高级算法设计与分析课程报告

2023202210126 于清

无人机配送路径规划问题

为了解决无人机配送路径规划问题,可以使用启发式算法,例如**遗传算法**(Genetic Algorithm, GA）**或蚁群算法**（Ant Colony Optimization, ACO），因为这些算法很适合解决复杂的路径优化问题，如无人机配送路径规划。

本报告使用蚁群算法来解决该问题。蚁群算法通过模拟蚂蚁觅食行为，能有效解决无人机配送路径规划问题。在处理动态订单和优先级别时，具有很好的灵活性和适应性。通过合理的参数设置和多次迭代，可以找到总配送路径最短的解。

选择蚁群算法的理由

1. **适合路径优化问题**:蚁群算法是一种基于自然界蚂蚁觅食行为的启发式算法，特别适用于求解旅行商问题（TSP）等路径优化问题。
2. **动态性**:可以动态地添加新的订单和调整路径规划，适应实时变化的订单需求。
3. **全局搜索能力**：通过信息素的更新和积累，能够有效探索全局最优解。

具体步骤

1. 参数和距离矩阵初始化

- **参数设定**：设定蚁群算法的参数，如蚂蚁数量、信息素重要程度（α）、启发式信息重要程度（β）、信息素挥发系数（ρ）等。
  距离矩阵：计算配送中心与各个卸货点之间的距离矩阵。

2. 状态空间表示

- **节点**：包括所有配送中心和卸货点。
- **路径**:从配送中心出发,经过一个或多个卸货点,再返回配送中心的路径。

3. 订单的优先级处理

- 订单分类：将订单分为一般（3 小时内配送到即可）、较紧急（1.5 小时内配送到）和紧急（0.5 小时内配送到）三个优先级;

- 订单时间窗口：为每个优先级订单设置对应的配送时间窗口。

4. 蚂蚁构造解

- **初始点选择**：每只蚂蚁从随机选择的配送中心出发。
- **路径选择规则**：蚂蚁在每一步选择下一个节点作为卸货点，根据以下公式计算转移概率：

$$P_{ij} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{k \in \text{allowed}} [\tau_{ik}]^{\alpha} \cdot [\eta_{ik}]^{\beta}}$$

其中，τ_{ij}是路径上的信息素强度，η_{ij}是启发式信息（如路径的倒数距离），α和β分别是信息素和启发式信息的重要程度。

5. 路径更新与信息素更新

- **路径长度计算**：每只蚂蚁完成其路径后，计算其总路径长度。

代码实现

- **信息素更新**：根据路径长度更新信息素，路径越短的信息素增加越多：

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \sum_{\text{all ants}} \Delta\tau_{ij}$$

其中，ρ是信息素挥发系数，$\Delta\tau_{ij}$是路径上的增量信息素。

6. 迭代

- **多次迭代**：重复蚂蚁构造解和信息素更新步骤，经过多次迭代后，逐步收敛到最优或次优解。

实现步骤

1. **数据准备**：获取配送中心、卸货点及其位置数据，以及订单生成的规则和频率。
2. **算法初始化**：设定蚁群算法的参数并初始化距离矩阵。
3. **订单处理**：根据订单的优先级别分类，并生成相应的订单队列。
4. **路径规划**：每隔 t 分钟运行蚁群算法，为当前订单生成最佳配送路径。
5. **路径执行与更新**：执行生成的路径，同时更新信息素并处理新生成的订单。

伪代码示例

```
# 初始化参数
alpha = 1.0
beta = 2.0
rho = 0.5
num_ants = 20
num_iterations = 100
```

```
7.
8.# 初始化距离矩阵和信息素矩阵
9.distance matrix = compute distance matrix(centers, delivery
   points)
10.pheromone matrix = initialize pheromone matrix(num nodes)
11.
12.for iteration in range(num iterations):
13.    all routes = []
14.    for ant in range(num ants):
15.        route = []
16.        current center = random.choice(centers)
17.        visited = set()
18.        while len(route) < max order size:
19.            next point = select next point(current center,
   visited, pheromone matrix, distance matrix, alpha, beta)
20.            route.append(next point)
21.            visited.add(next point)
22.            current center = next point
23.        route.append(route[0])  # 回到配送中心
24.        all routes.append(route)
25.
26.    # 更新信息素矩阵
27.    pheromone matrix = update pheromone matrix(pheromone ma
   trix, all routes, distance matrix, rho)
28.
29.# 选择最佳路径
30.best_route = select_best_route(all_routes)
```

代码实现

定义初始化参数、配送中心、卸货点位置

```python
import numpy as np
import random
import matplotlib.pyplot as plt

# 初始化参数
alpha = 1.0  # 信息素重要程度
beta = 2.0  # 启发式信息重要程度
rho = 0.5  # 信息素挥发系数
num_ants = 10  # 蚂蚁数量
num_iterations = 50  # 迭代次数
t = 10  # 时间离散化步长（分钟）
n = 5  # 无人机一次最多携带的物品数量
max_distance = 20  # 无人机一次飞行最远路程（公里）
speed = 60  # 无人机速度（公里/小时）

# 假设配送中心和卸货点的位置
centers = [(0, 0), (10, 10), (12, 6), (3, 1), (5, 5)]  # 4个配送中心
delivery_points = [(2, 3),(4, 1), (5, 2), (6, 7),(7, 1), (8, 4),(9, 9), (10, 15),(11, 12), (12, 5), (13, 12),]  # 卸货点位置
```

距离矩阵计算，需要避免自身举例为 0 的非法情况。

```python
# 计算距离矩阵
def compute_distance_matrix(centers, delivery_points):
    points = centers + delivery_points
    num_points = len(points)
    distance_matrix = np.zeros((num_points, num_points))
    for i in range(num_points):
        for j in range(num_points):
            if i != j:
                distance_matrix[i][j] = np.sqrt((points[i][0] - points[j][0])**2 + (points[i][1] - points[j][1])**2)
            else:
                distance_matrix[i][j] = np.inf  # 避免自身距离为零的情况
    return distance_matrix
```

初始化信息素矩阵为全 1 矩阵

```python
# 初始化信息素矩阵
def initialize_pheromone_matrix(num_nodes):
    return np.ones((num_nodes, num_nodes))
```

选择下一个配送点：在选择下一个点时，`select_next_point` 确保不会选择当前的中心点。如果所有点都已访问过，则随机选择未访问过的点或返回中心点。 确保每次迭代时能够选择有效的下一个点，如果没有可选点则跳过当前蚂蚁的路线选择；在添加回到配送中心的点之前，检查当前路线如果为空，则跳过该路线。

```python
# 选择下一个配送点
def select_next_point(current_point, visited, pheromone_matrix, distance_matrix, alpha, beta):
    """
    根据当前节点、已访问节点、信息素矩阵和距离矩阵，选择下一个配送点。

    参数:
    current_point (int): 当前所在的节点索引。
    visited (set): 已访问节点的集合。
    pheromone_matrix (numpy.ndarray): 信息素矩阵。
    distance_matrix (numpy.ndarray): 距离矩阵。
    alpha (float): 信息素的重要程度。
    beta (float): 启发式信息的重要程度。

    返回:
    int: 下一个配送点的索引。
    """
    num_points = len(distance_matrix)
    probabilities = np.zeros(num_points)
    total_prob = 0.0
    for i in range(num_points):
        if i not in visited:
            if distance_matrix[current_point][i] > 0:
                probabilities[i] = (pheromone_matrix[current_point][i] ** alpha) * ((1.0 / distance_matrix[current_point][i]) ** beta)
            else:
                probabilities[i] = 0
            total_prob += probabilities[i]

    if total_prob == 0:
        return None

    probabilities /= total_prob
    next_point = np.random.choice(range(num_points), p=probabilities)
    return next_point if next_point not in visited else None
```

更新信息素矩阵：

```python
# 更新信息素矩阵
def update_pheromone_matrix(pheromone_matrix, all_routes, distance_matrix, rho):
    """
    根据所有蚂蚁的路径更新信息素矩阵。

    参数:
    pheromone_matrix (numpy.ndarray): 原始信息素矩阵。
    all_routes (list): 所有蚂蚁的路径列表。
    distance_matrix (numpy.ndarray): 距离矩阵。
    rho (float): 信息素挥发系数。

    返回:
    numpy.ndarray: 更新后的信息素矩阵。
    """
    num_nodes = len(pheromone_matrix)
    delta_pheromone = np.zeros((num_nodes, num_nodes))
    for route in all_routes:
        route_length = sum(distance_matrix[route[i]][route[i+1]] for i in range(len(route) - 1))
        if route_length > 0:
            for i in range(len(route) - 1):
                delta_pheromone[route[i]][route[i+1]] += 1.0 / route_length
    pheromone_matrix = (1 - rho) * pheromone_matrix + delta_pheromone
    return pheromone_matrix
```

选择最佳的路径：

```python
# 选择最佳路径
def select_best_route(all_routes, distance_matrix):
    """
    从所有路径中选择最短的路径。

    参数:
    all_routes (list): 所有蚂蚁的路径列表。
    distance_matrix (numpy.ndarray): 距离矩阵。

    返回:
    list: 最短路径。
    """
    best_route = None
    best_length = float('inf')
    for route in all_routes:
        route_length = sum(distance_matrix[route[i]][route[i+1]] for i in range(len(route) - 1))
        if route_length < best_length:
            best_length = route_length
            best_route = route
    return best_route
```

模拟生成订单，每个配送点随机生成 0 到 5 个订单，并分配优先级。

```python
# 生成订单
def generate_orders(delivery_points, max_orders=5):
    """
    模拟生成订单，每个配送点随机生成0到5个订单，并分配优先级。

    参数:
    delivery_points (list): 配送点列表。
    priorities (list): 订单优先级列表。

    返回:
    list: 生成的订单列表，格式为 (配送点，优先级)。
    """
    orders = []
    for i, point in enumerate(delivery_points):
        num_orders = random.randint(0, max_orders)
        for _ in range(num_orders):
            priority = random.choice(['一般', '较紧急', '紧急'])
            orders.append((i + len(centers), priority))  # 订单格式：(卸货点索引，优先级)
    return orders
```

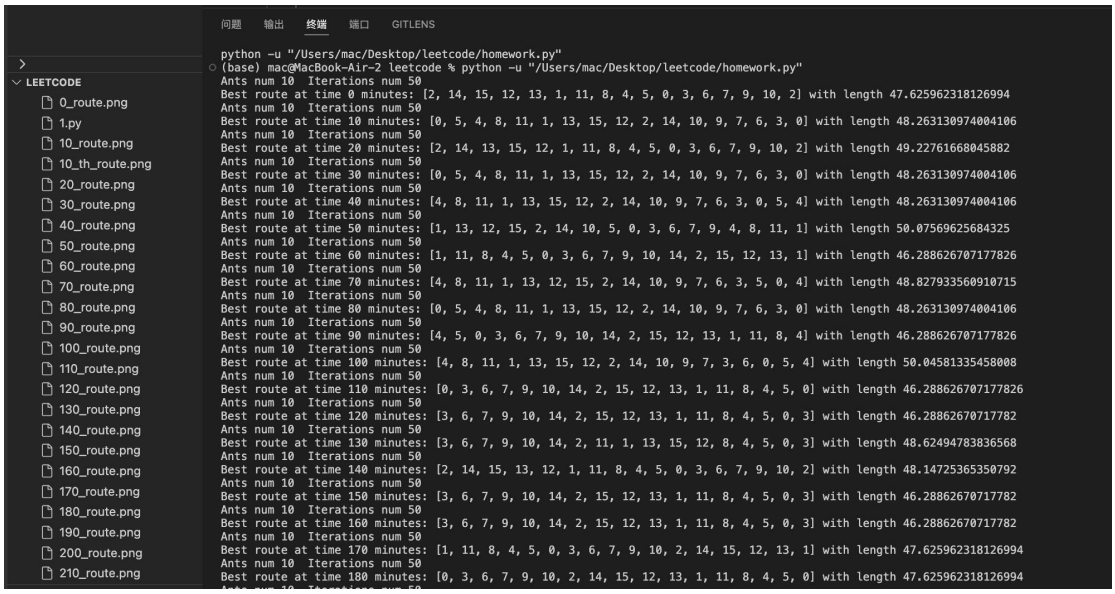
路径规划并分配无人机：

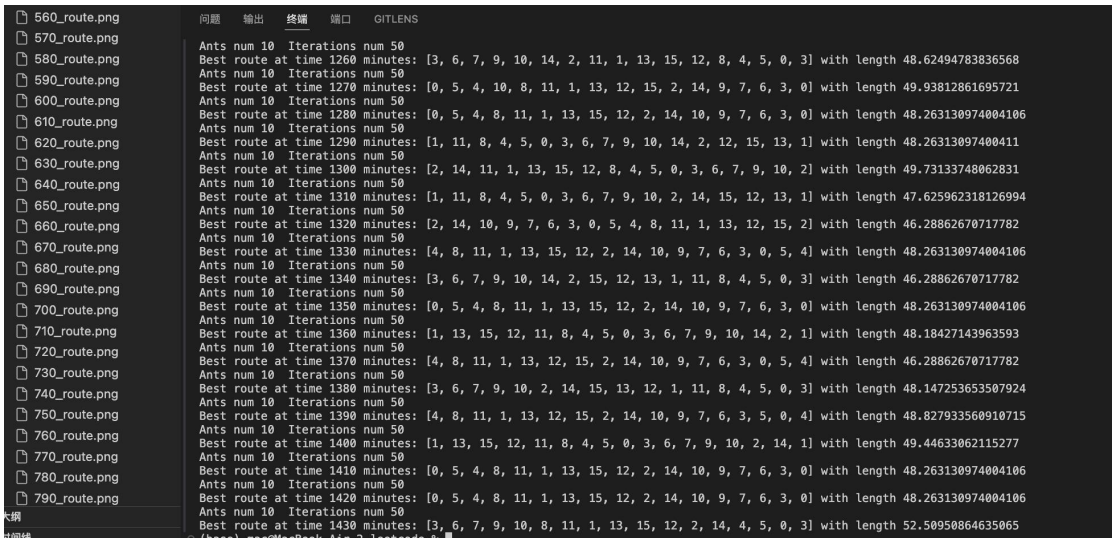
```python
# 模拟订单生成和路径规划
def simulate_delivery(centers, delivery_points, num_ants, num_iterations, alpha, beta, rho, t, simulation_time):
    total_time = 0
    all_orders = []

    while total_time < simulation_time:
        orders = generate_orders(delivery_points)
        all_orders.extend(orders)

        # 按优先级排序订单
        all_orders.sort(key=lambda x: ('一般', '较紧急', '紧急').index(x[1]))

        # 规划路径并分配无人机
        best_route, best_length = ant_colony_optimization(centers, delivery_points, num_ants, num_iterations, alpha, beta, rho)
        print(f"Ants num {num_ants}  Iterations num {num_iterations}")
        print(f"Best route at time {total_time} minutes: {best_route} with length {best_length}")
        plot_route(best_route, centers, delivery_points, total_time)

        # 模拟配送
        all_orders = []

        total_time += t
```

路径规划算法主函数：

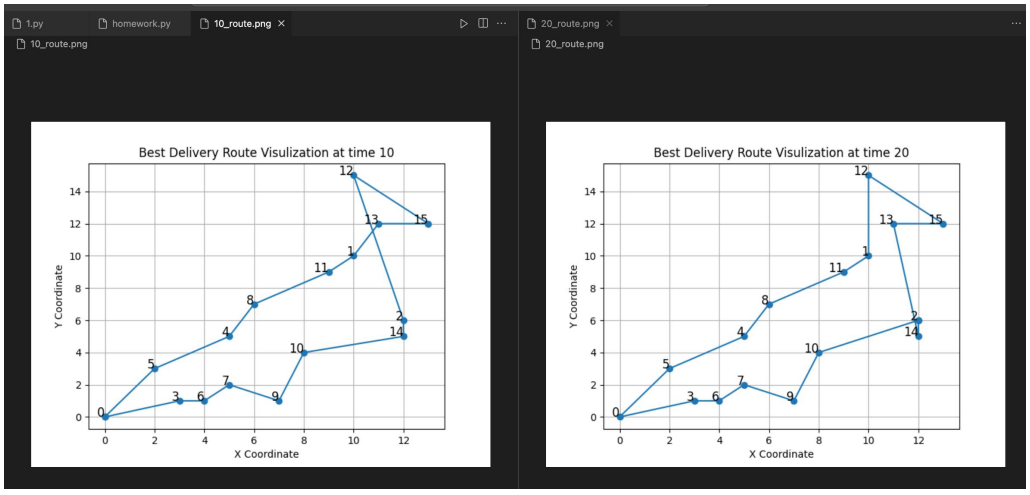
```python
# 主路径规划算法
def ant_colony_optimization(centers, delivery_points, num_ants, num_iterations, alpha, beta, rho):
    distance_matrix = compute_distance_matrix(centers, delivery_points)
    num_nodes = len(centers) + len(delivery_points)
    pheromone_matrix = initialize_pheromone_matrix(num_nodes)

    best_route = None
    best_length = float('inf')

    for iteration in range(num_iterations):
        all_routes = []

        for ant in range(num_ants):
            visited = set()
            current_point = random.choice(range(len(centers)))  # Start from a random center
            route = [current_point]
            visited.add(current_point)

            while len(visited) < num_nodes:
                next_point = select_next_point(current_point, visited, pheromone_matrix, distance_matrix, alpha, beta)
                if next_point is None:
                    break
                route.append(next_point)
                visited.add(next_point)
                current_point = next_point

            route.append(route[0])  # Return to the starting center
            all_routes.append(route)

        pheromone_matrix = update_pheromone_matrix(pheromone_matrix, all_routes, distance_matrix, rho)
        current_best_route = select_best_route(all_routes, distance_matrix)
        current_best_length = sum(distance_matrix[current_best_route[i]][current_best_route[i+1]] for i in range(len(current_best_route) - 1))

        if current_best_length < best_length:
            best_length = current_best_length
            best_route = current_best_route

    return best_route, best_length
```

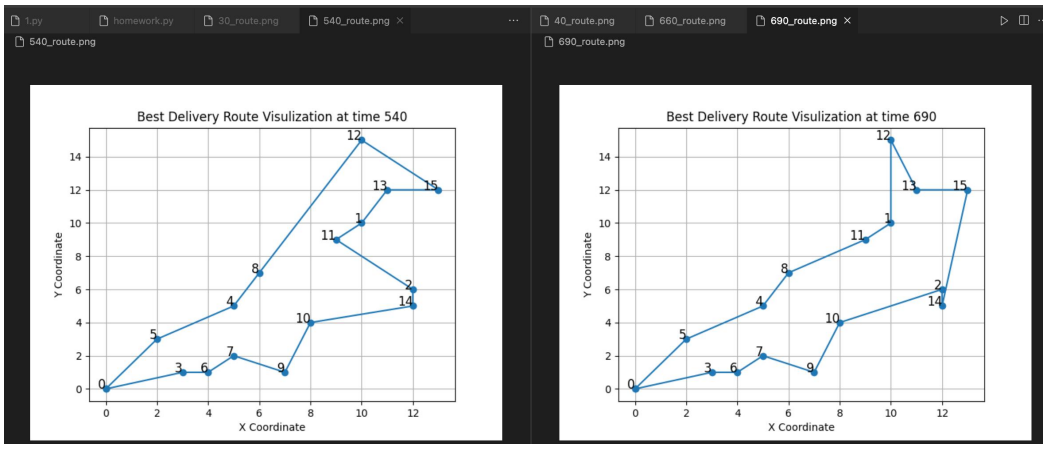
最优路径输出

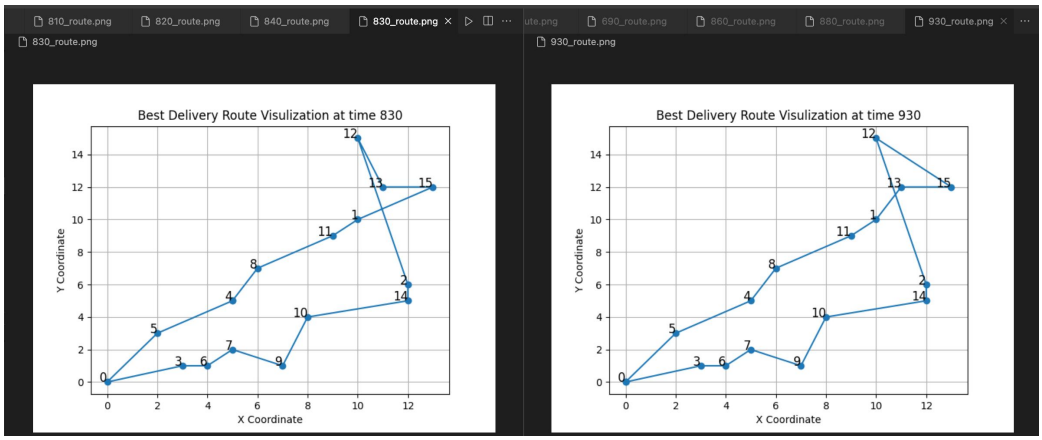


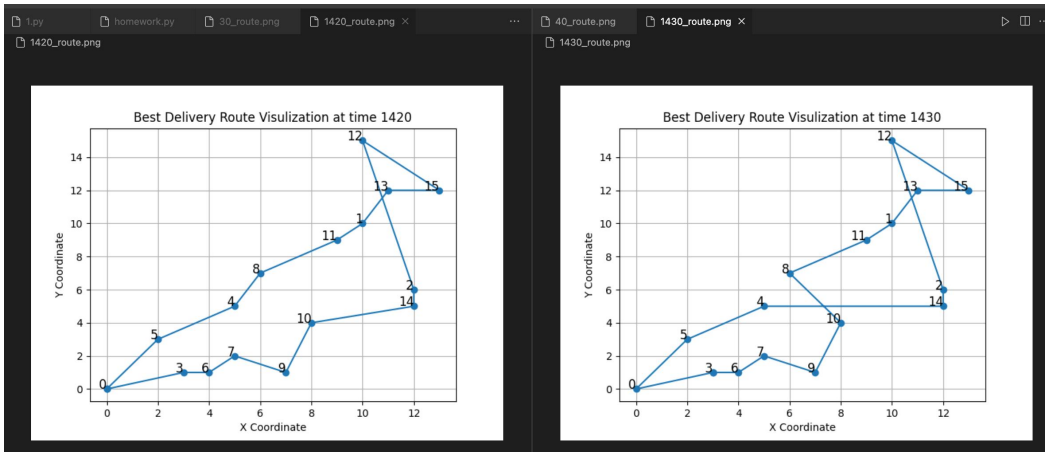

560_route.png
570_route.png
580_route.png
590_route.png
600_route.png
610_route.png
620_route.png
630_route.png
640_route.png
650_route.png
660_route.png
670_route.png
680_route.png
690_route.png
700_route.png
710_route.png
720_route.png
730_route.png
740_route.png
750_route.png
760_route.png
770_route.png
780_route.png
790_route.png
问题
输出
终端
端口
GITLENS
Ants num 10 Iterations num 50
Best route at time 1260 minutes: [3, 6, 7, 9, 10, 14, 2, 11, 1, 13, 15, 12, 8, 4, 5, 0, 3] with length 48.62494783836568
Ants num 10 Iterations num 50
Best route at time 1270 minutes: [0, 5, 4, 10, 8, 11, 1, 13, 12, 15, 2, 14, 9, 7, 6, 3, 0] with length 49.93812861695721
Ants num 10 Iterations num 50
Best route at time 1280 minutes: [0, 5, 4, 8, 11, 1, 13, 15, 12, 2, 14, 10, 9, 7, 6, 3, 0] with length 48.263130974004106
Ants num 10 Iterations num 50
Best route at time 1290 minutes: [1, 11, 8, 4, 5, 0, 3, 6, 7, 9, 10, 14, 2, 12, 15, 13, 1] with length 48.263130974004411
Ants num 10 Iterations num 50
Best route at time 1300 minutes: [2, 14, 11, 1, 13, 15, 12, 8, 4, 5, 0, 3, 6, 7, 9, 10, 2] with length 49.73133748062831
Ants num 10 Iterations num 50
Best route at time 1310 minutes: [1, 11, 8, 4, 5, 0, 3, 6, 7, 9, 10, 2, 14, 15, 12, 13, 1] with length 47.625962318126994
Ants num 10 Iterations num 50
Best route at time 1320 minutes: [2, 14, 10, 9, 7, 6, 3, 0, 5, 4, 8, 11, 1, 13, 12, 15, 2] with length 46.28862670717782
Ants num 10 Iterations num 50
Best route at time 1330 minutes: [4, 8, 11, 1, 13, 15, 12, 2, 14, 10, 9, 7, 6, 3, 0, 5, 4] with length 48.263130974004106
Ants num 10 Iterations num 50
Best route at time 1340 minutes: [3, 6, 7, 9, 10, 14, 2, 15, 12, 13, 1, 11, 8, 4, 5, 0, 3] with length 46.28862670717782
Ants num 10 Iterations num 50
Best route at time 1350 minutes: [0, 5, 4, 8, 11, 1, 13, 15, 12, 2, 14, 10, 9, 7, 6, 3, 0] with length 48.263130974004106
Ants num 10 Iterations num 50
Best route at time 1360 minutes: [1, 13, 15, 12, 11, 8, 4, 5, 0, 3, 6, 7, 9, 10, 14, 2, 1] with length 48.18427143963593
Ants num 10 Iterations num 50
Best route at time 1370 minutes: [4, 8, 11, 1, 13, 12, 15, 2, 14, 10, 9, 7, 6, 3, 0, 5, 4] with length 46.28862670717782
Ants num 10 Iterations num 50
Best route at time 1380 minutes: [3, 6, 7, 9, 10, 2, 14, 15, 13, 12, 1, 11, 8, 4, 5, 0, 3] with length 48.147253653507924
Ants num 10 Iterations num 50
Best route at time 1390 minutes: [4, 8, 11, 1, 13, 12, 15, 2, 14, 10, 9, 7, 6, 3, 5, 0, 4] with length 48.827933560910715
Ants num 10 Iterations num 50
Best route at time 1400 minutes: [1, 13, 15, 12, 11, 8, 4, 5, 0, 3, 6, 7, 9, 10, 2, 14, 1] with length 49.44633062115277
Ants num 10 Iterations num 50
Best route at time 1410 minutes: [0, 5, 4, 8, 11, 1, 13, 15, 12, 2, 14, 10, 9, 7, 6, 3, 0] with length 48.263130974004106
Ants num 10 Iterations num 50
Best route at time 1420 minutes: [0, 5, 4, 8, 11, 1, 13, 15, 12, 2, 14, 10, 9, 7, 6, 3, 0] with length 48.263130974004106
Ants num 10 Iterations num 50
Best route at time 1430 minutes: [3, 6, 7, 9, 10, 8, 11, 1, 13, 15, 12, 2, 14, 4, 5, 0, 3] with length 52.50950864635065

路径可视化

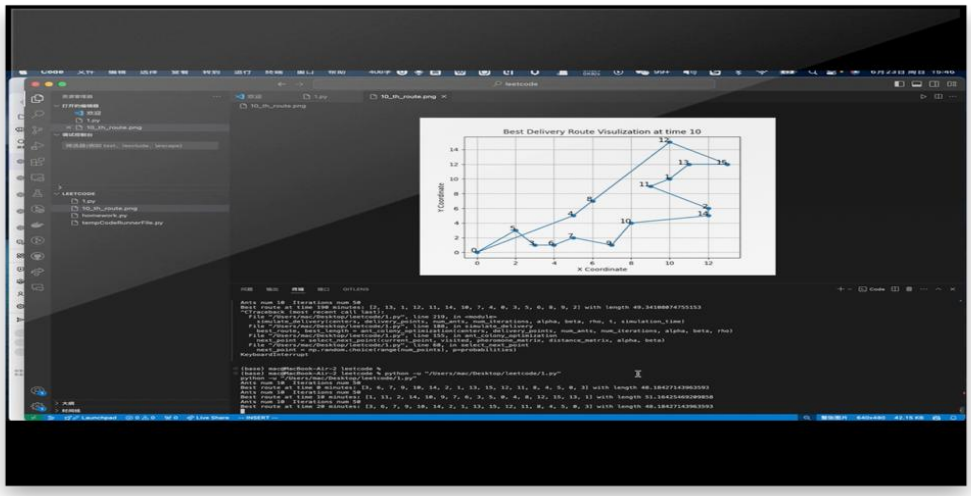

1.py
homework.py
10_route.png
10_route.png
20_route.png
20_route.png
Best Delivery Route Visulization at time 10
Best Delivery Route Visulization at time 20
Y Coordinate
X Coordinate


1.py
homework.py
30_route.png
540_route.png
540_route.png
40_route.png
660_route.png
690_route.png
690_route.png
Best Delivery Route Visulization at time 540
Best Delivery Route Visulization at time 690
Y Coordinate
X Coordinate





录屏视频演示

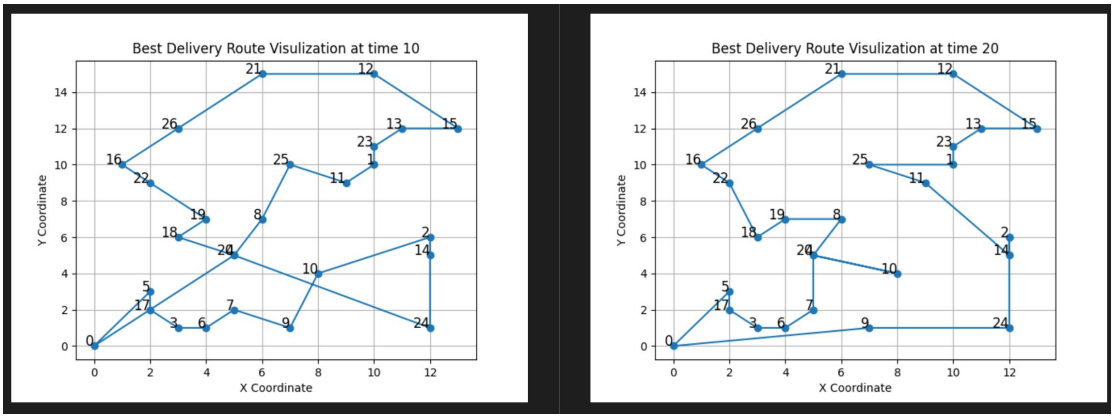


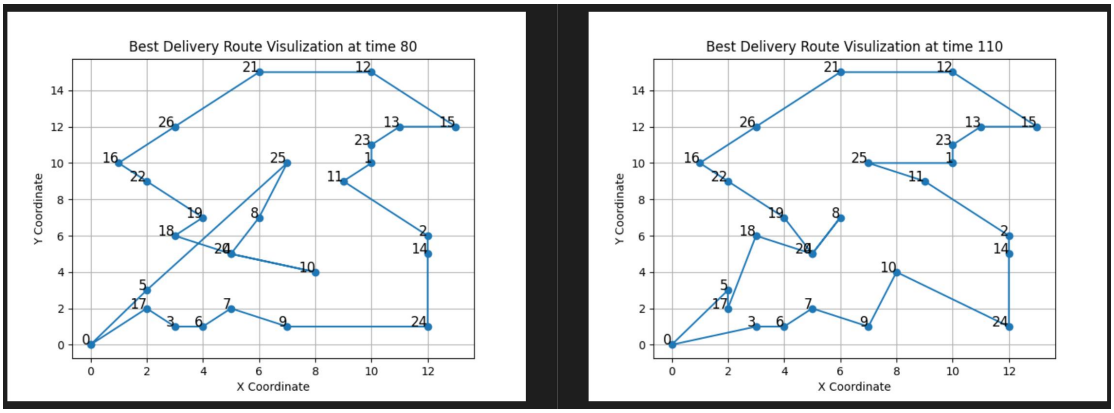
还可以尝试其他蚁群参数设置和配送点位置
进一步实验信息素挥发系数为 0.7，蚁群数量 20，增加一倍卸货点位置：

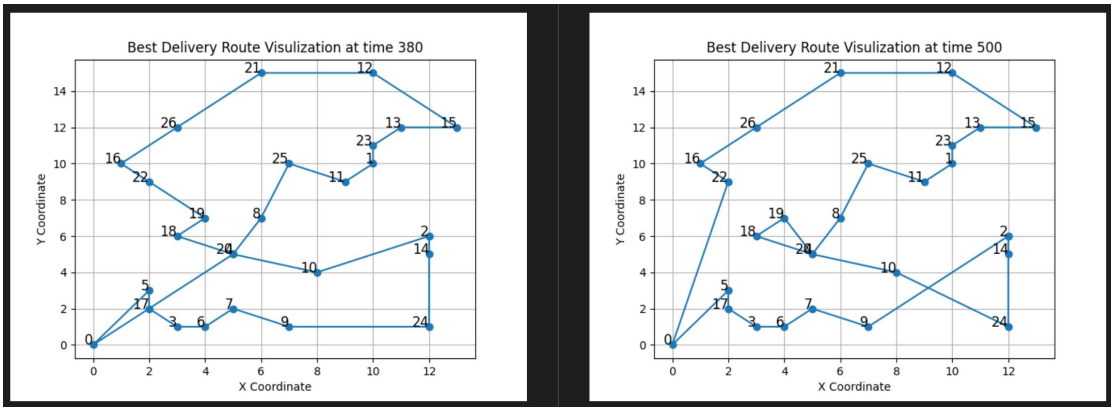
```python
# 初始化参数
alpha = 1.0  # 信息素重要程度
beta = 2.0  # 启发式信息重要程度
rho = 0.7  # 信息素挥发系数
num_ants = 10  # 蚂蚁数量
num_iterations = 50  # 迭代次数
t = 10  # 时间离散化步长（分钟）
n = 5  # 无人机一次最多携带的物品数量
max_distance = 20  # 无人机一次飞行最远路程（公里）
speed = 60  # 无人机速度（公里/小时）

# 假设配送中心和卸货点的位置
centers = [(0, 0), (10, 10), (12, 6), (3, 1), (5, 5)]  # 5个配送中心
delivery_points = [(2, 3),(4, 1), (5, 2), (6, 7),(7, 1), (8, 4),(9, 9), (10, 15),(11, 12), (12, 5), (13, 12),
                   (1, 10),(2, 2), (3, 6), (4, 7),(5, 5), (6, 15),(2, 9), (10, 11),(12, 1), (7, 10), (3, 12),]  # 卸货点位置

```

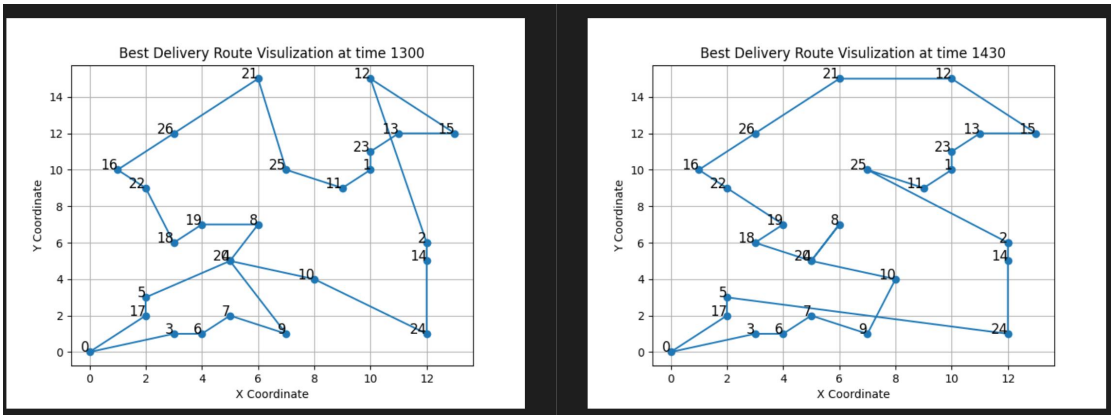
不同时刻下最优路径结果：

```
(base) mac@MacBook-Air-2 leetcode % python -u "/Users/mac/Desktop/leetcode/homework.py"
Ants num 10  Iterations num 50
Best route at time 0 minutes: [1, 23, 13, 15, 12, 21, 26, 16, 22, 20, 10, 2, 14, 24, 9, 7, 6, 3, 17, 5, 0, 19, 18, 4, 8, 25, 11, 1] with length 75.20776208232135
Ants num 10  Iterations num 50
Best route at time 10 minutes: [1, 23, 13, 15, 12, 21, 26, 16, 22, 19, 18, 20, 0, 5, 17, 3, 6, 7, 9, 10, 2, 14, 24, 4, 8, 25, 11, 1] with length 75.10725701893303
Ants num 10  Iterations num 50
Best route at time 20 minutes: [2, 14, 11, 25, 1, 23, 13, 15, 12, 21, 26, 16, 22, 18, 19, 8, 20, 10, 4, 7, 6, 3, 17, 5, 0, 9, 24, 2] with length 76.0203643336844
Ants num 10  Iterations num 50
Best route at time 30 minutes: [4, 8, 25, 11, 1, 23, 13, 15, 12, 21, 26, 16, 22, 18, 19, 5, 17, 0, 3, 6, 7, 9, 10, 2, 14, 24, 20, 4] with length 71.57704766706702
Ants num 10  Iterations num 50
Best route at time 40 minutes: [1, 23, 13, 15, 12, 21, 26, 16, 22, 19, 18, 20, 8, 4, 5, 17, 0, 3, 6, 7, 9, 10, 14, 24, 2, 11, 25, 1] with length 73.10360979400616
Ants num 10  Iterations num 50
Best route at time 50 minutes: [1, 23, 13, 15, 12, 21, 26, 16, 22, 18, 19, 8, 4, 9, 7, 6, 3, 17, 5, 0, 20, 10, 24, 14, 2, 11, 25, 1] with length 74.80893129313469
Ants num 10  Iterations num 50
Best route at time 60 minutes: [2, 14, 24, 9, 7, 6, 3, 17, 5, 0, 20, 18, 19, 22, 16, 26, 21, 12, 15, 13, 23, 1, 11, 25, 8, 4, 10, 2] with length 72.04499927063449
Ants num 10  Iterations num 50
Best route at time 70 minutes: [4, 18, 19, 22, 16, 26, 21, 12, 13, 15, 23, 1, 11, 25, 8, 20, 9, 7, 6, 3, 0, 17, 5, 10, 14, 2, 24, 4] with length 77.71842072073692
Ants num 10  Iterations num 50
Best route at time 80 minutes: [0, 17, 3, 6, 7, 9, 24, 14, 2, 11, 1, 23, 13, 15, 12, 21, 26, 16, 22, 19, 18, 20, 10, 4, 8, 25, 5, 0] with length 76.1013982653461
Ants num 10  Iterations num 50
Best route at time 90 minutes: [4, 7, 6, 3, 17, 5, 20, 8, 25, 11, 1, 23, 13, 15, 12, 21, 26, 16, 22, 19, 18, 0, 9, 24, 14, 2, 10, 4] with length 77.28106724813428
Ants num 10  Iterations num 50
Best route at time 100 minutes: [0, 3, 6, 7, 9, 24, 2, 14, 10, 20, 19, 18, 22, 16, 26, 21, 12, 13, 15, 23, 1, 11, 25, 8, 4, 5, 17, 0] with length 71.20294395819512
Ants num 10  Iterations num 50
Best route at time 110 minutes: [1, 23, 13, 15, 12, 21, 26, 16, 22, 19, 20, 8, 4, 18, 17, 5, 0, 3, 6, 7, 9, 10, 24, 14, 2, 11, 25, 1] with length 72.09703708438666
Ants num 10  Iterations num 50
Best route at time 120 minutes: [4, 18, 19, 8, 25, 11, 1, 23, 13, 15, 12, 21, 26, 16, 22, 5, 17, 0, 3, 6, 7, 9, 24, 14, 2, 20, 10, 4] with length 74.64158582063479
Ants num 10  Iterations num 50
Best route at time 130 minutes: [4, 19, 18, 20, 5, 17, 0, 3, 6, 7, 9, 10, 2, 14, 24, 1, 23, 13, 15, 12, 21, 26, 16, 22, 25, 11, 8, 4] with length 77.30568755787226
Ants num 10  Iterations num 50
Best route at time 140 minutes: [2, 14, 24, 4, 18, 20, 8, 19, 25, 11, 1, 23, 13, 15, 12, 21, 26, 16, 22, 5, 17, 0, 3, 6, 7, 9, 10, 2] with length 76.08091947147645
Ants num 10  Iterations num 50
Best route at time 150 minutes: [4, 8, 19, 18, 22, 16, 26, 25, 11, 1, 23, 13, 15, 12, 21, 2, 14, 24, 9, 10, 7, 6, 3, 17, 5, 0, 20, 4] with length 76.92400060562538
Ants num 10  Iterations num 50
Best route at time 160 minutes: [3, 6, 7, 9, 10, 2, 14, 24, 20, 8, 25, 11, 1, 23, 13, 15, 12, 21, 26, 16, 22, 18, 19, 4, 5, 17, 0, 3] with length 72.94653096503122
Ants num 10  Iterations num 50
```






Best Delivery Route Visulization at time 380
Y Coordinate
X Coordinate
Best Delivery Route Visulization at time 500
Y Coordinate
X Coordinate


Best Delivery Route Visulization at time 1300
Y Coordinate
X Coordinate
Best Delivery Route Visulization at time 1430
Y Coordinate
X Coordinate