群智能算法：蚁群算法解决“旅行商”问题

# 1实验目的

1. 了解群智能算法的产生背景；
2. 了解群智能算法与进化算法的异同；
3. 了解蚁群算法基本原理；
4. 了解蚁群算法各个参数的影响。

# 2实验要求

本次试验后，要求学生能：

1. 理解蚁群算法的基本原理和实现流程；
2. 理解各个参数对蚁群算法结果的影响；
3. 学会用蚁群算法解决复杂问题；
4. 编写python程序，用蚁群算法解决“旅行商”问题；
5. 比较遗传算法、粒子群优化算法、蚁群算法的优劣，总结出他们各自的特点。

# 3实验原理

## 3.1实验介绍

本实验是运用遗传算法解决“旅行商”问题：

一位旅行商要去往n个城市经商，恰好访问每个城市一次，并最终回到出发城市。设旅行商从第i个城市前往第j个城市的开销为COSTij，周游完全部n个城市并回到原点的总开销为COST总，市编写程序求COST总的最小值。

## 3.2蚁群算法简介

蚁群算法(AG)是一种模拟蚂蚁觅食行为的模拟优化算法，它是由意大利学者Dorigo M等人于1991年首先提出，并首先使用在解决TSP（旅行商问题）上。

蚁群算法的基本原理:

（1）蚂蚁在路径上释放信息素。

（2）碰到还没走过的路口，就随机挑选一条路走。同时释放信息素。

（3）信息素浓度与路径长度成反比。后来的蚂蚁再次碰到该路口时，有更大的概率选择信息素浓度较高路径。

（4）最优路径上的信息素浓度越来越大。

（5）最终蚁群找到最优寻食路径。

## 3.3蚁群算法

（1）蚂蚁状态转移规则

蚂蚁按照一个由信息素浓度（pheromone）和局部启发信息（eta = 1/d）共同决定的概率，从还未经过的城市中选择周游的下一个目的地。

这个概率的计算公式如下：

公式1

（2）信息素浓度更新规则

信息素浓度的变化主要由两个原因导致，一个是信息素的挥发ρ，一个是蚂蚁再次经过路径是释放的新信息素Δτ。

所以t+1时刻，信息素更新:

公式2

而蚂蚁信息素的更新策略又可以分为蚂蚁圈模型、蚂蚁数量模型、蚂蚁密度模型这三种模型。其中蚂蚁全模型利用的是整体信息，即在所有蚂蚁完成一个循环后，再更新所有路径上的信息。后两种利用局部信息，每走一步都要更新一次信息素浓度。

本实验选用的是蚂蚁圈模型，信息素的更新公式为：

公式3-1

公式3-2

# 4实验步骤

## 4.1安装依赖

本实验需要用到python的扩展程序库Numpy进行数值计算，如果没有安装过这个库，使用国内的清华pip源安装，命令如下：

pip3 install numpy scipy matplotlib -i <https://pypi.tuna.tsinghua.edu.cn/simple>

导入依赖库，其中copy是python内建模块，本实验要用到copy.deepcopy()进行深度复制，无需额外下载，可以直接import。

import numpy as np

import copy

import random

4.2实现蚁群优化算法

### 4.2.1创建蚂蚁类

（1）蚂蚁类包含的成员函数如下：

①初始化蚂蚁init\_Ant(self)

给每个蚂蚁随机一个城市序号，作为它的出发点。每个蚂蚁都要建立一个禁忌表（记录还没访问过的城市），避免在一次循环中多次访问同一个城市（否则不符合旅行商问题的要求）。

②让蚂蚁移动到下一个城市move\_to\_next(self)

根据蚁群算法中的随机比例规则，由两个城市之间的信息素浓度和能见度（两个城市距离的倒数）计算出从当前城市选择其他可选城市（在unvisited表中的城市）的概率。

然后用轮盘赌选方式按概率选择下一个城市，并移动。

函数返回0表示城市已经全部访问完，返回1表示还要继续游历。

③计算蚂蚁走过的总路径get\_distance(self)

④蚂蚁周游城市travel(self)

蚂蚁开始周游城市，从随机城市出发，按照由信息素、能见度共同决定的概率选择下一个城市，最后回到原点。

（2）详细代码如下

class Ant:

def \_\_init\_\_(self, index):

self.index = index # 蚂蚁的序号

self.init\_Ant()

# 初始化蚂蚁的各个数据结构，随机选择一个出发点

def init\_Ant(self):

self.path = [] # 当前路径

self.total\_distance = 0

self.unvisited = np.ones(city\_size, int) # 还没被访问的城市(1代表未访问，0代表已访问)

first\_city = random.randint(0, city\_size - 1) # 随机一个起始城市序号

# 更新信息

self.current\_city = first\_city

self.path.append(first\_city)

self.unvisited[first\_city] = 0

# 选择下一个城市

# return 0 ：全部访问完

# return 1：还要继续游历

def move\_to\_next(self):

# 第一步：计算当前城市到其他城市的概率

p\_numerator = np.zeros(city\_size) # 选择概率的分子

p\_denominator = 0.0 # 选择概率的分母（分子之和）

trans\_p = np.zeros(city\_size) # 概率矩阵

for i in range(city\_size):

# 如果第i个城市已经访问过，则再访问i的概率为0

if self.unvisited[i] == 0:

p\_numerator[i] = 0

else:

# 随机比例规则公式

p\_numerator[i] = np.dot(pheromone\_tb[self.current\_city][i], a) \* np.dot(eta[self.current\_city][i], b)

p\_denominator += p\_numerator[i] # 将分子累加

# 分母等于0说明每个城市都已经访问过了

if p\_denominator != 0:

trans\_p = p\_numerator / p\_denominator

else:

return 0

# 第二步：根据概率轮盘赌选下一个城市

next\_city = np.random.choice(np.array(range(city\_size)), size=1, replace=True, p=trans\_p)[0]

# 第三步：移动到下一个城市

self.path.append(next\_city)

self.unvisited[next\_city] = 0

self.current\_city = next\_city

return 1

# 计算路径总距离

def get\_distance(self):

for i in range(-1, len(self.path) - 1):

self.total\_distance += map[self.path[i]][self.path[i + 1]]

def travel(self):

# 初始化蚂蚁，选择出发点

self.init\_Ant()

flag = 1

# 蚂蚁开始周游

while flag == 1:

flag = self.move\_to\_next()

# 计算路径，输出结果

self.get\_distance()

4.2.2更新信息素

先按蚂蚁圈系统的模型，计算信息素的增量，利用的是整体信息，等所有蚂蚁访问完一圈后，才更新所有路径上的信息。

若第k只蚂蚁在本次循环中从x走到y，则x到y上信息素的增量为 = Q/Lk，Lk为第k只蚂蚁本次循环中所走的路径总长度。

最终的信息素浓度为按（1-r）衰减后的信息素加上本次循环后增加的信息素。

def update\_pheromone():

global ant\_pop, pheromone\_tb

# 信息素增加矩阵

pheromone\_increase = np.zeros((city\_size, city\_size))

for ant in ant\_pop:

path\_temp = copy.deepcopy(ant.path)

for i in range(-1, len(path\_temp) - 1):

start = path\_temp[i]

end = path\_temp[i + 1]

pheromone\_increase[start][end] += Q / ant.total\_distance

# 信息量更新公式

pheromone\_tb = (1 - r) \* pheromone\_tb + pheromone\_increase

4.2.3迭代搜索

初始化蚂蚁种群，迭代搜索，每次让蚂蚁按概率周游城市，更新最佳路径信息。在所有蚂蚁周游完之后更新信息素浓度。

def searchTSP():

global best\_distance, best\_path

# 迭代搜索

i = time

while best\_distance > 21 and i > 0:

i -= 1

for ant in ant\_pop:

ant.travel()

if ant.total\_distance < best\_distance:

best\_distance = ant.total\_distance

best\_path = copy.deepcopy(ant.path)

# 本代的所有蚂蚁都周游完，再更新信息素

update\_pheromone()

"""print("本次迭代的结果：")

print(best\_path)

print(best\_distance)

print("搜索完成，最佳路径及其长度为：")

print(best\_distance)

print(best\_path)

"""

return best\_distance, time - i + 1

## 4.3实验与分析

更改r、a、b、pop\_size等参数，进行多次实验，观察各个参数对算法执行效率、搜索准确性的影响。这里设置了7种不同的参数组合方式，对每种参数组合方式进行50次实验。

最后统计最大迭代次数为100时，不同参数组合方式**各自**对应的50次实验中，最好解的距离、最坏解的距离、最好解次数、迭代次数、迭代次数的中位数、搜索到解的距离平均值、搜索到的解的距离中位数等数学特征，进行比较。

主函数代码如下。

if \_\_name\_\_ == '\_\_main\_\_':

# 几个常量

a = 2 # 认知因子

b = 2 # 社会因子

r = 0.5 # 信息素挥发速度

Q = 21

city\_size = 10

ant\_size = 9

# 地图矩阵(对角线上实际为0，为了计算eta矩阵方便，这里对角线设置为1)

map = np.array([

[1, 4, 11, 7, 15, 9, 1, 8, 10, 5],

[4, 1, 12, 3, 6, 5, 10, 7, 4, 7],

[11, 12, 1, 14, 7, 2, 3, 1, 2, 11],

[7, 3, 14, 1, 6, 13, 8, 15, 7, 2],

[15, 6, 7, 6, 1, 6, 5, 2, 16, 1],

[9, 5, 2, 13, 6, 1, 2, 18, 10, 9],

[1, 10, 3, 8, 5, 2, 1, 14, 5, 3],

[8, 7, 1, 15, 2, 18, 14, 1, 2, 8],

[10, 4, 2, 7, 16, 10, 5, 2, 1, 14],

[5, 7, 11, 2, 1, 9, 3, 8, 14, 1]

], dtype='int').reshape([city\_size, city\_size])

eta = 1 / map # 能见度矩阵

time = 100 # 最大迭代次数

parameters = [[2, 2, 0.5, 21, 7], [2, 50, 0.5, 21, 7], [50, 2, 0.5, 21, 7], [2, 2, 0.95, 21, 7],

[2, 2, 0.05, 21, 7], [2, 2, 0.5, 500, 7], [2, 2, 0.5, 21, 3]]

for parameter in parameters:

a = parameter[0]

b = parameter[1]

r = parameter[2]

Q = parameter[3]

ant\_size = parameter[4]

best\_distances = []

record = []

best\_count = 0

# 对每种参数组合执行50次实验

for j in range(50):

# 最佳路径及其距离

best\_path = []

best\_distance = 80

# 初始化信息素矩阵，全是为1组成的矩阵

pheromone\_tb = np.ones((city\_size, city\_size))

# 初始化蚂蚁群

ant\_pop = [Ant(index) for index in range(ant\_size)]

# 蚁群搜索

dis, num = searchTSP()

best\_distances.append(dis)

record.append(num)

if dis == 21:

best\_count += 1

print("参数为（顺序是a,b,r,Q,ant\_size)：")

print(parameter)

print("最佳距离与迭代次数为：")

print(best\_distances)

print(record)

print("得到最优解的次数：%d / 50" % best\_count)

print("平均值")

print(np.mean(best\_distances))

print(np.mean(record))

print("中位数")

print(np.median(best\_distances))

print(np.median(record))

# 5实验结果

## 5.1单次实验

将a设为2，b设为2，r设为0.5，Q设为21，ant\_size设为7，最大迭代次数设为100，用蚁群算法求解旅行商问题，结果如下：

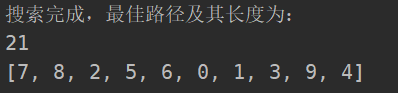


图1 单次实验得到的解路径和长度

## 5.2多次实验探究参数影响

对7种不同的参数组合方式，分别进行50次蚁群算法优化实验，每次实验迭代次数最高为100次。

（1）部分实验结果如下：

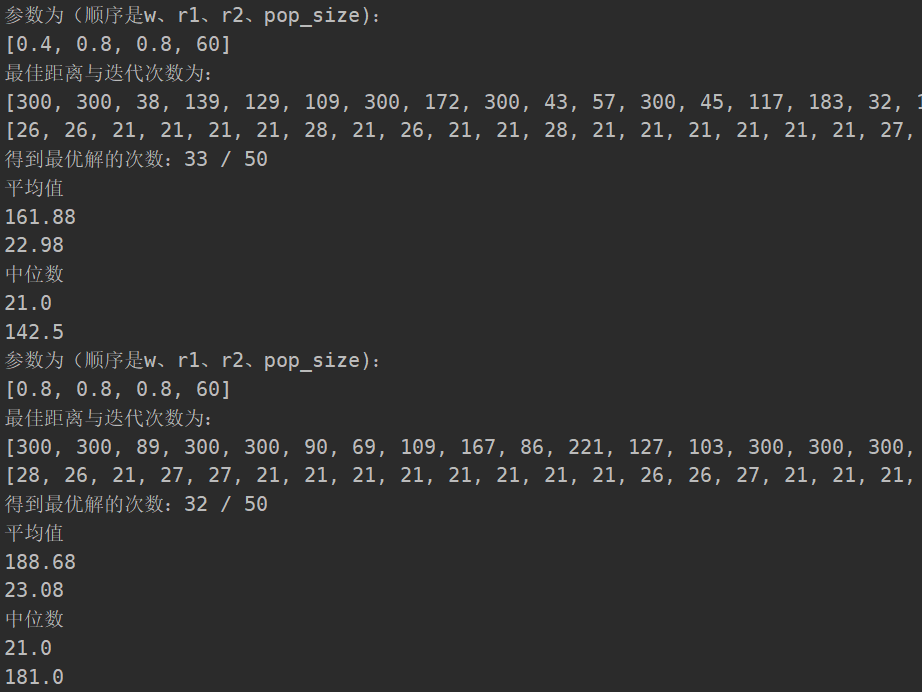


图2 调整参数的部分实验结果

（2）实验结果统计如下表：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 参数  (a,b,r,Q,ant\_size) | 最好解 | 最坏解 | 最好解次数 | 平均迭代次数 | 平均解 | 迭代次数中位数 | 解中位数 |
| (2,2,0.5,21,7) | 21 | 21 | 50 | 9.12 | 21 | 7.5 | 21 |
| (2,50,0.5,21,7) | 21 | 21 | 50 | 9.71 | 21 | 8.5 | 21 |
| (50,2,0.5,21,7) | 21 | 26 | 49 | 12.34 | 21.1 | 9 | 21 |
| (2,2,0.95,21,7) | 21 | 30 | 24 | 58.94 | 24.14 | 100 | 26 |
| (2,2,0.05,21,7) | 21 | 21 | 50 | 10.12 | 21 | 7.5 | 21 |
| (2,2,0.5,500,7) | 21 | 27 | 47 | 17.26 | 21.24 | 9 | 21 |
| (2,2,0.5,21,3) | 21 | 30 | 25 | 61.85 | 24.22 | 95.5 | 23.5 |

表1 蚁群优化算法7\*50次实验运行结果

鼓励同学们尝试设置更多的参数组合进行多次实验，深入探究如果设置参数可以使算法得到更优秀的性能。

同学们还可以尝试使用不同的信息素更新模型进行更多探索。