中山大学计算机学院

人工智能

本科生实验报告

（2022 学年春季学期）

课程名称：Artificial Intelligence

教学班级	信息与计算科学班	专业（方向）	信息与计算科学
学号	21311359	姓名	何凯迪

一、 实验题目

在 TSPLIB（http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/，多个地址有备份；其他网站还可以找到有趣的 art TSP 和 national TSP）中选一个大于 100 个城市数的 TSP 问题，使用**模拟退火**和**遗传算法**求解。

模拟退火算法：

●采用多种邻域操作的局部搜索 local search 策略求解；

●在局部搜索策略的基础上，加入模拟退火 simulated annealing 策略，并比较两者的效果；

●要求求得的解不要超过最优值的 10%，并能够提供可视化，观察路径的变化和交叉程度。

遗传算法：

●设计较好的交叉操作，并且引入多种局部搜索操作（可替换通常遗传算法的变异操作）

●和之前的模拟退火算法（采用相同的局部搜索操作）进行比较

●得出设计高效遗传算法的一些经验，并比较单点搜索和多点搜索的优缺点。

最优值查询：
http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/STSP.html

二、 实验内容

1. 算法原理

局部搜索算法：

局部搜索算法通常从一个初始解开始，然后根据一定的策略生成新的邻域解，从中选择最优的解作为下一个搜索点，直到找到一个局部最优解或到达预先设定的终止条件。在局部搜索过程中，搜索方向主要是基于局部梯度或类似方法的信息来进行的，因此其搜索范围比全局搜索算法要小得多。

模拟退火算法：

该算法基于固体物理中的“退火”过程，即将物质加热至高温，然后逐渐冷却，以获得热力学上稳定的状态。在模拟退火算法中，搜索过程类似于“退火”过程，即随着时间的推移，算法在搜索空间中逐渐减少搜索的范围，直到找到最优解。

模拟退火算法是通过引入随机性来避免陷入局部最优解的局限性，它可以接受一定的劣解以跳出局部最优，从而寻找更优的解。算法通过定义一个温度参数来控制随机性的程度，温度越高，算法接受劣解的概率越大，随机性也越大；温度越低，算法接受劣解的概率越小，随机性也越小。在退火过程中，随着时间的推移，温度会逐渐降低，算法会越来越倾向于接受更优的解，直到达到一定的停止准则为止。

遗传算法：

遗传算法的基本原理是通过模拟遗传和自然选择的过程来生成新的解，并不断地进化和改进这些解，最终找到最优解。遗传算法的搜索过程类似于自然界中的进化过程，它将候选解看作种群中的个体，利用选择、交叉和变异等操作来不断改进个体，并生成新的种群。

在遗传算法中，个体的适应度用于衡量其解决问题的能力，通常是将优化问题的目标函数作为适应度函数，将个体的解作为输入，计算出一个实数值作为个体的适应度。选择操作根据适应度大小来确定个体的选择概率，优秀的个体具有更高的选择概率，而较差的个体则具有较低的选择概率。交叉操作将两个个体的染色体进行随机交换，从而产生新的个体，这个过程类似于生物学中的杂交。变异操作则是在个体的染色体上进行一些随机的改变，以引入更多的多样性和随机性。

2. 伪代码

局部搜索算法：

```
Input: 初始解 x0, 步长 alpha, 终止条件 tol
Output: 最优解 x*
x = x0
while True:
    # 计算梯度
    grad = compute_gradient(x)
    # 计算下一个解
    x_new = x - alpha * grad
    if abs(x_new - x) < tol:
        break
    x = x_new
x* = x
```

模拟退火算法：

```
1. 初始化当前解 x 和温度 T
2. 重复执行以下步骤，直到达到停止条件：
   a. 从当前解 x 产生一个随机邻居解 x'
   b. 计算邻居解的代价函数差 Δf = f(x') - f(x)
   c. 如果 ΔE < 0，接受邻居解 x' 作为新的当前解 x
   d. 如果 ΔE >= 0，则以概率 p = exp(-ΔE/T) 接受邻居解 x'
   e. 降低温度 T
3. 返回当前解 x
```

遗传算法：

```
1. 初始化种群，包括 N 个个体（解），每个个体有 M 个基因（决策变量），每个基因有 K 种可能取值
2. 对每个个体进行适应度评估，即计算每个个体的适应度函数值 f(x)
3. 重复执行以下步骤，直到达到停止条件：
   a. 选择父代个体
      选择父代个体的方法可以是轮盘赌选择、锦标赛选择等，通常选择适应度较高的个体作为父代。
   b. 对父代个体进行交叉操作
      交叉操作将两个父代个体的基因进行随机组合，生成新的子代个体。交叉操作的概率可以根据问题特性进行调整。
   c. 对子代个体进行变异操作
      变异操作是为了增加种群的多样性，通常随机选择一个或多个基因，并以一定概率将其改变为另一种可能取值。变异操作的概率可以根据问题特性进行调整。
   d. 对新的种群进行适应度评估
      计算每个个体的适应度函数值，以便下一轮选择操作。
4. 返回适应度最高的个体作为最优解
```

3. 关键代码展示（带注释）

对于产生邻域的方法，我选用的三种，分别是交换两个城市点、将其中一段逆序以及随机插入一个节点，最后再随机挑选其中的一种进行扰动，从而增加随机性：

```
#方案 1 交换两个城市点
def nearby1(path):
    l=len(path)
    a1=random.randint(0,l-2)
    a2=random.randint(1,l-1)
    if a1!=a2:
        path[a1],path[a2]=path[a2],path[a1]
        tmp=path[:]
```

```
        path[a1],path[a2]=path[a2],path[a1] #还原列表
    else:
        tmp=path[:]
    return tmp
#方案 2 将一段逆序
def nearby2(path):
    l=len(path)
    a1,a2=random.sample(range(1,l-1),2)
    if a1>a2:
        a1,a2=a2,a1
    tmp=path[0:a1]+path[a1:a2][::-1]+path[a2:l]
    return tmp
#方案 3 随机插入一个节点
def nearby3(path):
    l=len(path)
    tmp=path[:]
    a=random.randint(1,l)
    tmp.remove(a)
    b=random.randint(0,l-1)
    tmp.insert(b,a)
    return tmp

#随机选择邻域方案
def nearby(path):
    a=random.randint(1,3)
    if a==1:
        return nearby1(path)
    elif a==2:
        return nearby2(path)
    elif a==3:
        return nearby3(path)
```

局部搜索算法：

```
# 局部搜索
count=300000
while count>0:
    count-=1
    print(count)
    sub=nearby(current)
    origin_eval=evaluate(current)
    new_eval=evaluate(sub)
    if new_eval<origin_eval:
        current=sub[:]
        change.append(new_eval)
```

模拟退火算法：

初始温度设置成了 100000，每次降温系数为 0.99，在每个温度下各进行 1000 次产生邻域的操作，截止温度设置为 0.001。

```
# 模拟退火
temperature=1000000
while temperature>0.001:
    temperature=0.99*temperature
    if temperature==0:
        break
    for i in range(1000):
        sub=nearby(current)
        origin_eval=evaluate(current)
        new_eval=evaluate(sub)
        change.append(origin_eval)
        if new_eval<origin_eval:
            current=sub
        # 以概率接收更差解
        else:
            k=min(1,math.exp((origin_eval-new_eval)/temperature))
            if k>random.random():
                current=sub
```

遗传算法：

```
# 遗传算法
while count<15000:
    new_population=[]
    #print(population)
    for m in range(0,10):
        fitness=[]
        # 赋予适应度
        for i in range(0,len(population)):
            #print(evaluate(population[i]))
            #距离总和越小权重越大
            fitness.append(1000000/evaluate(population[i]))

        #print(len(population))
        #print(len(fitness))
        # 随机父代
        p1=random.choices(population,fitness,k=1)
        p2=random.choices(population,fitness,k=1)
        # 保证当选到顺序交叉时也没有重复基因且包含父代的全部基因
```

```
        c1=crossover(p1[0],p2[0])
        c2=crossover(p2[0],p1[0])
        # 概率变异
        if random.random()<mutate:
            c1=nearby(c1)
            c2=nearby(c2)
        # 子代相同也变异
        if c1==c2:
            c1=nearby(c1)
            c2=nearby(c2)

        new_population.append(c1)
        new_population.append(c2)
    # 精英保留
    flag=-1
    max=float('inf')
    for j in range(0,len(population)):
        if evaluate(population[j])<max:
            max=evaluate(population[j])
            flag=j
    best=population[flag][:]
    print(count)
    change.append(max)
    population.clear()
    population=new_population[0:39]
    new_population.clear()
    population.append(best)
```

4. 创新点&优化（如果有）

1）对于扰动的产生方式进行了多次对比选择，最终决定随机挑选三种方法之一，结果也表明该方案是相对最优的，具体数据放在实验结果分析部分。

2）遗传算法中的初始群体改用了一半随机生成，另一半通过贪心算法生成，使得初始群体中有偏向性地包含较优个体。

3）遗传算法中的交叉部分选用了顺序交叉 OX、循环交叉 CX、部分映射交叉 PMX 及单点交叉 SPC，变异部分选用了与前面产生扰动同样的方案。

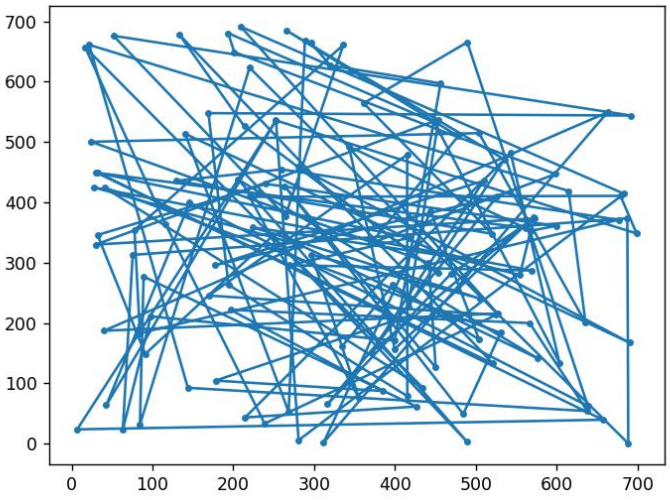
4）遗传算法中在产生子代时通过调换两个父代传参的顺序，保证了在选用到顺序交叉方案时也没有重复基因且包含父代的全部基因。

5）结果可视化了初始图、收敛曲线、结果图以及解的动态变化图。

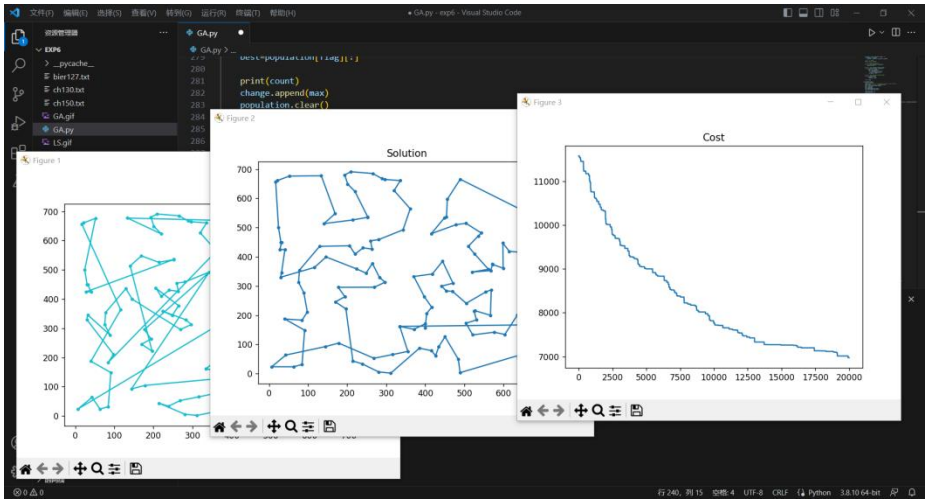
三、 实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）

运行程序首先会展示初始图

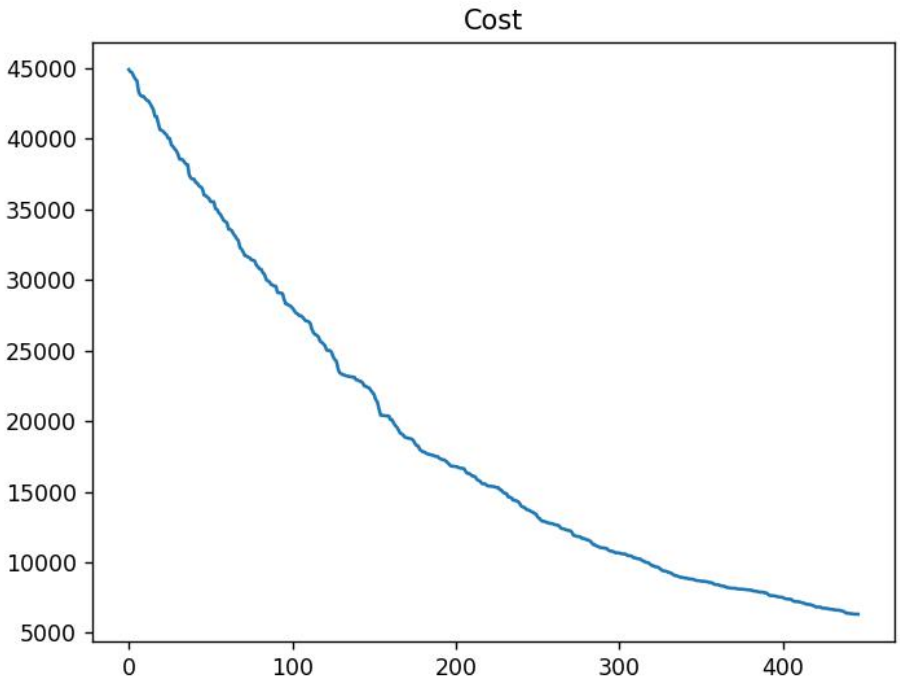


关闭该图后，则开始进行主要部分，运行完成过后会输出得到的解的数值，以及如下三张图，分别为解的动态生成图、最终结果图以及收敛曲线。

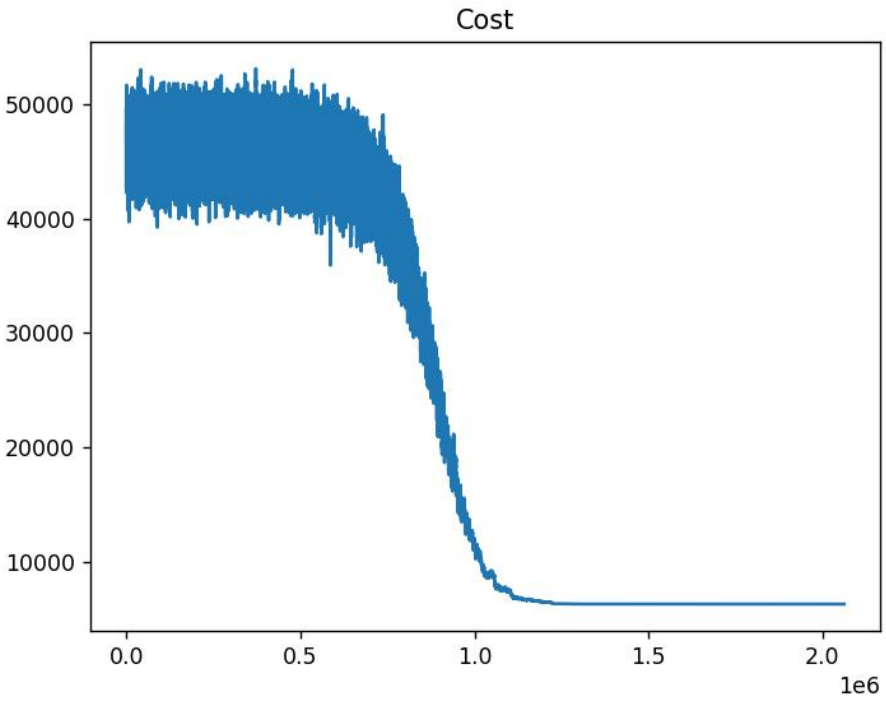


本次实验我选用了 ch130.tsp，经查询，最优解为 6110。经实验，局部搜索算法、模拟退火算法及遗传算法都有较好表现。

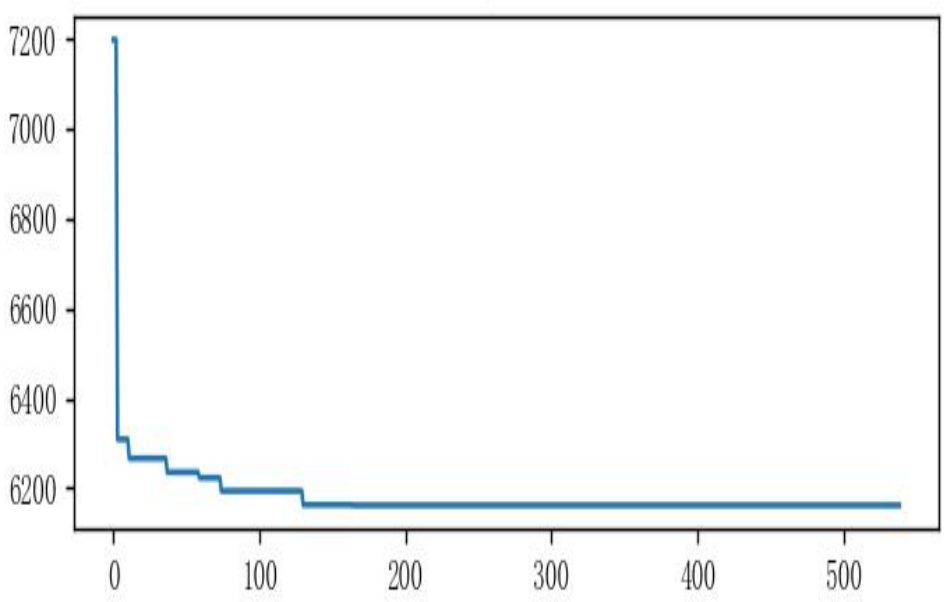
局部搜索算法得到解为 **6324.831036662459**



模拟退火算法得到解为 **6262.787970663097**



遗传算法得到的解为 **6161.882732014349**



求解结果均不超过标准值的 10%。

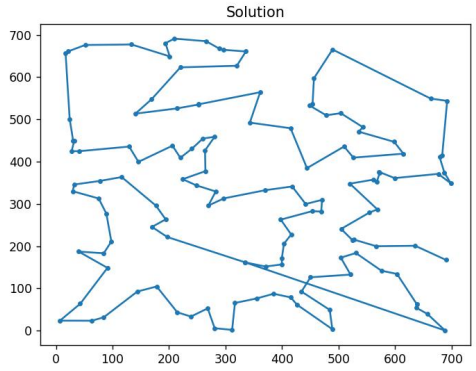
多种邻域产生方式比较（结果由局部搜索算法产生）：

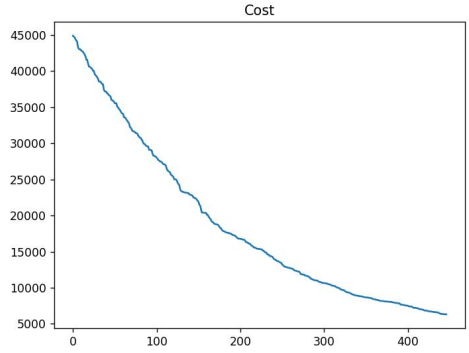
方案	Value1	Value2	Value3	Value4	Value5	均值
交换两点	11594.2564701749 93	11123.354613730 322	11594.517123068148	10359.56057682179	10520.963707168605	11038.5305
将一段逆序	7046.52337413216 6	7262.84332689628 7	7181.247755785107	7452.4031461629875	7006.027102768684	7189.808941
随机插入点	8444.22450138990 5	7661.85259686832	8271.369390152704	7972.680509706822	7577.524351991781	7985.530207
随机三选一	6398.62155356736 1	6464.48539329237 7	6516.578744849229	6430.4286807250655	6564.292282500142	6474.88133098683

不难看出，随机挑选三个方案中的一个为最佳。

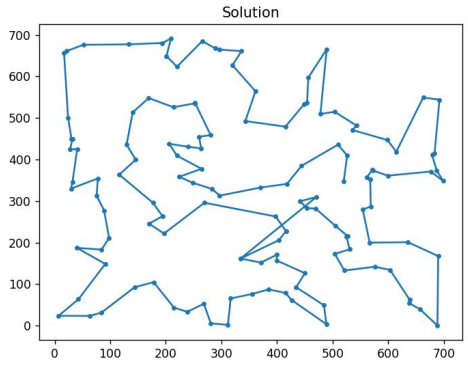
局部搜索算法和模拟退火算法比较：

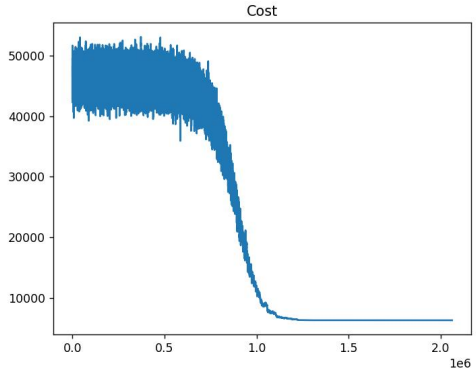
该为局部搜索算法结果：





该为模拟退火算法结果：





局部搜索算法通常是局部的、贪心的，只考虑当前状态下最优解，因此可能会陷入局部最优解而无法跳出。在我做的近 50 次实验中，最接近最优解的值为 6398.621553567361。

而模拟退火算法是一种全局优化算法，其基本思路是在一定的温度下随机跳跃搜索解空间，并接受一定概率的劣解，以防止陷入局部最优解。随着温度的下降，接受劣解的概率逐渐降低，搜索逐渐趋向于局部最优解，在我做的 5 次实验中，最接近最优解的值为 6262.787970663097，但是关于退火过程中的参数可能需要多次调试。

遗传算法和模拟退火算法比较：

遗传算法能够在大规模搜索空间中找到全局最优解或近似最优解，因为它具有遍历搜索空间的能力。而模拟退火算法适用于中等规模的搜索空间，其全局搜索能力相对较弱，在更大规模的问题中也很容易被卡在局部最优解中。

但是遗传算法的时间复杂度通常比模拟退火算法高，因为它需要更多的计算来执行基因交叉和变异操作。

同样的，这两种算法都需要大量的调参测试，本实验中对于遗传算法，我测试了变异概率为 0.01,0.05,0.1 和 0.3，最后选用 0.05 为变异概率。

设计高效遗传算法的经验：

1、选择适当的编码方式：选择适当的编码方式对于遗传算法的成功至关重要。编码方式应该能够正确地反映问题的特性，使得基因型和表现型之间的转换尽可能简单高效。
2、选择合适的遗传算法操作：遗传算法的操作包括选择、交叉和变异。选择操作用于选择适应度高的个体，交叉操作用于产生新的解决方案，变异操作用于保持种群的多样性。为了提高算法的效率，需要根据问题的特点选择适当的遗传算法操作。
3、设定合适的参数：遗传算法的性能很大程度上取决于参数的选择。例如，种群大小、交叉率、变异率等参数的设定都会影响算法的结果。合适的参数设置可以加快算法的收敛速度并提高算法的效率。
4、引入局部搜索：为了加速收敛和提高解决方案的质量，可以在遗传算法中引入局部搜索。局部搜索可以帮助算法跳出局部最优解并找到更优的全局最优解。
5、选择合适的停止准则：为了使遗传算法在合理的时间内停止并得到较优的解，需要选择合适的停止准则。常见的停止准则包括达到最大迭代次数、目标函数值的变化率达到一定阈值等。
6、选择合适的并行策略：对于一些计算量较大的问题，可以采用并行计算的方式来加速遗传算法的求解过程。选择合适的并行策略可以大大提高算法的效率。

单点搜索和多点搜索的优缺点：

单点搜索的优点是搜索速度较快，因为每次只需要比较当前解和一个邻域解的优劣。但是缺点是可能会陷入局部最优解，因为搜索空间受限于单个邻域解。

多点搜索的优点是搜索空间更大，可以更好地避免陷入局部最优解，因为搜索空间更广。但是缺点是搜索速度较慢，因为需要比较当前解和多个邻域解的优劣。

四、 参考资料

https://blog.csdn.net/xieju0605/article/details/109609413

https://blog.csdn.net/qq_34859482/article/details/80617391?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216836069701680018854566 7%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=168360697016800188545667&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-80617391-null-null.142^v86^insert_down1,239^v2^insert_chatgpt&utm_term=matplotlib&spm=1018.2226.3001.4187