智能计算技术大作业

实验报告

Intelligent computing technology

姓名：李广通

学号：919106840421

实验：仿生类的智能搜素算法实现旅行商问题

专业：计算机科学与技术专业

学院：计算机科学与工程学院

南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

目录

1.实验概述

1.1 实验目的

通过实验让学生掌握以遗传进化为代表的仿生类随机优化搜索算法，并根据实际问题灵活应用这些算法。

1.2 实验要求

C/C++或 Python

1.3 实验内容

随机生成 100 个结点（或更多结点）的图，在任意两结点之间赋予一条边，然后在这些边上赋予随机代价。最后从遗传算法、蚁群算法、粒子群算法、鱼群算法（上课介绍过），以及蛙跳算法、萤火虫算法，蝙蝠算法（课上没有介绍过）等仿生类算法中选择一个算法在这个图上求解从一个结点开始，经过且只经过每个结点一次，最后返回起点的最短路径的问题

2.实验源码

使用 python 语言实现，选择蚁群算法，所有实验代码已经做好注释说明，源文件打包在压缩包内，可执行验证

```
import math
import numpy as np
import matplotlib.pyplot as plt

# 随机生成图信息（节点数 100）
nCity = 100
City = np.random.uniform(-10, 10, [nCity, 2])
Dis = {}
for i in range(nCity):
    for j in range(nCity):
        if i > j:
            dis = ((City[i][0] - City[j][0]) ** 2 + (City[i][1] - City[j][1]) ** 2) **
0.5
            Dis[(i, j)] = dis
            Dis[(j, i)] = dis

Data_BestFit =[]    #用于保存每一代蚂蚁的最优适应度

#适应度计算函数 适应值= 节点数量 / 路径距离
def Cal_Fit(X):
    total_dis = Dis[(X[-1], X[0])]
    for i in range(nCity - 1):
        total_dis += Dis[(X[i], X[i + 1])]
    return nCity / total_dis

def ACA_TSP():
    nPop = 100          # 种群大小
    Maxit = 20          # 最大迭代次数
    Rou = 1.0           # 蒸发系数
    Rou_damp = 0.95     # 蒸发系数衰减度
    Rou_min = 0.1       # 最小蒸发系数
    alpha = 1           # 信息素重要程度
```

```
beta = 0.2            # 启发式信息重要程度
epsilon = 1e-5        # 初始信息素浓度
Phe = {}              # 保存信息素的字典
for key in Dis.keys():
    Phe[key] = epsilon

Fit = [0.0 for i in range(nPop)]
Best_Ant = None
Best_Fit = -math.inf

# 迭代求解
for j in range(Maxit):
    Ant = [[] for i in range(nPop)]
    #蚂蚁寻路
    for i in range(nPop):
        # 以第一个城市为起点，依次从剩下的城市中按照概率挑选目标
        # Open 保存已被选取的城市
        # Close 保存未被选取的城市
        Open = [0]
        Close = [i for i in range(1,nCity)]
        while Close:
            if len(Close) == 1:
                Open.append(Close.pop(0))
            else:
                P = np.zeros([len(Close)])
                for k in range(len(Close)):
                    P[k] = Phe[(Open[-1], Close[k])] ** alpha + Dis[(Open[-1], Close[k])] ** beta
                P = P / sum(P)
                next_index = np.random.choice(range(len(Close)), size=1, p=P)[0]
                Open.append(Close.pop(next_index))
        Ant[i] = Open.copy()

    #计算每只蚂蚁的路径适应值
    for i in range(nPop):
        Fit[i] = Cal_Fit(Ant[i])
        if Fit[i] > Best_Fit:
```

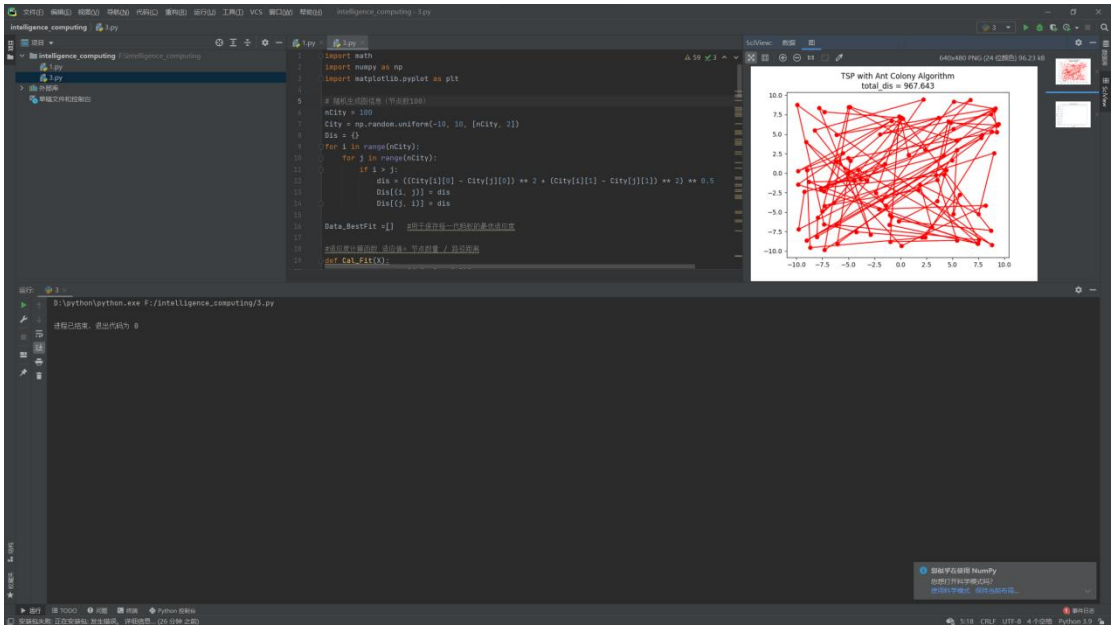
```
                    Best_Fit = Fit[i]
                    Best_Ant = Ant[i].copy()

            #根据蚂蚁路径更新信息素表
            for i in range(nPop):
                for k in range(nCity-1):
                    Phe[(Ant[i][k],Ant[i][k+1])] = Fit[i] + \
                                                    (1 - Rou) *
Phe[(Ant[i][k],Ant[i][k+1])]
            Rou = max(Rou * Rou_damp ,Rou_min)

            Data_BestFit.append(Best_Fit)

        return Best_Ant, Best_Fit

    # 绘制路径与迭代曲线
    def Draw_City(City, X ,Best_Fit):
        X = list(X)
        X.append(X[0])
        coor_x = []
        coor_y = []
        for i in X:
            i = int(i)
            coor_x.append(City[i][0])
            coor_y.append(City[i][1])

        plt.plot(coor_x, coor_y, 'r-o')
        plt.title('TSP with Ant Colony Algorithm\n'+'total_dis =
'+str(round(Best_Fit,3)))
        plt.show()

        plt.plot(range(len(Data_BestFit)), Data_BestFit)
        plt.title('Iteration_BestFit')
        plt.xlabel('iteration')
        plt.ylabel('fitness')
        plt.show()
```

```
if __name__ == '__main__':
    Best_X, Best_Fit = ACA_TSP()
    Draw_City(City, Best_X, (Best_Fit/nCity)**-1)
```

3.实验结果演示及说明

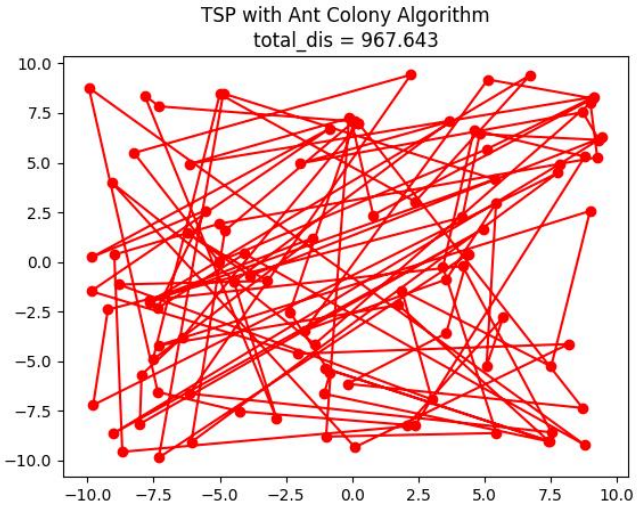
设计权值时开始全部使用正数，但权值上限过高以及随机性可能会带来越界的影响（前期实验缺陷），于是加入负权，每个边权在-10~10 之间随机，即可解决上述问题

使用 pycharm 运行结果如下所示（使用 matplotlib 进行可视化设计，绘制路径和迭代收益的曲线，通过实验对超参数进行了调整，最高迭代次数降低，使得运行速度适当加快）



实验结果主要由两张图组成

第一张是路径图（对蚁群算法得到的 TSP 问题解进行可视化输出，路径总长是 967.643）



第二章是迭代曲线，可以看到蚁群算法的收敛速度非常快