南開大學
Nankai University

人 工 智 能 技 术 实 验
实 验 报 告

实验名称：基于蚁群算法的旅行商问题

姓　　名：刘崇轩

学　　号：2312801

专　　业：智能科学与技术

人工智能学院

2024 年 9 月

一、 问题简述

旅行商问题：设有 N 个互相可直达的城市，给定每对城市之间的距离，某推销商准备从其中的 A 城出发，周游各城市一遍，最后又回到 A 城，要求访问每一座城市并回到起始城市的最短回路。这是非常经典的组合优化问题中的 NP 难问题（多项式复杂程度的非确定性问题），通过常规的暴力枚举，遍历等方法难以计算出最优解，因此，本实验使用蚁群算法进行最优解的计算。

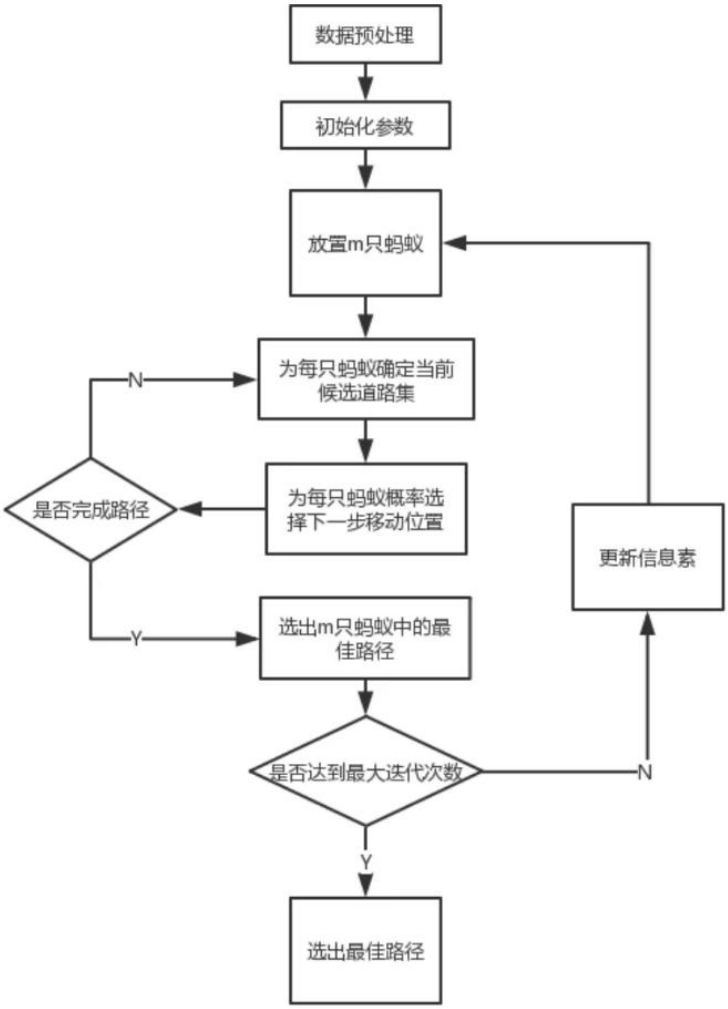
蚁群算法(Ant Colony Optimization, ACO)是一种基于群体智能的优化算法，由 Marco Dorigo 于 1992 年提出。蚁群算法模拟了自然界中蚂蚁群体寻找最短路径的行为，利用蚂蚁在行走过程中留下的信息素来引导后续蚂蚁选择路径。通过不断迭代和信息素更新，算法逐渐找到问题的最优解或近似最优解。

蚁群算法的核心思想是利用信息素(Pheromone)的正反馈机制来逐步优化解。蚂蚁在行走过程中会释放信息素，信息素的浓度与路径的质量（通常是路径的长度或花费的时间）有关。路径越短，信息素越浓，后续蚂蚁选择该路径的概率就越高。这种机制可以通过多次迭代不断强化优质路径，逐渐逼近最优解。

对于本实验，求解我国 34 个城市的旅行商问题，给定 34 个城市的名称和坐标数据，求遍历一次所有城市回到起点期间，所经过路程的较小值。

基本思想：

1. 根据具体问题设置多只蚂蚁，分头并行搜索。
2. 每只蚂蚁完成一次周游后，在行进的路上释放信息素，信息素量与解的质量成正比。
3. 蚂蚁路径的选择根据信息素强度大小（初始信息素量设为相等），同时考虑两点之间的距离，采用随机的局部搜索策略。这使得距离较短的边，其上的信息素量较大，后来的蚂蚁选择该边的概率也较大。
4. 每只蚂蚁只能走合法路线（经过每个城市 1 次且仅 1 次）。
5. 所有蚂蚁都搜索完一次就是迭代一次，每迭代一次就对所有的边做一次信息素更新，然后新的蚂蚁进行新一轮搜索。
6. 更新信息素包括原有信息素的蒸发和经过的路径上信息素的增加。
7. 达到预定的迭代步数，或出现停滞现象（所有蚂蚁都选择同样的路径，解不再变化），则算法结束，以当前最优解作为问题的最优解。



二、 实验目的

1. 了解旅行商问题的基本概念，理解旅行商问题的求解难度；
2. 了解蚁群算法的基本原理及实现流程；
3. 能够利用蚁群算法解决旅行商问题；
4. 进一步理解可视化设计。（选做）

三、 实验内容

1. 实现基于蚁群算法的 34 个城市的旅行商问题，求解访问每一座城市一次并最终回到起始城市的最短回路；

2. 对比蚁群算法下不同蚂蚁数量 m、信息素挥发系数 ρ 、信息素总量 Q、信息素因子 α 、启发函数因子 β 等参数下，遗蚁群算法对问题求解的效果影响并分析原因。

3. 进行蚁群算法可视化，展示搜索过程中每次迭代过程中种群最优路径走势的变化和迭代最优结果。（选做）

四、 编译环境

1. 操作系统：Windows 11
2. Python：3.12
3. 主要第三方库：matplotlib、math、random
4. 在 PyCharm Community Edition 下编译

五、 实验步骤（包括重要程序代码解释）

1. 数据准备与距离矩阵计算

定义了包含 34 个城市的列表 cities，每个元素为 (城市名, 经度, 纬度)。使用 calc_dist 函数计算欧氏距离，并预先生成 34×34 的距离矩阵 dist_matrix，以减少重复计算。

```
# 计算两点欧氏距离
def calc_dist(c1, c2):
    return math.hypot(c1[1] - c2[1], c1[2] - c2[2])
# 预计算距离矩阵 (二维列表)
dist_matrix = [[calc_dist(cities[i], cities[j]) for j in range(N)] for i in range(N)]
# 初始化信息素矩阵 (所有边初始为 1.0)
pheromone = [[1.0 for _ in range(N)] for _ in range(N)]
```

2.参数设置

根据代码设定，使用以下基准参数：

```
ANT_COUNT = 50  # 蚂蚁数量 m
MAX_ITER = 500  # 迭代次数
ALPHA = 1.0  # 信息素重要度
BETA = 2.0  # 启发因子重要度
RHO = 0.1  # 挥发系数
Q = 100.0  # 信息素总量常数
```

3. 核心算法流程

主循环模拟了迭代过程：

✧ 构建路径：每只蚂蚁从随机起点出发，利用 random.choices 根据计算出的权重（概率）选择下一个城市，直到遍历所有城市并回到起点。

```
    for ant in range(ANT_COUNT):
        # 1. 随机起点
        curr = random.randint(0, N - 1)
        path = [curr]
        visited = {curr}
        path_len = 0

        # 2. 构建路径
        while len(path) < N:
            # 筛选未访问城市
            candidates = [i for i in range(N) if i not in visited]
```

```
            # 计算选择概率权重: (信息素^alpha) * ((1/距离)^beta)
            weights = []
            for next_city in candidates:
                tau = pheromone[curr][next_city]
                dist = dist_matrix[curr][next_city]
                if dist==0:
                    eta = 1.0 / (dist + 1e-10)  # 避免除 0
                else:
                    eta = 1.0 / dist
                weights.append((tau ** ALPHA) * (eta ** BETA))

            # 轮盘赌选择
            next_node = random.choices(candidates, weights=weights, k=1)[0]

            path.append(next_node)
            visited.add(next_node)
            path_len += dist_matrix[curr][next_node]
            curr = next_node

        # 加上回到起点的距离
        path_len += dist_matrix[path[-1]][path[0]]
        round_paths.append((path, path_len))
```

✧ 更新最优解：如果当前蚂蚁的路径长度小于全局历史最优 best_dist，则更新 best_path。

```
        # 更新全局最优
        if path_len < best_dist:
            best_dist = path_len
            best_path = path

    history_best.append(best_dist)
```

✧ 信息素更新：所有边乘以 (1 - RHO) 进行挥发，并且遍历本轮所有蚂蚁的路径，根据 Q / path_len 增加经过边上的信息素。

```
# 3. 信息素更新
    # 挥发
    for i in range(N):
        for j in range(N):
            pheromone[i][j] *= (1 - RHO)

    # 增强 - 遍历本轮所有蚂蚁
    for path, path_len in round_paths:
        delta = Q / path_len
```

```
        for i in range(N):
            u, v = path[i], path[(i + 1) % N]  # (i+1)%N 自动处理回到起点
            pheromone[u][v] += delta
            pheromone[v][u] += delta  # 无向图对称更新
```

✧ 结果可视化：使用 matplotlib 绘制两张图表

收敛曲线：展示 history_best 随迭代次数的变化；

路径图：根据最优路径的索引，提取经纬度坐标并连线，标注城市名称。

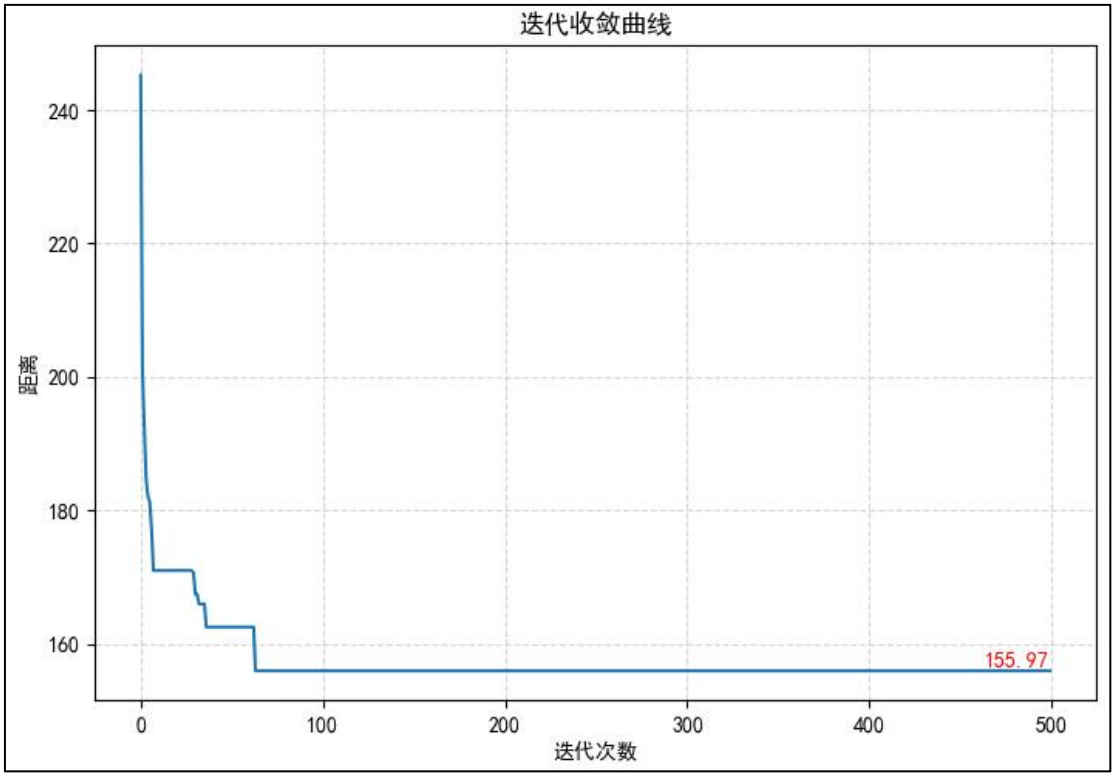
```
# 图 1: 收敛曲线
plt.figure(figsize=(12, 8))
plt.plot(history_best)
plt.title("迭代收敛曲线")
plt.xlabel("迭代次数")
plt.ylabel("距离")
plt.grid(True, linestyle='--', alpha=0.5)
    # 在最后一点标记数值
final_cost = history_best[-1]
plt.text(len(history_best) - 1, final_cost, f"{final_cost:.2f}",
         color='red', ha='right', va='bottom', fontsize=10, fontweight='bold')
plt.show()

# 图 2: 路径图
plt.figure(figsize=(12, 8))
x = [cities[i][1] for i in best_path + [best_path[0]]]  # 闭环坐标 X
y = [cities[i][2] for i in best_path + [best_path[0]]]  # 闭环坐标 Y
plt.plot(x, y, 'o-', color='blue', alpha=0.6, linewidth=1, markersize=4)
for name, lx, ly in cities:  # 标注城市名
    plt.text(lx+0.3, ly+0.3, name, fontsize=8)
plt.title(f"规划路线 (距离: {best_dist:.2f})")
plt.xlabel("经度")
plt.ylabel("纬度")
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
```

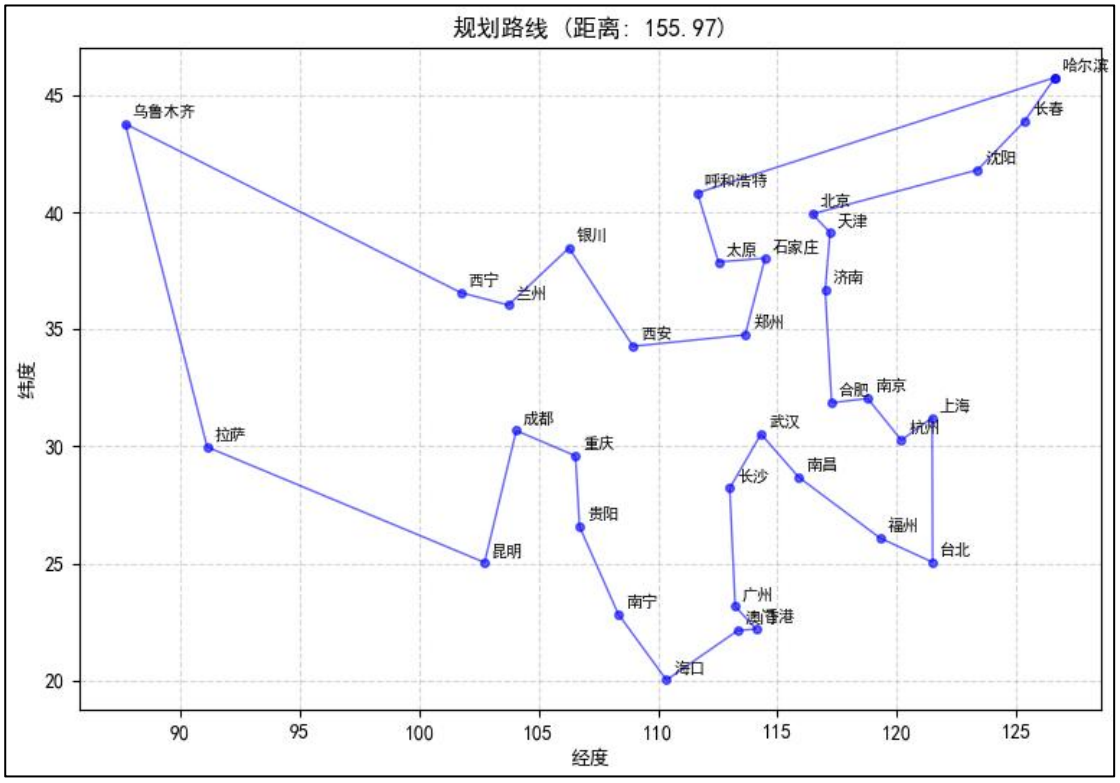
六、 实验结果

最优路径: 哈尔滨 -> 长春 -> 沈阳 -> 北京 -> 天津 -> 济南 -> 合肥 -> 南京 -> 杭州 -> 上海 -> 台北 -> 福州 -> 南昌 -> 武汉 -> 长沙 -> 广州 -> 香港 -> 澳门 -> 海口 -> 南宁 -> 贵阳 -> 重庆 -> 成都 -> 昆明 -> 拉萨 -> 乌鲁木齐 -> 西宁 -> 兰州 -> 银川 -> 西安 -> 郑州 -> 石家庄 -> 太原 -> 呼和浩特

总距离: 155.97

可视化：


迭代收敛曲线
距离
迭代次数
155.97
240
220
200
180
160
0
100
200
300
400
500


规划路线（距离：155.97）
纬度
经度
乌鲁木齐
哈尔滨
长春
沈阳
呼和浩特
北京
天津
银川
太原
石家庄
西宁
兰州
济南
西安
郑州
合肥
南京
上海
成都
武汉
杭州
拉萨
重庆
长沙
南昌
贵阳
福州
台北
昆明
南宁
广州
澳门
香港
海口
45
40
35
30
25
20
90
95
100
105
110
115
120
125

实验组	蚂蚁数量	迭代次数	信息素重要度α	启发因子重要度β	挥发系数ρ	信息素总量常数 Q	最优距离
基准组	50	500	1.0	2.0	0.1	100.0	155.97
对照组 A	5	500	1.0	2.0	0.1	100.0	157.08
对照组 B	50	50	1.0	2.0	0.1	100.0	158.56
对照组 C	50	500	1.0	0	0.1	100.0	325.92
对照组 D	50	500	0	2.0	0.1	100.0	196.35
对照组 E	50	500	1.0	2.0	0.9	100.0	156.40
对照组 F	50	500	1.0	2.0	0.1	1.0	158.00

七、 分析总结

（1）基于实验结果，各参数对算法性能的影响分析如下：

1、启发因子（决定性影响）

✧ 结果：对照组 C 最优距离 325.92，远超基准组 155.97，是表现最差的一组。

✧ 分析：β=0 意味着蚂蚁完全忽略城市间距离，算法退化为纯随机搜索。这表明距离启发信息是引导蚂蚁找到短路径的最关键因素。

2、信息素重要度（显著影响）

✧ 结果：对照组 D 最优距离 196.35，明显劣于基准组。

✧ 分析：α=0 意味着蚂蚁忽略群体经验，算法退化为贪婪算法。虽比纯随机好，但因缺乏正反馈机制，极易陷入局部最优，无法达到全局最优。

3、种群规模与迭代次数（微调影响）

✧ 结果：对照组 A (蚂蚁=5) 和 对照组 B (迭代=50) 的结果分别为 157.08 和 158.56，略差于基准组。

✧ 分析：减少蚂蚁数量或迭代次数会导致搜索不充分（欠拟合），但并未导致算法失效。说明算法在早期收敛较快，增加资源主要用于后期的精细化寻优。

4、挥发系数 ρ 与信息素总量 Q（鲁棒性）

✧ 结果：对照组 E 和对照组 F 结果均在 156-158 之间，与基准组差异极小。

✧ 分析：在种群数量足够（50 只）的情况下，算法对挥发速度和信息素强度的变化具有较强的鲁棒性，高挥发并未显著破坏记忆。

（2）总结：β 和 α 是核心参数，缺失任意一个都会导致算法性能剧烈下降；而其他参数主要影响收敛精度和速度，在一定范围内调整对结果影响较小。