无人机配送路径规划问题报告

一、数据生成

假设配送中心有n个，坐标随机分布在坐标为0到100的平面上，对于每个配送中心，随机生成5到20个卸货点，这里为了简化问题，假设随机生成的卸货点距离配送中心都在10km以内。

生成代码如下：

```
def generate_centers_and_drops(n):
    centers_coords = {f'Center{i+1}': (random.uniform(0, 100), random.uniform(0,
100)) for i in range(n)}
    drop_points_coords = {}
    drop_points = []

    for center, (cx, cy) in centers_coords.items():
        mi = random.randint(5, 20)
        for i in range(mi):
            drop_name = f'Drop{len(drop_points) + 1}'
            drop_points.append(drop_name)
            distance = random.uniform(0, 10)  # DISTANCE_BETWEEN_POINTS
            angle = random.uniform(0, 2 * math.pi)
            drop_x = cx + distance * math.cos(angle)
            drop_y = cy + distance * math.sin(angle)
            drop_points_coords[drop_name] = (drop_x, drop_y)

    return centers_coords, drop_points_coords, drop_points
```

生成的距离矩阵如图：

二、问题描述

在某区域内，有j 个配送中心和k 个卸货点。配送中心有足够的商品和无人机，可以满足所有订单需求。每个订单只有一个商品，并且具有三个优先级别：一般（3小时内配送到）、较紧急（1.5小时内配送到）、紧急（0.5小时内配送到）。系统每隔t 分钟生成订单，并每隔t 分钟做出决策，包括分配无人机和规划路径。

目标

在一段时间内（如一天），最小化所有无人机的总配送路径长度，同时满足订单的优先级别要求。

约束条件

1. 无人机必须在订单的优先级时间内完成配送。
2. 无人机一次最多携带n 个物品。
3. 无人机一次飞行的总距离（包括返回）不超过20公里。
4. 无人机的速度为60公里/小时。

假设条件

1. 配送中心的商品数量和无人机数量无限。
2. 每个配送中心都能满足用户的订货需求。

三、数学建模

问题化简

为了简化问题，可以将每个配送中心单独处理，并针对每个配送中心及其对应的卸货点进行订单分配和路径规划。这样可以减少问题的复杂性，并使得路径规划更加高效。

具体步骤如下：

1. **单独处理配送中心**：
   - 对于每个配送中心Dj，提取该配送中心的所有订单集合Oj和对应的卸货点集合kj。
   - 针对每个Dj，独立进行订单分配和路径规划。
2. **订单分配和路径规划**：
   - 对于每个配送中心Dj及其对应的订单集合Oj，按照订单优先级和距离排序。
   - 为每个配送中心Dj 规划无人机的配送路径，并确保在优先级和飞行距离限制内完成订单。

变量定义

- O：订单集合。
- P：路径集合。
- D：配送中心集合。
- U：无人机集合。

目标函数

最小化总配送路径：

$$\text{Minimize} \sum_{u \in U} \sum_{p \in P_u} d_p$$

其中，dp 表示路径p 的长度。

约束条件

1. **订单优先级时间限制**：

$$t_o \leq T_o \quad \forall o \in O$$

其中，to 表示订单o 的配送时间，To 表示订单o 的优先级时间限制。

2. **无人机载荷限制**：

$$\sum_{o \in O_u} w_o \leq n \quad \forall u \in U$$

其中，wo 表示订单o 的重量。

3. **无人机飞行距离限制**：

$$\sum_{p \in P_u} d_p \leq 20 \quad \forall u \in U$$

四、算法设计思路

算法设计将原问题分成两个主要部分，配送中心订单分配和单个中心路径规划。其中配送中心订单分配将距离卸货点最近的配送中心作为订单的分配中心；单个配送中心采用贪心算法完成路径的规划。

初始化

- **输入**：配送中心数量n、每个配送中心的最大负载MAX_LOAD、最大飞行距离MAX_DISTANCE、无人机速度SPEED。
- **输出**：配送中心和卸货点的坐标集合。

```
def generate_centers_and_drops(n):
    centers_coords = {f'Center{i+1}': (random.uniform(0, 100), random.uniform(0,
100)) for i in range(n)}
    drop_points_coords = {}
    drop_points = []

    for center, (cx, cy) in centers_coords.items():
        mi = random.randint(5, 20)
        for i in range(mi):
            drop_name = f'Drop{len(drop_points) + 1}'
            drop_points.append(drop_name)
            distance = random.uniform(0, 10)  # DISTANCE_BETWEEN_POINTS
            angle = random.uniform(0, 2 * math.pi)
            drop_x = cx + distance * math.cos(angle)
            drop_y = cy + distance * math.sin(angle)
            drop_points_coords[drop_name] = (drop_x, drop_y)

    return centers_coords, drop_points_coords, drop_points
```

订单生成

- **输入**：时间间隔t、每个卸货点的最大订单数m。
- **输出**：订单集合O。

```
def generate_orders(time_interval):
    orders = []
    for drop in drop_points:
        num_orders = random.randint(0, 5)
        for _ in range(num_orders):
            priority = random.choice(list(DELIVERY_TIME.keys()))
            orders.append((drop, priority))
    return orders
```

订单分配和路径规划

单独处理每个配送中心： 对每个配送中心Dj，提取订单集合Oj 和卸货点集合kj。

- **输入**：订单集合O、配送中心集合D、卸货点集合k。
- **输出**：每个配送中心对应的订单集合。

```
def assign_orders_to_centers(orders):
    center_orders = defaultdict(list)
    for order in orders:
        drop, priority = order
        closest_center = min(centers, key=lambda center: distances[(center,
drop)])
        center_orders[closest_center].append(order)
    return center_orders
```

订单排序： 对每个Oj，按照订单距离配送中心的距离从小到大排序。

- **输入**：配送中心对应的订单集合。
- **输出**：按距离排序的订单集合。

```
orders_heap = []
for order in orders_list:
    heapq.heappush(orders_heap, (distances[(center, order[0])], order))
```

路径构建： 对于每个Dj 的订单列表，按照以下规则构建路径：

- **输入**：按距离排序的订单集合、无人机的最大负载和最大飞行距离。
- **输出**：无人机的配送路径。
- 无人机从订单列表中获取订单，并将订单的卸货点加入路径。

```
while orders_heap:
    _, next_order = orders_heap[0]
    drop, priority = next_order
```

- 如果该订单不是第一个订单，则判断该订单的卸货点距离路径中最后一个地址的距离和该订单的第一个地址之间的距离，如果前者较小，则加入路径中，否则结束添加订单。

```
if len(drone_path) > 1:
    last_drop = drone_path[-1]
    trip_distance = distances[(last_drop, drop)]
    dist_to_center = distances[(center, drop)]

    if dist_to_center < trip_distance:
        drone_path.append(center)
        print(f"无人机从{center}出发的路径: {' -> '.join(drone_path)}, 总距离:
{total_distance + distances[(last_drop, center)]:.2f}公里")
        all_paths.append(drone_path)

        drone_path = [center]
        total_distance = 0
        num_orders = 0
```

```
        continue
```

- 在每次将订单加入路径前，判断无人机在完成该订单时的配送时间是否满足订单的优先级要求。
- 如果满足，则将订单加入路径，否则跳过该订单。

```
delivery_time_required = trip_distance / SPEED * 60  # 转换为分钟
if delivery_time_required <= DELIVERY_TIME[priority]:
    heapq.heappop(orders_heap)
    drone_path.append(drop)
    total_distance += trip_distance
    num_orders += 1
else:
    break
```

- 确保订单的数量不超过无人机的载荷限制。
- 确保无人机的总飞行距离（包括返回）不超过飞行距离限制。

```
if num_orders < MAX_LOAD and potential_total_distance <= MAX_DISTANCE:
    ...
else:
    drone_path.append(center)
    print(f"无人机从{center}出发的路径: {' -> '.join(drone_path)}, 总距离:
{total_distance + distances[(last_drop, center)]:.2f}公里")
    all_paths.append(drone_path)

    drone_path = [center]
    total_distance = 0
    num_orders = 0
```

- 如果任何一个限制不满足，则当前无人机完成取货，开始从剩余订单列表中为下一个无人机取货。

实时更新

- **输入**：时间间隔t。
- **输出**：更新后的无人机状态和订单池。

完成配送和返回

- **输入**：所有分配的订单。
- **输出**：无人机完成所有订单后的返回路径。

结果展示

为了评估算法的性能和效果，进行了一些实验。以下是一些关键结果展示：

实验设置

1. **配送中心数量**：5个。
2. **卸货点数量**：38个。
3. **最大订单数**：每个卸货点每次生成最多5个订单。
4. **无人机最大负载**：10个物品。

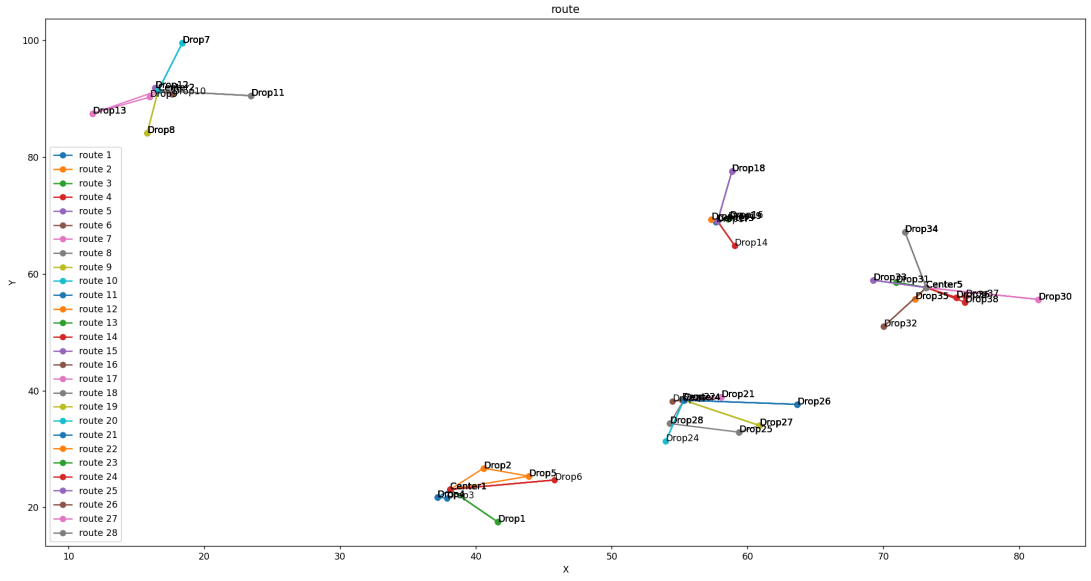
5. **无人机最大飞行距离**：20公里（包括返回）。
6. **无人机速度**：60公里/小时。

实验结果

1. **路径总长度**：所有无人机在一天内的总配送路径长度。
2. **无人机路径**：每个无人机飞行的路径。
3. **完成订单数**：成功完成的订单数量。

```
[('Drop1', '紧急'), ('Drop1', '普通'), ('Drop1', '普通'), ('Drop2', '普通'), ('Drop2', '普通'), ('Drop2', '普通'), ('Drop2', '普通'), ('Drop3', '较 紧急'), ('Drop4', '普通'), ('Drop4', '普通'), ('Drop4', '普通'), ('Drop4', '紧急'), ('Drop4', '紧急'), ('Drop5', '普通'), ('Drop5', '较紧急'), ('Drop5', '较紧急'), ('Drop5', '普通'), ('Drop6', '较紧急'), ('Drop7', '紧急'), ('Drop7', '普通'), ('Drop7', '较紧急'), ('Drop7', '紧急'), ('Drop7', '紧 急'), ('Drop8', '普通'), ('Drop8', '紧急'), ('Drop8', '较紧急'), ('Drop8', '较紧急'), ('Drop8', '较紧急'), ('Drop9', '较紧急'), ('Drop9', '普通'), ('Drop10', '紧急'), ('Drop11', '普通'), ('Drop11', '紧急'), ('Drop11', '普通'), ('Drop通'), ('通')通'), ('Drop18', '较紧急'), ('Drop18', '紧急'), ('Drop18', '紧急'), ('Drop18', '较紧急'), ('Drop18', '较紧急'), ('Drop19', '较紧急'), ('Drop19', '紧急'), ('Drop19', '普通'), ('Drop21', '较紧急'), ('Drop21', '紧急'), ('Drop21', '较紧急'), ('Drop21', '普通'), ('Drop22', '普通'), ('Dr通'), ('Drop18'通'),通'), ('Drop18', '较紧急'), ('Drop18', '紧急'), ('Drop18', '紧急'), ('Drop18', '较紧急'), ('Drop19', '较紧急'), ('Drop19', '紧急'), ('Drop19', '普通'), ('Drop21', '较紧急'), ('Drop21', '紧急'), ('Drop21', '较紧急'), ('Drop21', '普通'), ('Drop22', '普通'), ('Drop22', '较紧急'), ('Drop22', '较紧急'), ('Drop24', '较紧急'), ('Drop25', '紧急'), ('Drop25', '较紧急'), ('Drop25', '普通'), ('Drop26', '紧急'), ('Drop26', '紧急'), ('Drop26', '紧急'), ('Drop26', '普通'), ('Drop27', '普通'), ('Drop27', '较紧急'), ('Drop27', '较紧急'), ('Drop27', '较紧急'), ('Drop27', '较紧急'), ('Drop28', '普通'), ('Drop28', '较紧急'), ('Drop28', '普通'), ('Drop28', '普通'), ('Drop29', '紧急'), ('Drop30', '较紧急'), ('Drop30', '普通'), ('Drop30', '普通'), ('Drop30', '紧急'), ('Drop31', '紧急'), ('Drop31', '较紧急'), ('Drop31', '普通'), ('Drop31', '普通'), ('Drop32', '紧急'), ('Drop32', '普通'), ('Drop33', '紧急'), ('Drop33', '紧急'), ('Drop33', '普通'), ('Drop33', '较紧急'), ('Drop34', '普通'), ('Drop34', '普通'), ('Drop34', '紧急'), ('Drop34', '较紧急'), ('Drop34', '紧急'), ('Drop35', '紧急'), ('Drop35', '普通'), ('Drop35', '较紧急'), ('Drop35', '普通'), ('Drop36', '紧急'), ('Drop36', '紧急'), ('Drop36', '普通'), ('Drop36', '普通'), ('Drop37', '普通'), ('Drop37', '较紧急'), ('Drop38', '普通'), ('Drop38', '紧急'), ('Drop38', '较紧急')]
订单数： 117
无人机从Center1出发的路径：Center1 -> Drop3 -> Drop4 -> Drop4 -> Drop4 -> Drop4 -> Center1，总距离：3.87公里
无人机从Center1出发的路径：Center1 -> Drop2 -> Drop2 -> Drop2 -> Drop2 -> Drop5 -> Drop5 -> Drop5 -> Drop5 -> Center1，总距离：14.18公里
无人机从Center1出发的路径：Center1 -> Drop1 -> Drop1 -> Drop1 -> Center1，总距离：13.23公里
无人机从Center1出发的路径：Center1 -> Drop6 -> Center1，总距离：15.69公里
无人机从Center2出发的路径：Center2 -> Drop12 -> Drop12 -> Drop12 -> Drop12 -> Center2，总距离：1.07公里
无人机从Center2出发的路径：Center2 -> Drop10 -> Center2，总距离：2.48公里
无人机从Center2出发的路径：Center2 -> Drop9 -> Drop9 -> Drop13 -> Drop13 -> Drop13 -> Center2，总距离：12.51公里
无人机从Center2出发的路径：Center2 -> Drop11 -> Drop11 -> Drop11 -> Drop11 -> Center2，总距离：13.78公里
无人机从Center2出发的路径：Center2 -> Drop8 -> Drop8 -> Drop8 -> Drop8 -> Center2，总距离：14.52公里
无人机从Center2出发的路径：Center2 -> Drop7 -> Drop7 -> Drop7 -> Drop7 -> Center2，总距离：16.82公里
无人机从Center3出发的路径：Center3 -> Drop17 -> Center3，总距离：0.43公里
无人机从Center3出发的路径：Center3 -> Drop15 -> Drop15 -> Center3，总距离：1.15公里
无人机从Center3出发的路径：Center3 -> Drop19 -> Drop19 -> Drop19 -> Drop16 -> Drop16 -> Center3，总距离：2.15公里
无人机从Center3出发的路径：Center3 -> Drop14 -> Center3，总距离：8.70公里
无人机从Center3出发的路径：Center3 -> Drop18 -> Drop18 -> Drop18 -> Drop18 -> Center3，总距离：17.19公里
无人机从Center4出发的路径：Center4 -> Drop22 -> Drop22 -> Drop22 -> Drop29 -> Center4，总距离：1.72公里
无人机从Center4出发的路径：Center4 -> Drop21 -> Drop21 -> Drop21 -> Drop21 -> Center4，总距离：5.59公里
无人机从Center4出发的路径：Center4 -> Drop28 -> Drop28 -> Drop28 -> Drop28 -> Drop25 -> Drop25 -> Drop25 -> Drop27 -> Drop27 -> Drop27 -> Center4，总距离：18.38公里
无人机从Center4出发的路径：Center4 -> Drop27 -> Drop27 -> Center4，总距离：14.10公里
无人机从Center4出发的路径：Center4 -> Drop24 -> Center4，总距离：14.32公里
无人机从Center4出发的路径：Center4 -> Drop26 -> Drop26 -> Drop26 -> Center4，总距离：16.78公里
无人机从Center5出发的路径：Center5 -> Drop35 -> Drop35 -> Drop35 -> Drop35 -> Center5，总距离：4.23公里
无人机从Center5出发的路径：Center5 -> Drop31 -> Drop31 -> Drop31 -> Center5，总距离：4.84公里
无人机从Center5出发的路径：Center5 -> Drop36 -> Drop36 -> Drop36 -> Drop36 -> Drop37 -> Drop37 -> Drop38 -> Drop38 -> Drop38 -> Center5，总距离：8.30公里
无人机从Center5出发的路径：Center5 -> Drop33 -> Drop33 -> Drop33 -> Center5，总距离：8.21公里
无人机从Center5出发的路径：Center5 -> Drop32 -> Drop32 -> Center5，总距离：14.67公里
无人机从Center5出发的路径：Center5 -> Drop30 -> Drop30 -> Drop30 -> Center5，总距离：16.97公里
无人机从Center5出发的路径：Center5 -> Drop34 -> Drop34 -> Drop34 -> Drop34 -> Center5，总距离：19.35公里
总飞行里程： 285.2409031177523 公里
```

4. **路径可视化**：显示出规划的路径



性能评估

1. **时间复杂度**：订单分配和路径规划算法的时间复杂度。
2. **空间复杂度**：算法运行过程中所需的存储空间。

时间复杂度

- 订单排序的时间复杂度为 O(nlogn)。
- 路径规划的时间复杂度为 O(n)。

空间复杂度

- 存储订单和路径所需的空间为 O(n)。

算法改进：使用遗传算法

为了进一步优化无人机的路径规划，可以尝试引入遗传算法来改进现有的路径规划方法。遗传算法通过模拟自然选择过程，逐步进化出最优解，适用于解决复杂的组合优化问题。

1. 基因编码（Genetic Encoding）

在遗传算法中，每个个体表示一种潜在的解决方案。对于无人机配送问题，一个个体可以表示为多条路径，其中每条路径由一个无人机执行：

- **基因**：每个基因代表一个订单的配送点（Drop点）。
- **染色体**：每个染色体代表一个无人机的配送路径。
- **个体**：个体由多个染色体组成，每个染色体对应一个无人机的任务列表。

例如，个体可以表示为二维数组：

```
[
    ["Drop1", "Drop3", "Drop5"],
    ["Drop2", "Drop4", "Drop6", "Drop7"]
]
```

这表示有两个无人机，第一个无人机负责配送Drop1、Drop3和Drop5，第二个无人机负责配送Drop2、Drop4、Drop6和Drop7。

2. 适应性函数（Fitness Function）

适应性函数用于评价个体的适应度，即解决方案的质量。对于无人机配送问题，需要确保：

- 所有订单都必须被配送。
- 任一无人机的载荷不能超过其最大载荷。
- 每个订单的配送时间必须满足用户设定的时间限制。
- 无人机的总飞行距离不能超过最大飞行距离。

适应性函数可以设计为：

```
def fitness(individual, distances, max_load, max_distance, delivery_time, speed):
    total_distance = 0
    penalty = 0

    # 遍历每个无人机的配送路径
    for path in individual:
        if len(path) > max_load:
            penalty += (len(path) - max_load) * penalty_per_overload_item

        path_distance = 0
        last_location = "Center"
        for drop in path:
            path_distance += distances[(last_location, drop)]
            last_location = drop
```

```
            # 返回中心的距离
            path_distance += distances[(last_location, "Center")]

            if path_distance > max_distance:
                penalty += (path_distance - max_distance) *
penalty_per_excess_distance

            total_distance += path_distance

        if not all_orders_delivered(individual):
            penalty += penalty_for_missing_orders

        if not meet_delivery_time_requirements(individual, delivery_time, speed):
            penalty += penalty_for_time_violation

        # 计算总适应度
        return 1 / (total_distance + penalty)
```

3. 遗传操作（Genetic Operations）

- **选择（Selection）**：可以使用锦标赛选择或轮盘赌选择法来选择用于繁殖的个体。
- **交叉（Crossover）**：可以使用部分映射交叉（PMX）或顺序交叉（OX）等适用于序列问题的交叉方法。
- **变异（Mutation）**：可以使用交换变异，即随机选择染色体中的两个基因（两个Drop点）并交换它们的位置，以引入随机性。

4. 算法参数

- **种群大小**：根据问题的规模，选择适当的种群大小。
- **迭代次数**：执行足够的迭代次数以找到一个好的解决方案。
- **交叉率和变异率**：适当设定，通常交叉率较高（如0.7-0.9），变异率较低（如0.1-0.2）。

总结

在本次项目中，设计并尝试实现了一种无人机配送路径规划系统。虽然最终的遗传算法未能完全实现，但通过其他方法有效地探索了无人机配送路径的最优化问题，并提出了初始的数学建模和基本的路径规划算法。

成果总结

1. **问题理解与建模**：成功地将实际的无人机配送问题转化为数学模型，定义了清晰的目标函数和约束条件，这是解决任何优化问题的第一步。
2. **基本路径规划实现**：通过简化的算法初步处理了订单分配和无人机路径规划问题。这包括订单的接收、排序和基于距离及优先级的初步路径规划。
3. **系统架构设计**：为该问题建立了完整的系统架构，包括订单生成、订单分配、路径规划等模块。

遇到的挑战

实现遗传算法的复杂性超出了初期预期，主要挑战包括：

- **算法复杂度**：遗传算法的设计和调试过程中，面临参数调优、编码方式选择等技术挑战。
- **时间限制**：完成算法的全部实现并进行充分测试在有限时间内难道较高。