 # 基因突变的概率  
  
results = [[]]  
bestindividual = []  
bestfit = 0  
fitvalue = []  
tempop = [[]]  
pop = [[0, 1, 0, 1, 0, 1, 0, 1, 0, 1] for i in range(popsize)]  
  
def calfitvalue(objvalue): # 转化为适应值，目标函数值越大越好，负值淘汰。  
 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  
  
def decodechrom(pop): # 将种群的二进制基因转化为十进制（0,1023）  
 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  
  
def calobjvalue(pop): # 计算目标函数值  
 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(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  
  
def best(pop, fitvalue): # 找出适应函数值中最大值，和对应的个体  
 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]  
  
def sum(fitvalue):  
 total = 0  
 for i in range(len(fitvalue)):  
 total += fitvalue[i]  
 return total  
  
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  
  
def selection(pop, fitvalue): # 自然选择（轮盘赌算法）  
 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  
  
def crossover(pop, pc): # 个体间交叉，实现基因交换  
 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  
  
def mutation(pop, pm): # 基因突变  
 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  
  
for i in range(10): # 繁殖10代  
 objvalue = calobjvalue(pop) # 计算目标函数值  
 fitvalue = calfitvalue(objvalue) # 计算个体的适应值  
 [bestindividual, bestfit] = best(pop, fitvalue) # 选出最好的个体和最好的函数值  
 results.append([bestfit, b2d(bestindividual)]) # 每次繁殖，将最好的结果记录下来  
 selection(pop, fitvalue) # 自然选择，淘汰掉一部分适应性低的个体  
 crossover(pop, pc) # 交叉繁殖  
 mutation(pop, pc) # 基因突变  
  
results.sort()  
  
print(results[-1]) # 打印函数最大值和对应的

运行结果：

