中山大學
SUN YAT-SEN UNIVERSITY

人工智能程设

题目 Title：期末大作业

院 系
School (Department)：智能工程学院

专业
Major：智能科学与技术

学生姓名
Student Name：周德峰，林佳盈

学 号
Student No.：21312210，21312237

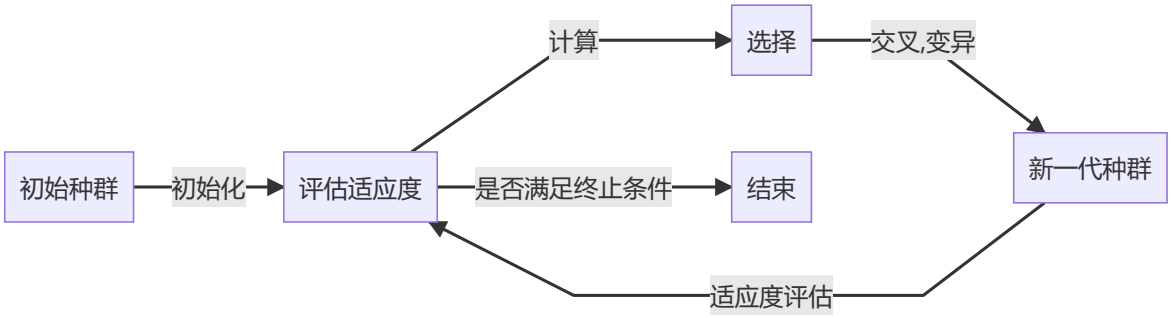
指导教师(职称)
Supervisor （Title）：王帅

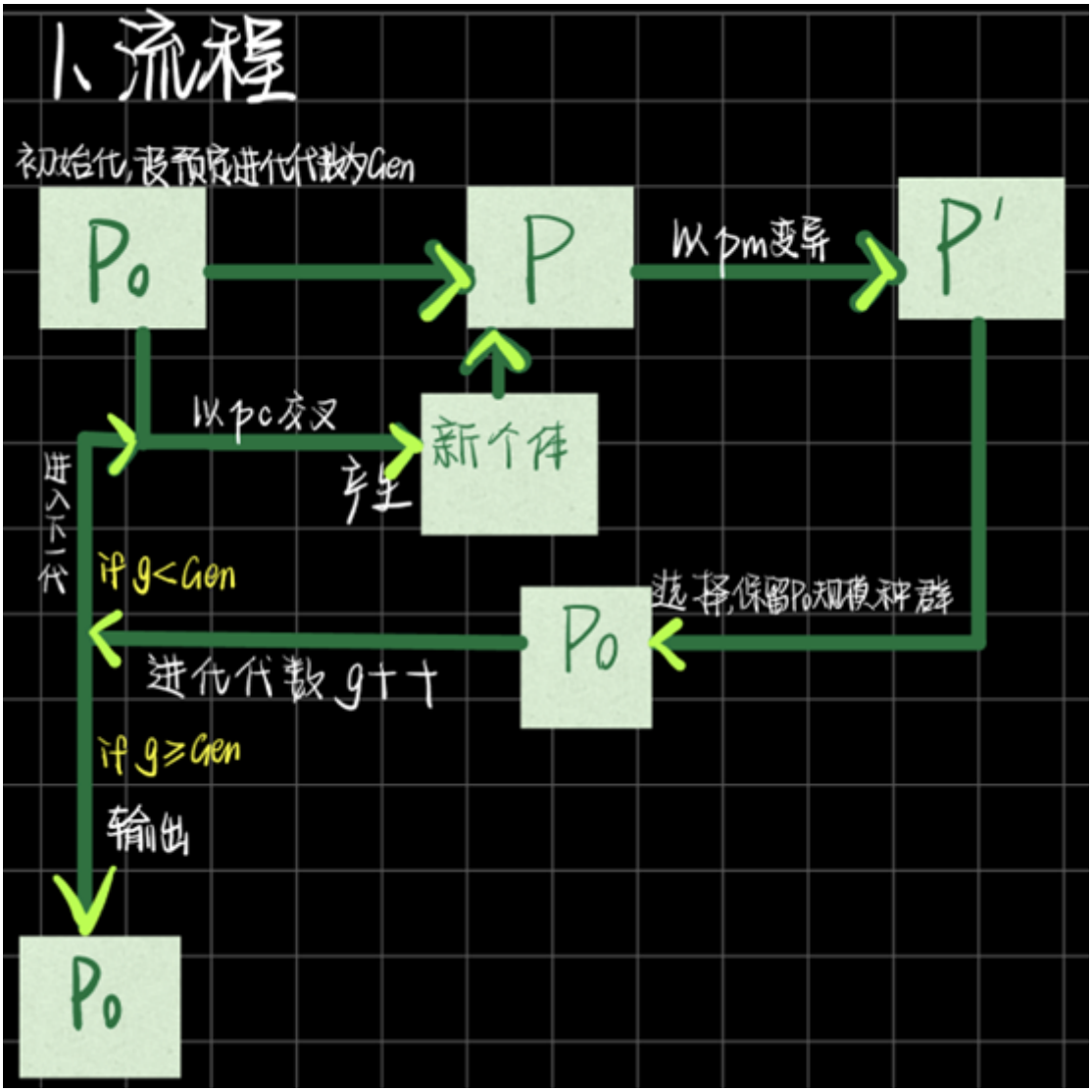
时间： 2022 年 12 月 17 日

前言

本实验由周德峰与林佳盈合作完成，周德峰负责python部分，林佳盈负责matlab部分，在实验过程中两人互相交流，对算法有了更深的理解

流程图





示例

编码

核心代码

python

关于编码第一位的问题，由于题目此处没有说明清楚，所以经协商我们决定将第一位整数位表示为不固定数位的二进制，即2->10　4->100，以此递推

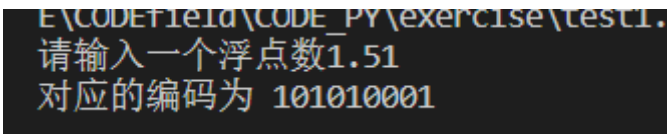
```python
def encode(n):     #编码!
    n=str(round(n,2))
    n_=n.split(".")      #讲字符串在.号拆分
    n1=int(n_[0])        #取出整数位

    list1=list(n_[1])    #先将其转为列表
    list1.reverse()      #反转，因为后续输出是反转输出
    s1=(''.join(list1)) #转为字符串，方便后续转为int
    n2=int(s1)           #得到反转后数字形式
    code=[]

```

```
    def two(n):
        b = []  # 存储余数
        while n:  # 一直循环，商为0时利用break退出循环
            b.append(n%2)
            n = n // 2  # 商
        while len(b)<4:
            b.append(0)
        b.reverse()  # 使b中的元素反向排列
        b = [ str(i) for i in b ]
        b=(''.join(b))
        #print ('该数字转换为二进制后是：')
        #print(b)
        return b

    n1=two(n1).lstrip('0')  #除去前导0
    code.append(n1)
    while n2!=0:    #对小数点后的数字进行转码
        remainder=n2%10
        code.append(two(remainder))
        n2=n2//10
    code = [ str(i) for i in code ]
    code=(''.join(code))
    #print("the code is :")
    #print(code)
    return code
```

matlab

```
function X_2=ten2two(X_)
% 功能：十进制转换二进制
% 输入：十进制的二位小数
% 输出：二进制的字符串
X=X_*100; % X_是一个二位小数 令其变为三位数整数
X1=rem(X,10); % 个位
X2=rem(floor(X/10),10); % 十位
X3=rem(floor(X/100),10); % 百位
X_2=string(sprintf(dec2bin(X3)))+string(sprintf('%04s',dec2bin(X2)))+string(sprintf('%04s',dec2bin(X1))); % 转为二进制 不足位数的补零
X_2=string(X_2);
end

```

实现及结果



```
>> main2
请输入一个浮点数：1.51
101010001
>> main2
请输入一个浮点数：1.52
101010010
```

交叉

核心代码

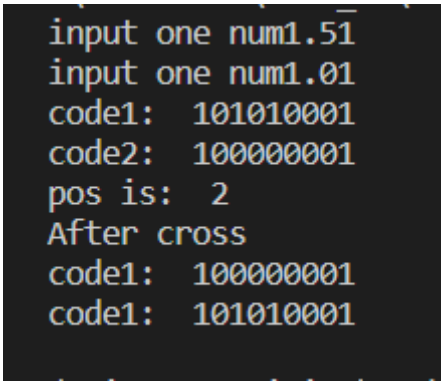
python

```python
def cross(code1, code2):
    cron=int(P*p_m)     #需要交叉的数量
    len1=len(code1)
    len2=len(code2)
    minlen=min(len1,len2)
    pos=random.randint(0,minlen-1)  #发生交叉的位置pos
    print("pos is: " ,pos+1)
    str1=code1[pos:]
    str2=code2[pos:]
    newcode1=code1[0:pos]+str2
    newcode2=code2[0:pos]+str1
    return newcode1, newcode2
```

matlab

```matlab
function result=exchange_info(code1,code2)
% 功能：两两交叉
% 输入：二进制编码（字符串）
% 输出：交叉后的结果

% 转换为char型
char1=char(code1);
char2=char(code2);

% 交叉拼接
result_1=string(char1(1:4))+string(char2(5:9));
result_2=string(char2(1:4))+string(char1(5:9));
result=[result_1,result_2];
end

```

实现及结果




```
disp(exchange_info("101010001","100000001"))

    >> main2
        "101000001"    "100010001"
```

变异

核心代码

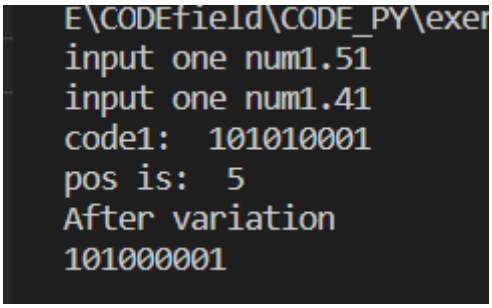
python

```python
def variation(code):
    #varn=int(P*p_m)       #变异的数量
    #sam=random.sample(range(P),varn)       #采样
    #for i in range(varn):
    #code=codes[sam[i]]     #取出对应index的code
    pos=random.randint(0,len(code)-1)
    #print(code)
    ls=list(code)
    print("pos is: " ,pos+1)
    ls[pos]=str(int(not int(code[pos])))
    code=(''.join(ls))
    return code
```

matlab

```matlab
function X_1=variation(X_1)
% 功能：变异
% 输入：变异对象
% 输出：变异后的结果
X_1=char(X_1); % 转换为字符数组
% 对最后一位变异
if X_1(length(X_1))=='1' % 若最后一位是1,变异为0
X_1(length(X_1))='0';
else
X_1(length(X_1))='1';
end
```

实验结果




```
disp(variation("101010001"))
```

```
>> main2
101010000
```

进化算法

python

将初始化，交叉，变异，迭代全部封装成一个类heredity

问题1

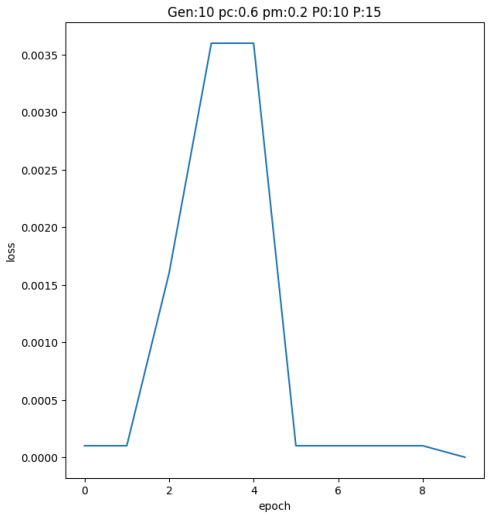
关于编码，交叉，变异的代码已在上文详细叙述，在类内实现时只需稍加修改即可，大体思路不变，所以下面介绍其他三个函数：解码，择优，forward函数（具体类的实现详见后文源码）

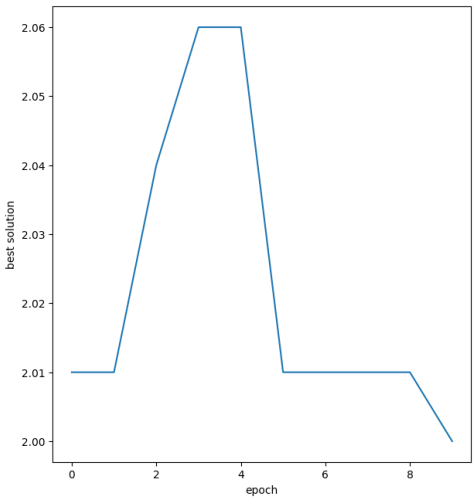
```python
    def decodes(self,code):
        len1=len(code)%4
        q=len(code)//4
        n1=int(code[0:len1],2)
        bit=0
        n2=0
        #每隔四位取出
        while q:
            n3=int(code[len1:len1+4],2)
            n2=n2*10+n3
            bit+=1
            len1+=4
            q-=1
        n2=n2/(10**bit)
        num=n1+n2
        return num
    def lossfunction(self, value):
        return (value-2)**2
    def choose(self):
        loss=[]
        numbers=[]
        for i in range(len(self.codes)):
            numbers.append(self.decodes(self.codes[i]))
        #self.numbers=copy.deepcopy(numbers)
        for i in range(self.P):
            loss.append(self.lossfunction(numbers[i]))
```

```
        def getListMinNumIndex(num_list,topk=self.P_O):
            #tmp_list=copy.deepcopy(num_list)
            #tmp_list.sort()
            #min_num_index=[num_list.index(one) for one in tmp_list[:topk]]
存在重复值，不可用!
            #print ('min_num_index:',min_num_index)
            min_number = heapq.nsmallest(topk, num_list)
            min_index = []
            for t in min_number:
                index = num_list.index(t)
                min_index.append(index)
                a=float('-inf')
                num_list[index] = a
            return min_index, min_number[0],min_index[0]
        index, min, best_index=getListMinNumIndex(loss)
        best_solution=numbers[best_index]
        #选择出来后，更新!
        #self.numbers=copy.deepcopy(numbers[index])
        self.numbers=[numbers[i] for i in index]
        self.codes=[self.codes[i] for i in index]
        return min,best_solution        #返回每一轮的最小loss值和最优解
    def forward(self):
        #初始化
        self.inputinit()
        minlist=[]
        best_list=[]
        for i in range(self.Gen):
            #交叉
            self.cross()
            #变异
            self.variation()
            #选择优化
            min,best_solution=self.choose()
            best_list.append(best_solution)
            minlist.append(min)
        return minlist,best_list        #返回最后一次迭代numbers和每次迭代的最优解
loss值
```

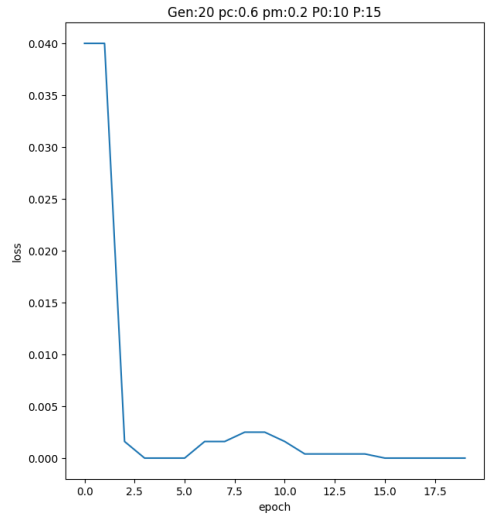
问题2

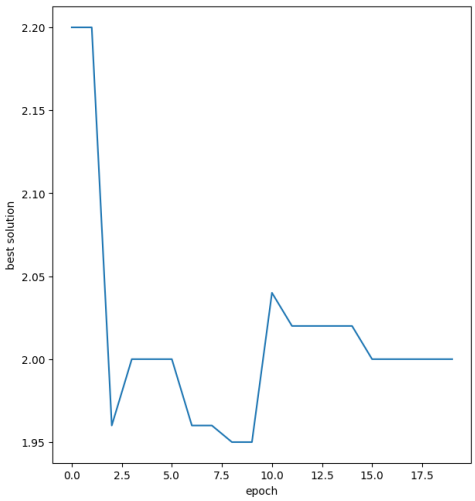
Test1



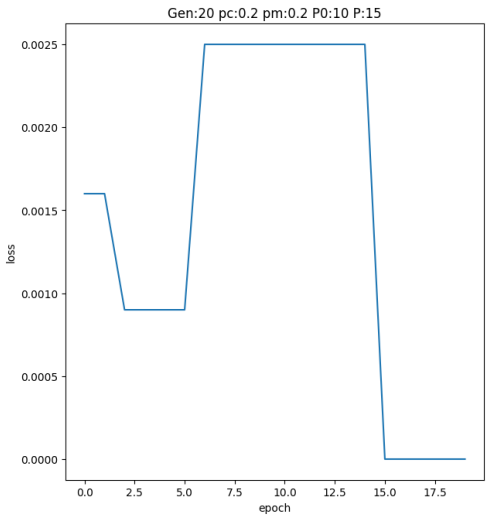


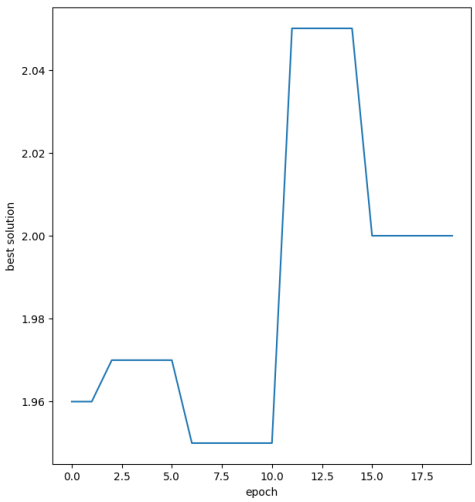
Test2



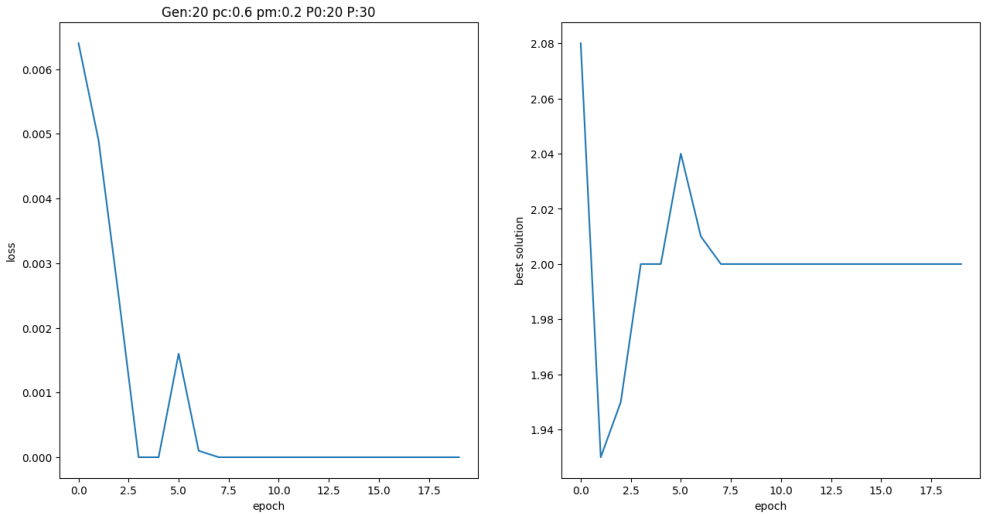


Test3

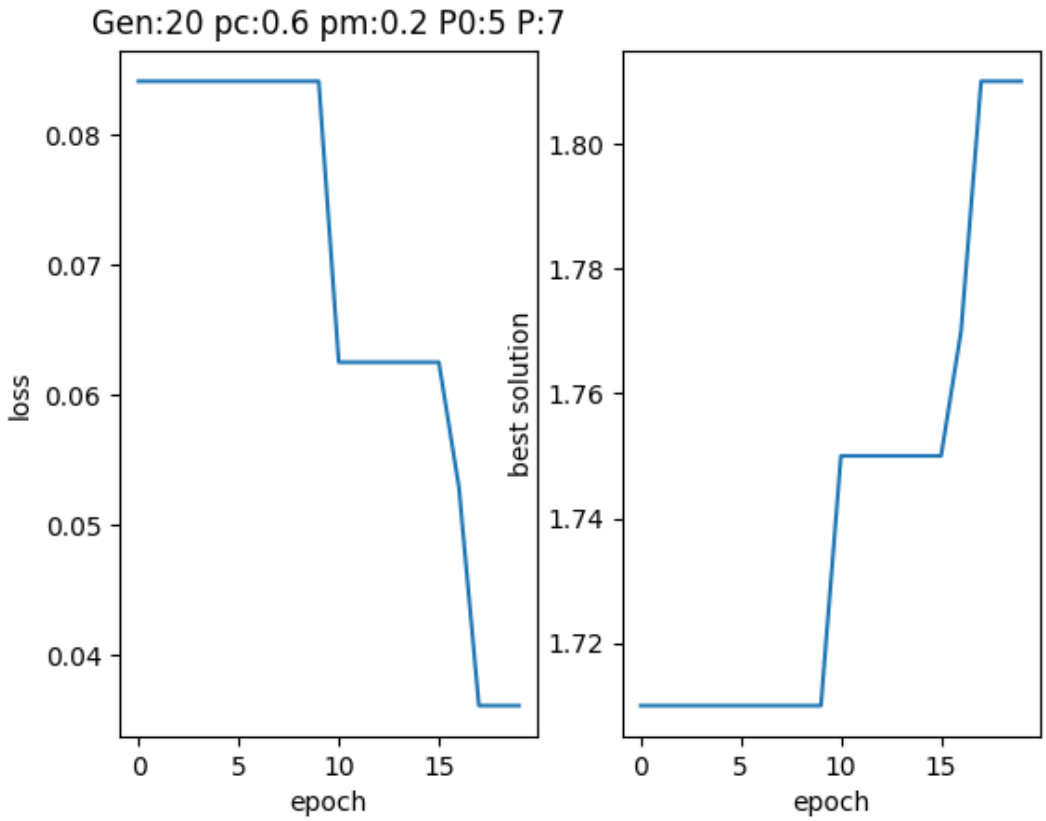




Test4



Test5



问题3

- **不同参数对算法性能的影响**

1. 根据图2可知，当**迭代次数越多**时，遗传算法的**收敛速度越快，精度越高**，但同时，迭代次数越多，计算复杂度也会增加
2. 根据图3可知，当**交叉概率较小**时，会导致遗传算法的**收敛速度变慢，精度降低。**这是因为，在交叉概率较小的情况下，个体之间的基因交叉较少，新基因的产生也就相对较少，算法的探索能力就会受到限制
3. 根据图4可知，当**种群规模增大**时，遗传算法的**探索能力更强**，能更稳定的收敛到最优解，但同时计算复杂度和计算时间会相应增加。

4. 根据图5不难看出，当**P0和P都较小**时，算法会比较难以收敛到最优解，原因是当种群数量太小时，
   交叉和变异的样本数较少，因此种群的**多样性**也会较小，所以会比较**难以收敛到最优解。**

- **每次运行获得的最优解是否一致**

观察下图，即可发现每次得到的最优解不一致，一部分会收敛到**全局最优解**，一部分则**没有收敛**，一部分则会收敛到**局部最优解**

推测与三个原因有关

1. 样本的**初始随机化**（因为遗传算法是**启发式算法**，所以会收敛到局部最优解，而不一定是全局最优解）
2. **迭代次数**的限制，若样本初始随机解离最优解较远，且迭代次数较少，则无法收敛或者收敛到局部最优
3. 其他**算法参数**的影响（上问已讨论过）

```
ython-2022.20.1\pythonFiles\lib\python\debugpy\adapter/../..\
E\CODEfield\CODE_PY\homework\遗传算法.py "
最后一次最优解为:  2.0

(VoiceRecognition) E:\VSCODE\CODEfield\CODE_PY> e: && cd e:\V
 "E:\Anaconda3\envs\VoiceRecognition\python.exe c:\Users\lenc
ython-2022.20.1\pythonFiles\lib\python\debugpy\adapter/../..\
E\CODEfield\CODE_PY\homework\遗传算法.py "
最后一次最优解为:  2.0

(VoiceRecognition) E:\VSCODE\CODEfield\CODE_PY> e: && cd e:\V
 "E:\Anaconda3\envs\VoiceRecognition\python.exe c:\Users\lenc
ython-2022.20.1\pythonFiles\lib\python\debugpy\adapter/../..\
E\CODEfield\CODE_PY\homework\遗传算法.py "
最后一次最优解为:  2.0

(VoiceRecognition) E:\VSCODE\CODEfield\CODE_PY> e: && cd e:\V
 "E:\Anaconda3\envs\VoiceRecognition\python.exe c:\Users\lenc
ython-2022.20.1\pythonFiles\lib\python\debugpy\adapter/../..\
E\CODEfield\CODE_PY\homework\遗传算法.py "
最后一次最优解为:  1.97

(VoiceRecognition) E:\VSCODE\CODEfield\CODE_PY> e: && cd e:\V
 "E:\Anaconda3\envs\VoiceRecognition\python.exe c:\Users\lenc
ython-2022.20.1\pythonFiles\lib\python\debugpy\adapter/../..\
E\CODEfield\CODE_PY\homework\遗传算法.py "
最后一次最优解为:  2.0

(VoiceRecognition) E:\VSCODE\CODEfield\CODE_PY> e: && cd e:\V
 "E:\Anaconda3\envs\VoiceRecognition\python.exe c:\Users\lenc
ython-2022.20.1\pythonFiles\lib\python\debugpy\adapter/../..\
E\CODEfield\CODE_PY\homework\遗传算法.py "
最后一次最优解为:  1.9300000000000002

(VoiceRecognition) E:\VSCODE\CODEfield\CODE_PY> e: && cd e:\V
 "E:\Anaconda3\envs\VoiceRecognition\python.exe c:\Users\lenc
ython-2022.20.1\pythonFiles\lib\python\debugpy\adapter/../..\
E\CODEfield\CODE_PY\homework\遗传算法.py "
最后一次最优解为:  1.81
```

- **心得体会**

1. **将各个函数封装成一个类，会让后续的维护，调整参数等各种操作更加简便，快捷**
2. 具体实现代码方面，在选出**较优个体的index**时，会出现多个物体的**loss值相同**，而我们应把这些符合条件的index都取出，而不是取出重复的index
3. 在编码时，采取统一标准：先保留三位有效数字再进行编码，提高了**代码的简洁性**以及避免了**不必要的复杂计算**
4. 做出了**Loss随epoch变化**的图像，可以更加清楚的看到**迭代过程中算法是否异常**

matlab

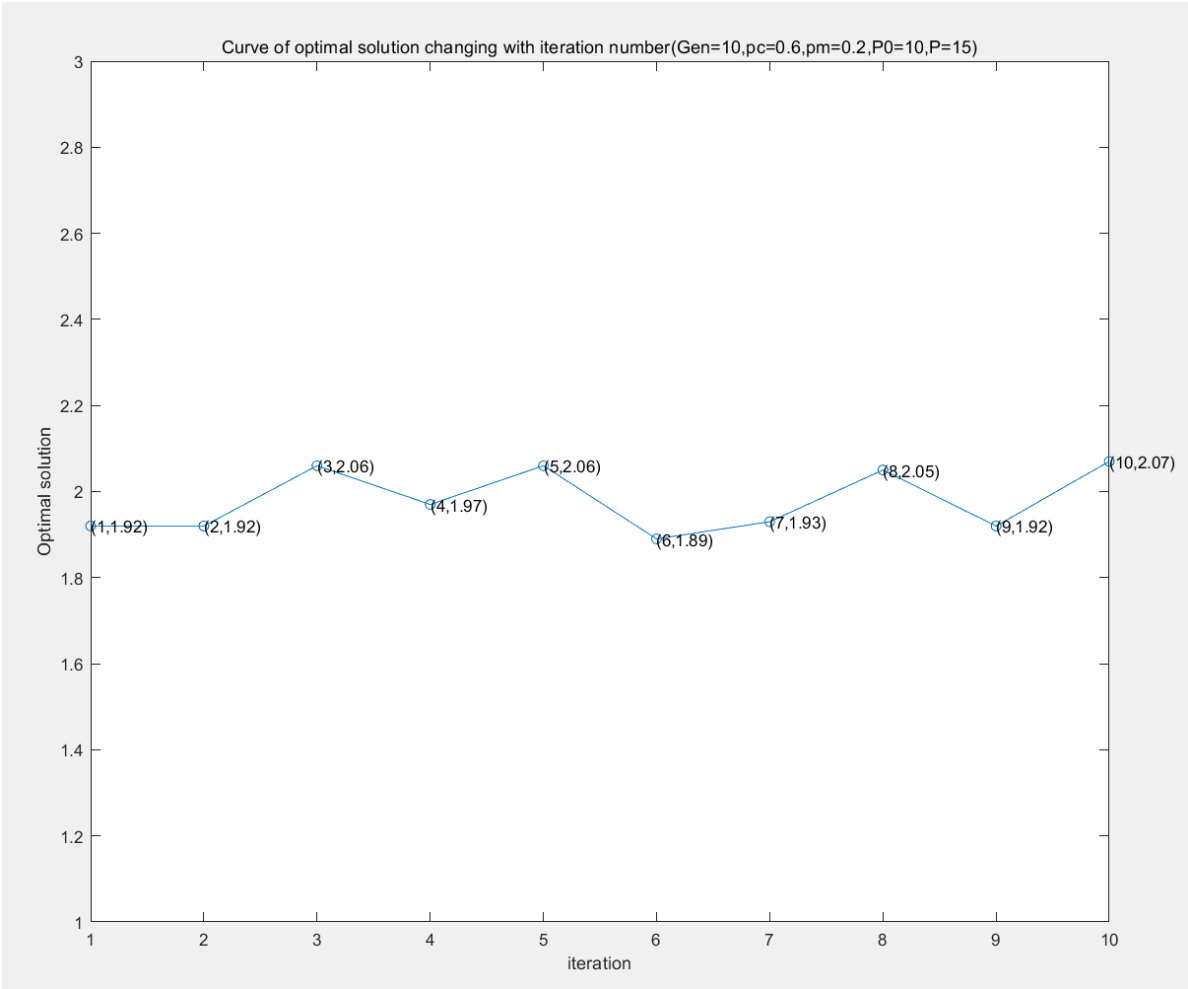
问题1

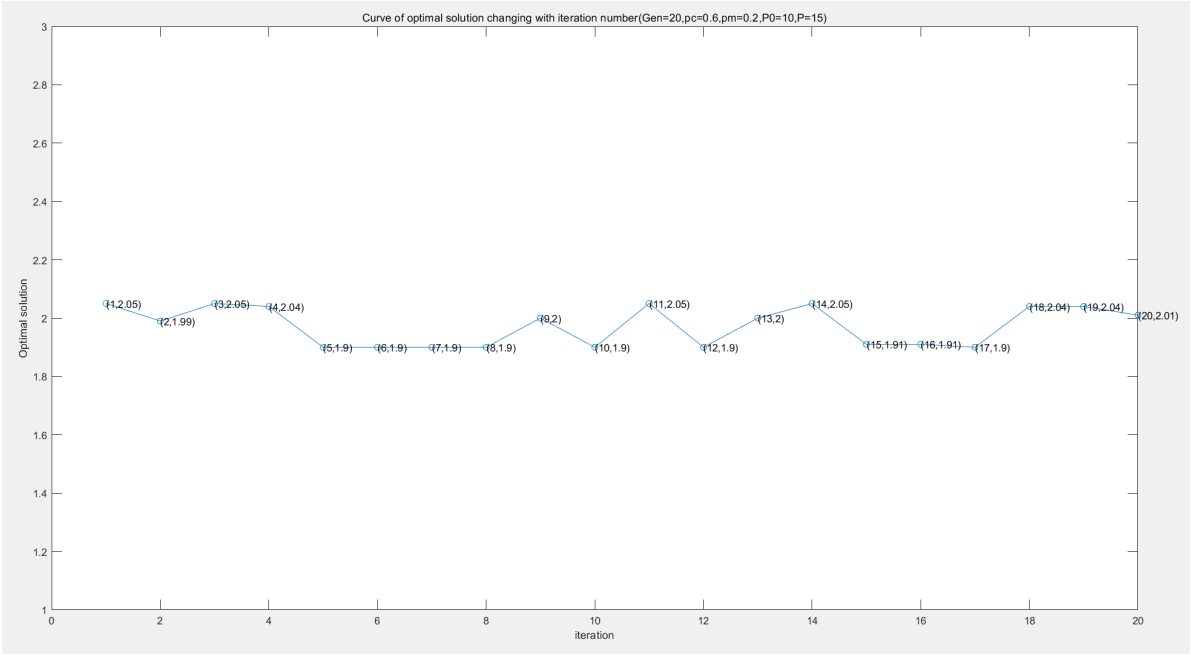
同样，编码，交叉，变异函数已在上文中涉及，此处不再赘述

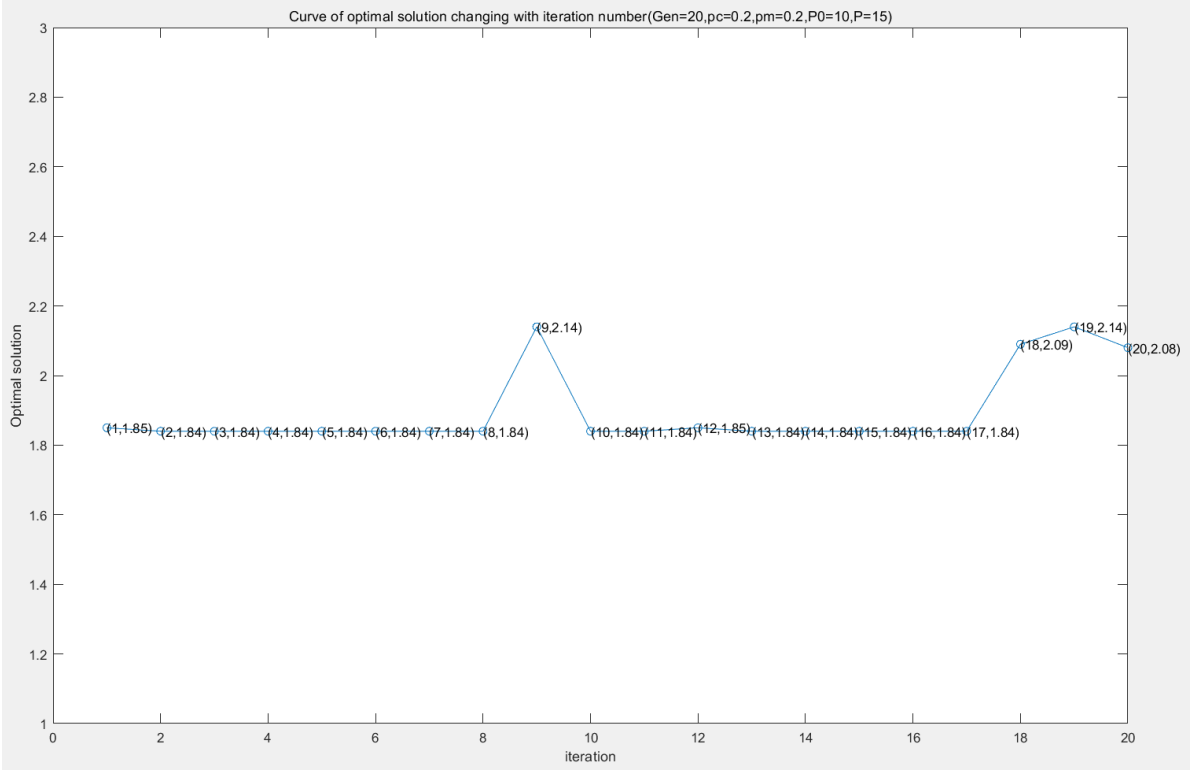
```
function result=match(L,final_L)
% 功能：产生两两配对的L*2维匹配结果（是index）
% 输入：需要进行配对的数的个数
% 输出：L*2维匹配结果（是index）
%记录配对结果，n*2维的矩阵
if rem(L,2)~=0
L=L-1; % 转为偶数
end
result = zeros(L/2,2);

%matlab的索引是从1开始的
i = 1;
%记录已经出现过的下标
index = zeros(L,1);

while i<=L/2 %直到生成所有交叉对
%生成进行交叉操作的二者下标序号
x = randi(L);
y = randi(L);

%保证不与先前的随机序号重复
while ~all(ismember(index,[x,y])==0) % index中是否x,y已经匹配过
x = randi(L);
y = randi(L);
end

%在x和y不相等的条件下进行配对结果的更新
if(x~=y)
result(i,:)=[x,y];
i=i+1;
index(x)=x;
index(y)=y;
end
end
if rem(final_L,2)~=0
final_L=final_L-1; % 转为偶数
end
result=result(1:final_L/2,:);

function result=Optimize(P0,X)
% 功能：从种群中选择一些较优个体，保留进入下一代规模为 P0 种群
% 输入：选择出的规模，种群
% 输出：选优结果
X_=[];
for i=1:length(X)
X_=[X_,two2ten(X(i))]; % 转换十进制
end

X_=reshape(X_,length(X_),1);
Y=reshape(round((X_-2).^2,2),length(X_),1);

```

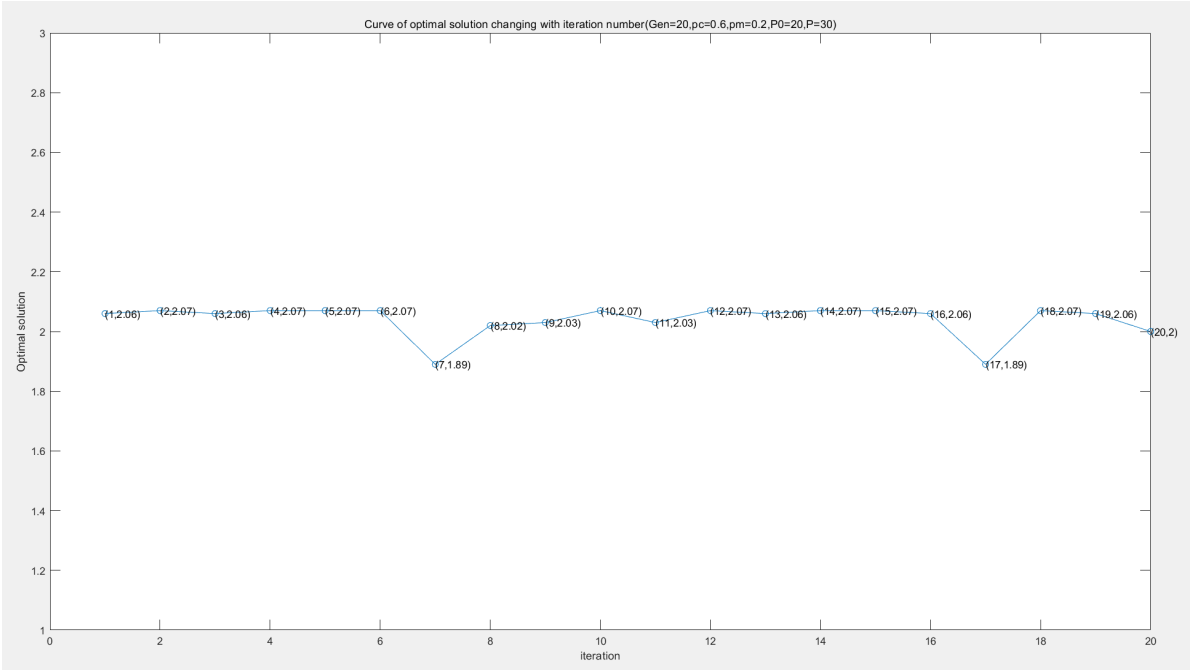
```
res=[X_,Y];
res2=sortrows(res,2); % 按照距离大小排序
result=res2(1:P0,1); % 保留P0进入下一代
end

```

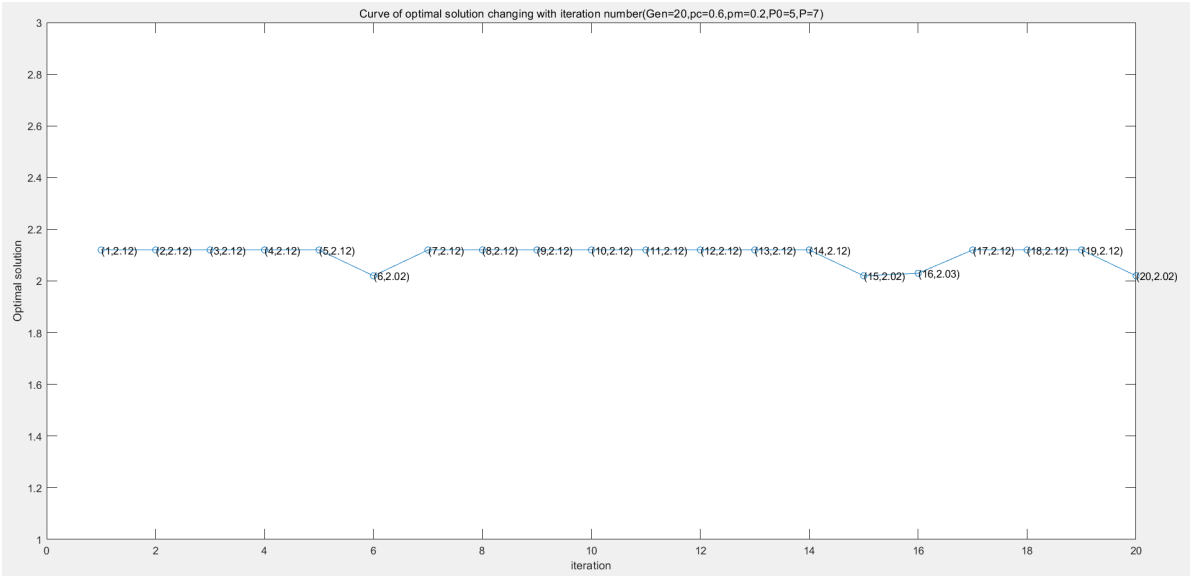
问题2




Curve of optimal solution changing with iteration number(Gen=20,pc=0.6,pm=0.2,P0=10,P=15)
Optimal solution
iteration
(1,2.05)
(2,1.99)
(3,2.05)
(4,2.04)
(5,1.9)
(6,1.9)
(7,1.9)
(8,1.9)
(9,2)
(10,1.9)
(11,2.05)
(12,1.9)
(13,2)
(14,2.05)
(15,1.91)
(16,1.91)
(17,1.9)
(18,2.04)
(19,2.04)
(20,2.01)


Curve of optimal solution changing with iteration number(Gen=20,pc=0.2,pm=0.2,P0=10,P=15)
Optimal solution
iteration
(1,1.85)
(2,1.84)
(3,1.84)
(4,1.84)
(5,1.84)
(6,1.84)
(7,1.84)
(8,1.84)
(9,2.14)
(10,1.84)
(11,1.84)
(12,1.85)
(13,1.84)
(14,1.84)
(15,1.84)
(16,1.84)
(17,1.84)
(18,2.09)
(19,2.14)
(20,2.08)





问题3

影响：

- P0
  - 规模较大的群体可以避免算法陷入局部最优解，但增大群体规模的同时也会增加计算复杂度。
  - 规模选择过小会使搜索空间分布范围不足，因而搜索有可能停止在一个次优解。
- P
  - P增大时，算法的搜索空间更广，种群更具多样性，更有利于收敛到全局最优解，但同时也会增加计算复杂度。
- 迭代次数
  - 迭代次数越多，遗传算法的收敛速度越快，精度越高，但同时会增加计算复杂度、降低算法效率
- 交叉概率
  - 当交叉概率比较小时，产生的新基因较少，群体多样性较低，搜索的范围扩展较慢，但能够较大程度的保留现有基因。
  - 当交叉概率比较大时，有助于增加种群多样性，是提高全局最优搜索能力，但是上一代最优解被过滤的概率也随之增大

每次运行获得的最优解是否一致

由图可知，每次得到的最优解并不一致，其大部分收敛于全局最优解2的附近，部分收敛至局部最优解，部分在迭代次数内未达到收敛。

原因：

- 初始值的选择：进化算法在每次迭代中会收敛到局部最优。若初始值和全局最优解差的较大，则其更难收敛。
- 算法中参数的影响：算法中的P0、P、迭代次数、交叉概率、变异概率都会对运行得到的最优解产生影响。具体分析见第一问。

心得

- **代码：**
  - 巧用Matlab内置函数：Matlab为用户提供了丰富的内置函数，使用内置函数不仅可以提高运行效率，也可以使代码更简洁、清晰。
  - 善写函数、善于设置超参数：函数封装可使得代码更清晰明了、更具可读性，利于后续维护。设置超参数便于后续的调整。
- **算法：**在第一问提到，当交叉、变异概率过大时，可能会过滤掉当前的最优解。为解决这个问题，可以采用一种保留策略，使上一代的当前最优解强行遗传到下一代。

算法源码

源码参考附录！