
# 《高级算法设计与分析》课程作业

## 无人机配送路径规划问题

## UAV Delivery Path Planning Problem


| | | |
|---|---|---|
| 姓　　名 | ： | 赵路路 |
| 学　　号 | ： | 2023202210120 |
| 专　　业 | ： | 网络空间安全 |


2024 年 6 月 4 日

# 目录

# 1 问题重述

无人机配送是解决最后 10 公里问题的有效方法。在某区域内存在 j 个配送中心，每个配送中心具有无限多的任意商品和无限多的无人机；存在 k 个卸货点，每个卸货点每隔 t 分钟随机产生 0～m 个订单；订单存在优先级别，一般订单要求 3 小时内送到，较紧急订单要求 1.5 小时内送到，紧急订单要求 0.5 小时内送到。

问题的目标是：一段时间内，所有无人机的总配送路径最短。

问题的约束是：

a) 满足订单的优先级别要求；

b) 无人机一次最多可以携带 n 个物品；

c) 无人机一次飞行最远路程 20 公里（包含返回配送点）；

d) 无人机的速度是 60 公里/小时；

# 2 解决思路

根据题目要求和数据模拟需要，解决思路可分为三个步骤：地图与订单数据模拟生成、订单分配与无人机调度、无人机配送路径规划，下面具体介绍每个步骤的内容。

## 2.1 地图与订单数据模拟生成

首先介绍地图的模拟生成过程。由于无人机配送用于解决最后 10 公里问题，故每个配送中心可以辐射到其周围 10 公里内的卸货点，在地图模拟生成时，要保证所有卸货点均在某个配送中心的辐射范围内。因此，设计地图模拟生成算法如下：

**Alg1:** Map Simulation Algorithm (MSA)

**Input:** $scaleX$, $scaleY$, $j$, $k$ // 区域范围、配送中心与卸货点数量

**Output:** $distributionCenterSet$, $dischargePointSet$ // 配送中心与卸货点集合

**while** $distributionCenterSet.size < j$:
  $(x, y) = \big(random.nexInt(scaleX), random.nexInt(scaleY)\big)$
  **if** $distributionCenterSet.isEmpty$:
    $distributionCenterSet.add\big((x, y)\big)$ // 随机生成配送中心位置并加入集合
  **for** $i = 1$ **to** $distributionCenterSet.size$:
    **if** $distance\big(distributionCenterSet[i].getPostion, (x, y)\big) < 8$:
      $distributionCenterSet.remove\big((x, y)\big)$ // 配送中心间距离不小于 8
  **end for**
**end while**
**while** $dischargePointSet.size < k$:
  $(x, y) = \big(random.nexInt(scaleX), random.nexInt(scaleY)\big)$
  **for** $i = 1$ **to** $j$:
    **if** $distance\big(distributionCenterSet[i].getPosition, (x, y)\big) \leq 10$:
      $dischargePointSet.add\big((x, y)\big)$ // 卸货点需在任意一个配送中心 10 公里内
  **end for**
**end while**

根据 MSA 地图模拟生成算法，随机生成两组不同规模的地图如图 2-1 和 2-2 所示：

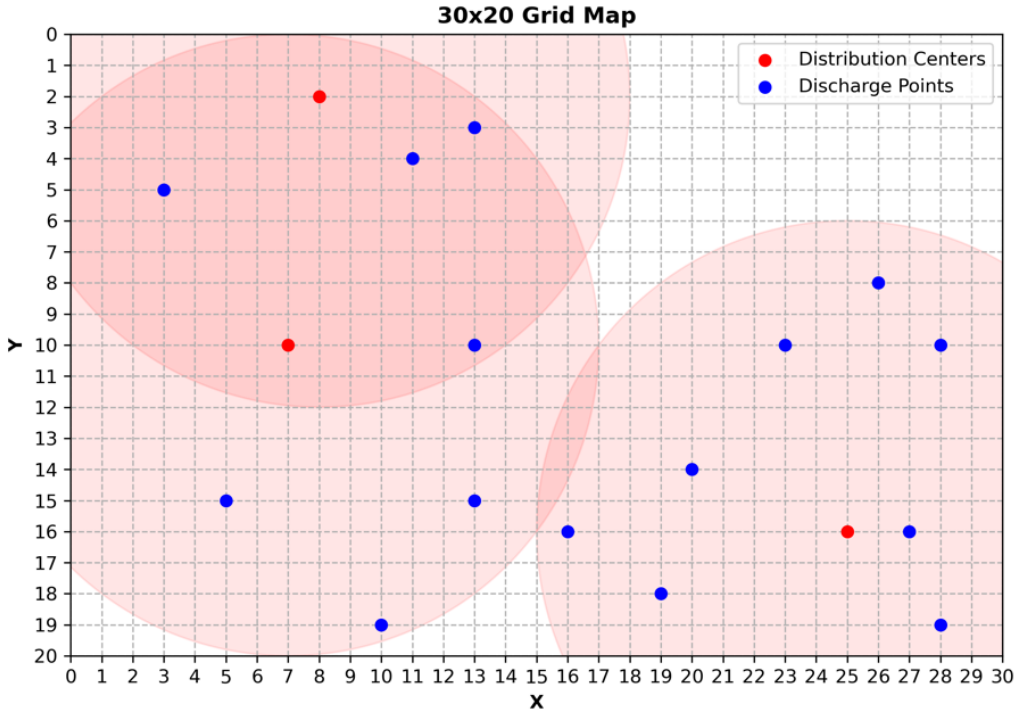


图 2-1 30×20 地图（3 中心 15 卸货点）示例

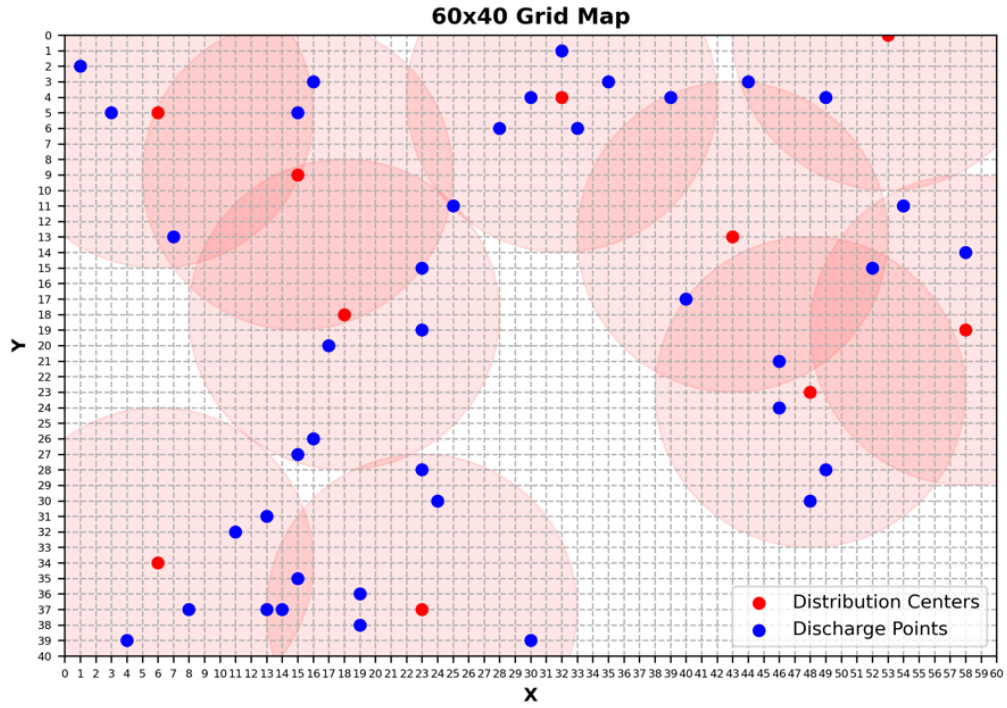


图 2-2 30×20 地图（10 中心 40 卸货点）示例

下面介绍订单的模拟生成过程。根据题目要求，每个卸货点每隔 t 分钟随机产生 0～m 个不同优先级别的订单，将其加入待处理订单队列，设计订单模拟生成算法如下：

**Alg2:** Order Simulation Algorithm (OSA)

**Input:** $dischargePointSet,\ m,\ currentTime$ // 卸货点集合、最大订单数和当前时间

**Output:** $orders$ // 待处理订单队列

1: **for** $i = 1$ **to** $dischargePointSet.size$:
2: $num = random.nextInt(m + 1)$ // 随机生成 0～m 个订单
3: **for** $j = 1$ **to** $num$:
4: $priority = Util.randomPriority$
5: $createTime = currentTime$
6: $remainTime = priority.getTime$
7: $position = dischargePointSet[i].getPosition$
8: $orders.add((priority, createTime, remainTime, position))$
9: **end for**
10: **end for**

根据 OSA 订单模拟生成算法，在 0 时刻对于如图 2-1 所示地图（3 配送中心 15 卸货点），最大订单数 m=3，生成一组订单如表 2-1 所示：

表 2-1 订单模拟生成算法示例

| 序号 | 余时 | 位置 | 序号 | 余时 | 位置 | 序号 | 余时 | 位置 |
|---|---|---|---|---|---|---|---|---|
| 1 | 90 | (28,19) | 10 | 90 | (28,10) | 19 | 90 | (3,5) |
| 2 | 180 | (28,19) | 11 | 30 | (28,10) | 20 | 180 | (10,19) |
| 3 | 30 | (20,14) | 12 | 180 | (28,10) | 21 | 90 | (10,19) |
| 4 | 180 | (20,14) | 13 | 90 | (13,10) | 22 | 30 | (16,16) |
| 5 | 30 | (19,18) | 14 | 90 | (13,10) | 23 | 90 | (23,10) |
| 6 | 30 | (27,16) | 15 | 30 | (5,15) | 24 | 90 | (23,10) |
| 7 | 30 | (27,16) | 16 | 30 | (5,15) | 25 | 180 | (23,10) |
| 8 | 90 | (13,15) | 17 | 90 | (5,15) | 26 | 30 | (13,3) |
| 9 | 180 | (13,15) | 18 | 180 | (3,5) | 27 | 90 | (13,3) |

## 2.2 订单分配与无人机调度

根据题目要求，每隔 t 分钟将会产生一组新订单加入待处理订单队列，所以假设系统每隔 t 分钟将进行一次订单分配与无人机调度，完成其中一部分订单的配送。

首先介绍订单分配过程。对于如图 2-1 所示地图（3 配送中心 15 卸货点），配送中心辐射范围相互重叠，存在某些卸货点既可以被 A 配送中心配送，又可以被 B 配送中心配送。对于这种情形，我们考虑卸货点与配送中心的距离，每个卸货点均由距离其最近的配送中心配送，因此需要将待处理订单队列进行订单分配。设计订单分配算法如下：

**Alg3:** Order Distribution Algorithm (ODA)

**Input:** $orders$, $distributionCenterSet$ // 待处理订单队列与配送中心集合

**Output:** $orders_i$ // 根据最近配送中心分配后的待处理订单队列

1: **for** $i = 1$ **to** $orders.size$:
2: **for** $j = 1$ **to** $distributionCenterSet.size$:
3: $d = distance(orders[i].getPosition, distributionCenterSet[i].getPosition)$
4: **if** $d < minDistance$: // 寻找与此订单目的卸货点最近的配送中心
5: $minDistance = d$
6: $flag = j$
7: **end if**
8: **end for**
9: $orders_{flag}.add(orders[i])$ // 将此订单加入最近配送中心的待处理订单队列
10: **end for**

下面介绍无人机调度过程。对于 ODA 订单分配算法得到的每个配送中心的待处理订单，将订单按照剩余时间从小到大排序，优先配送紧急订单（剩余时间≤30 分钟），适量配送较紧急订单（剩余时间≤90 分钟），暂不配送一般订单（剩余时间≤180 分钟）。这样做的好处在于，对于一些相对不紧急的订单，可以在下一次调度时一起配送，使无人机负载尽量保持在较高水平，减少无人机资源的浪费，进而达到更优的总体配送路径长度。根据题目要求，一段时间后（例如一天），所有订单都需要配送完毕。所以在最后一个时刻，需要不惜无人机资源，将所有订单配送完毕。

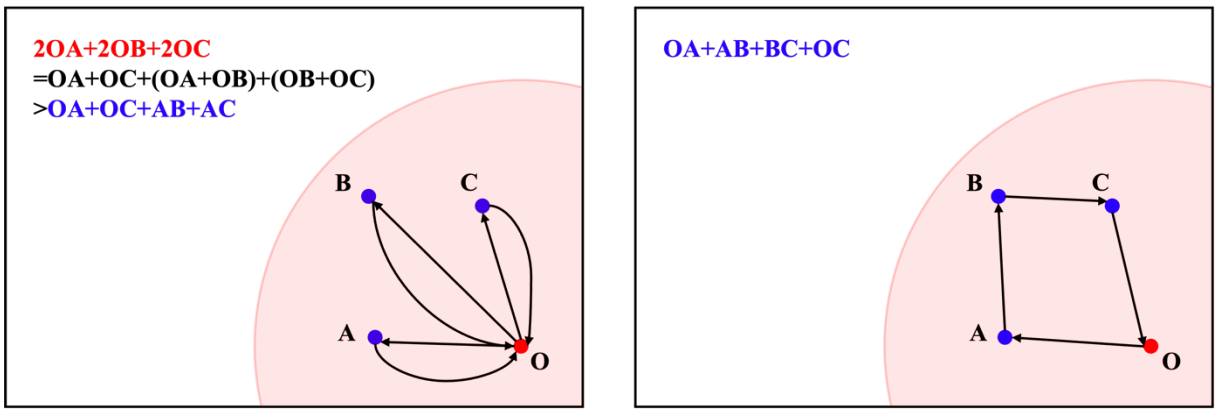


图 2-3 折返配送与连续配送对比

如图 2-3 所示，根据三角形两边之和大于第三边的性质可知，多次折返配送的代价大于连续配送的代价。所以，在确定本次调度需要配送的订单后，使用简单贪心算法，按顺序为订单安排无人机。

为满足问题约束 b～d，假设无人机一次最多可以携带 n=5 个物品，一次最远飞行距离为 20 公里，速度为 60 公里/小时。根据简单贪心原则，为每架无人机尽可能多的安排订单，安排时需要检验：（1）是否达到无人机最大负载；（2）订单是否会超时；（3）无人机能否返回配送中心。设计无人机调度算法如下：

**Alg4:** Drone Scheduling Algorithm (DSA)

**Input:** $orders$, $n$, $maxDis$, $s$, $crrentTime$ // 订单队列、最大负载/距离/速度和当前时间

**Output:** $drones$ // 该配送中心本次调度的无人机群

1: **while** $!orders.isEmpty$: // 某中心订单待处理订单队列中本次调度需配送的订单

2: $drones[i].path = [centerPosition]$

3: **while** $drones[i].currentLoad + 1 \leq n$: // 该无人机的负载允许增加新订单

4: **for** $j = 0$ **to** $orders.size$:

5: $drones.path.add(orders[j])$

6: **if** $dis(drones.path) \leq 20$ and $time(drones.path) \leq orders[j].remainTime$:

7: **break** // 找到合适的新订单后结束本次循环

8: **end if**

9: $drones.path.remove(orders[j])$ // 该订单会超时或导致无法返回中心故去除

10: **end for**

11: **end while**

12: **end while**

## 2.3 无人机配送路径规划

本节介绍更进一步的无人机路径规划过程。如图 2-4 所示，根据简单贪心原则，DSA 无人机调度算法给出了每个配送中心派出的无人机群。对于每架无人机，其配送订单已经确定，但并没有得到最优配送路径，该问题可以建模为旅行商问题，许多算法可以得到其近似解。

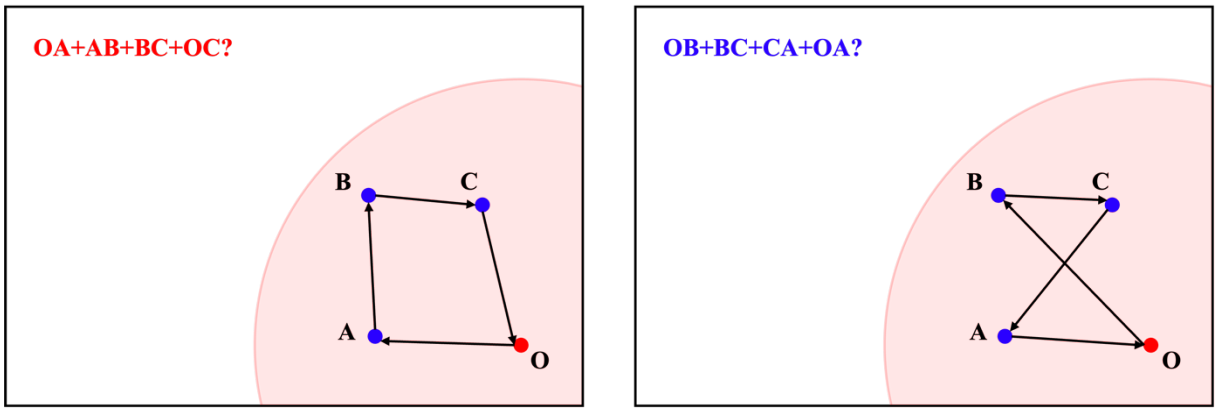


图 2-3 无人机最优配送路径不确定

但对于当前假设而言，一架无人机每次最多携带 n=5 个物品，故最多途径 5 个卸货点，所以可以穷举所有可能找出最优路径的精确解，故此处不再赘述具体的算法设计。

# 3 问题解决

# 4 问题总结

无人机配送是解决最后 10 公里问题的有效方法。

本文从问题的场景假设、目标和约束入手，通过地图与订单数据模拟生成、订单分配与无人机调度和无人机配送路径规划三个步骤，设计并实现了 MSA 地图模拟生成算法、OSA 订单模拟生成算法、ODA 订单分配算法和 DSA 无人机调度算法四个算法，最终得出了无人机配送问题较优的可行解。