武漢大學

高级算法设与分析课程报告

基于动态规划的无人机配送路径方案设计

学　　院：国家网络安全学院

专　　业：网络空间安全

姓　　名：杨晓帅

学　　号：2023202210065

Contents

基于动态规划的无人机配送路径方案设计

June 30, 2024

1 问题描述

无人机可以快速解决最后 10 公里的配送，将无人机应用于物流配送可以极大提高效率。假设有如下区域（Figure1），需要我们完成无人机的配送路径规划。在此区域中，共有 j 个配送中心（用蓝色方形表示），以及 k 个卸货点（用绿色圆圈表示），无人机只需要将货物放到相应的卸货点即可。

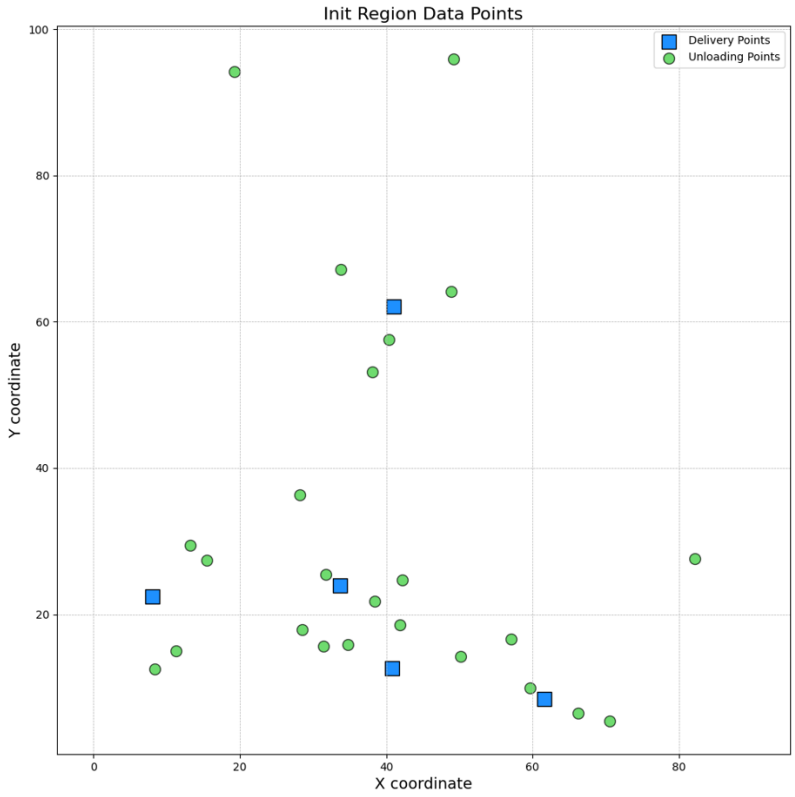

Figure 1: 初始区域的配送中心与卸货点分布

对于每个配送中心有以下假设条件：

(1) 无人机一次最多只能携带 n 个物品；

(2) 无人机一次飞行最远路程为 20 公里（无人机送完货后需要返回配送点）；

(3) 无人机的速度为 60 公里/小时；

（4）配送中心的无人机数量无限；

（5）任意一个配送中心都能满足用户的订货需求；

对于每个卸货点有以下假设条件：

（6）假设每个卸货点每隔一段时间（假设为 t 分钟）会随机生成 m 个订单，一个订单只有一个商品;

（7）订单有优先级别，分为三个优先级别（用户下订单时，会选择优先级别，优先级别高的付费高）：一般：3 小时（180 分钟）内配送到即可；较紧急：1.5 小时（90 分钟内）配送到即可；紧急：0.5 小时（30 分钟内）配送到即可。

此时配送中心需要做出决策，包括：

（1）哪些配送中心出动多少无人机完成哪些订单；

（2）每个无人机的路径规划，即先完成那个订单，再完成哪个订单，最后返回原来的配送中心；

（3）系统做决策时，可以不对当前的某些订单进行配送，因为当前某些订单可能紧急程度不高，可以累积后和后面的订单一起配送。

目标：一段时间内（如一天），所有无人机的总配送路径最短

约束条件：满足订单的优先级别要求以及所有假设条件

2 背景

2.1 车辆路径规划问题

上述无人机配送路径规划问题是车辆路径规划问题 (Vehicle Routing Problem，VRP) 的一种。VRP 问题可以简单地描述为：给定一系列发货点和收货点，如何组织车辆以最小的总成本或最短的总时间完成所有送货任务，并返回到起始点。这个问题涉及到多个约束条件，如车辆容量限制、时间窗口限制、路线长度限制等。具体地说是在路线长度限制情况下的 VRP 问题。

VRP 问题可以分为多种类型，包括带时间窗的车辆路径问题（Vehicle Routing Problem with Time Windows, VRPTW）、带容量限制的车辆路径问题（Capacitated Vehicle Routing Problem, CVRP）等。每种类型都有其特定的约束条件和优化目标，需要采用不同的算法和策略来解决。

解决 VRP 问题的方法主要可以分为两大类：精确算法和启发式算法。精确算法，如整数规划、动态规划等，能够找到最优解，但对于大规模问题可能非常耗时。启发式算法，如遗传算法、模拟退火算法、蚁群算法等，能够在较短时间内找到近似最优解，适用于大规模问题。

2.2 旅行商问题

旅行商问题 (TravelingSalesmanProblem，TSP) 是一个经典的组合优化问题。经典的 TSP 可以描述为：一个商品推销员要去若干个城市推销商品，该推销员从一个城市出发，需要经过所有城市后，回到出发地。应如何选择行进路线，以使总的行程最短。从图论的角度来看，该问题实质是在一个带权完全无向图中，找一个权值最小的 Hamilton 回路。由于该问题的可行解是所有顶点的全排列，随着顶点数的增加，会产生组合爆炸，它是一个 NP 完全问题。由于其在交通运输、电路板线路设计以及物流配送等领域内有着广泛的应用，国内外学者对其进行了大量的研究。早期的研究者使用精确算法求解该问题，常用的方法包括：分枝定界法、线性规划法、动态规划

法等。但是，随着问题规模的增大，精确算法将变得无能为力，因此，在后来的研究中，国内外学者重点使用近似算法或启发式算法，主要有遗传算法、神经网络等。

TSP 是旅行购买者问题与车辆路径规划问题（VRP）的一种特殊情况。

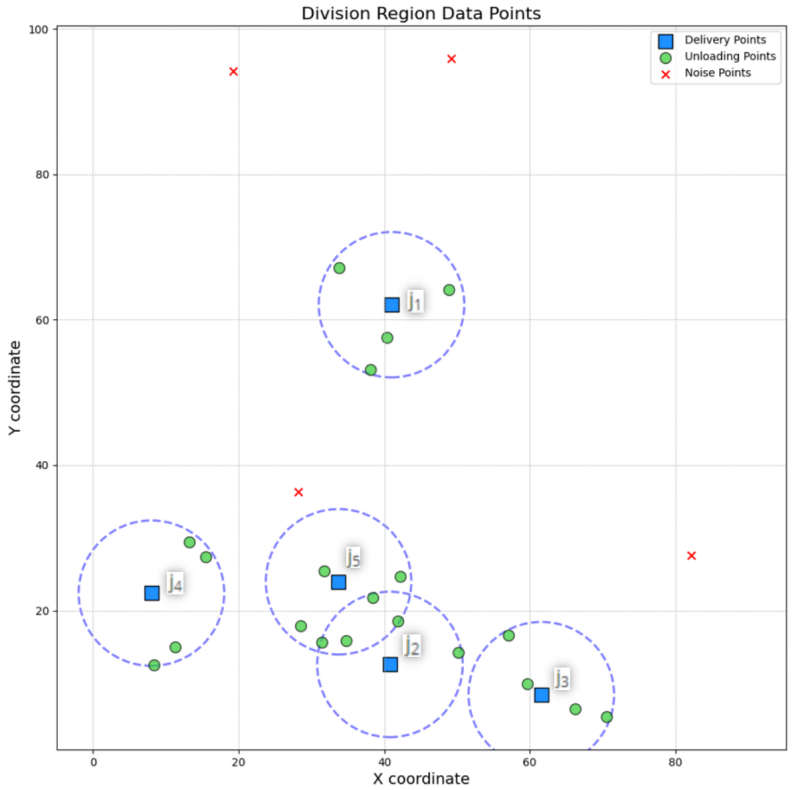

Figure 2: 划分后的配送中心与卸货点分布

3 实验

这一节将详细描述问题的数学建模过程以及算法流程。

3.1 责任圆的生成以及可视化

依据第 1 节的假设条件 (2)、条件 (4) 以及条件 (5)，即无人机一次最远的飞行路程为 20 公里，且任一配送中心都有无限数量的无人机和满足所有用户所需的货物。我们将问题简化，以配送中心为圆心，10 公里为半径画圆。我们将在该圆内的卸货点称为该配送中心的“责任点”，这个圆称为配送中心的“责任圆”，今后这些卸货点的配送任务将交由该配送中心负责，如 Figure 2。同时，有些点无法被任何配送中心的责任圆所覆盖，我们暂时不考虑这些点，将其视为噪声点。我们返回所有配送中心的坐标，如 Table 1 所示。

我们将配送中心表示为 j_i，i 为配送中心对应的责任圆。用 d 表示距离，$d(j,k)$ 即为配送中心 j 与卸货中心 k 之间的距离。

如果无人机的配送区域相互独立，不存在交集，我们分而治之，独立的处理每一个配送区域即可。这里为了不失一般性，我们在生成数据点的时候选取了无人机配送区域存在交集的情况。对于这种情况，我们需要做特殊的处理，处理思路采用贪心算法，即当某一卸货点同时处于两个配

配送中心	坐标
1	[41.00483389 62.06192927]
2	[40.78822419 12.58522706]
3	[61.58471335 8.43218232]
4	[8.01787758 22.3891286]
5	[33.72449907 23.95258972]

Table 1: 五个配送中心与其对应的二维坐标

送中心的配送区域时，该卸货点由距离其更近的配送中心负责。为此，我们首先需要计算任意两个配送中心之间的距离 $d(j_m, j_n)$，如果 $d(j_m, j_n) < 20$ 则表明这两个责任圆之间存在交集，需要特殊处理。代码如下：

```python
def calculate_distances(points):
    """
    计算点集内所有点之间的距离矩阵。
    """
    return np.sqrt(((points[:, np.newaxis, :] - points[np.newaxis, :, :]) ** 2).sum(axis = 2))
```

返回结果如下，从结果可以看出配送中心 j_2, j_5 之间的距离小于 20。

```
[[ 0.       49.47717637 57.44285158 51.59525559 38.79851848]
 [49.47717637 0.       21.20711536 34.20543966 13.38331597]
 [57.44285158 21.20711536 0.     55.35523682 31.89160682]
 [51.59525559 34.20543966 55.35523682 0.   25.75412198]
 [38.79851848 13.38331597 31.89160682 25.75412198 0. ]]
```

因此，我们找出两个责任圆共同拥有的卸货点，j_2, j_5 共同拥有的卸货点如下：

```
[[31.38651741 15.63868617]
 [34.77839174 15.85826984]
 [38.44347605 21.76962673]
 [41.89185044 18.56368174]]
```

然后计算任意一个卸货点和两个配送中心之间的距离。比较这些距离，得出最终的划分结果，见 Figure 3。从图中可以看出 j_2 配送中心只辐射到其责任圆内的 3 个卸货点，j_5 辐射到其责任圆内的 5 个卸货点。后续我们将依据此划分结果进行算法演示。

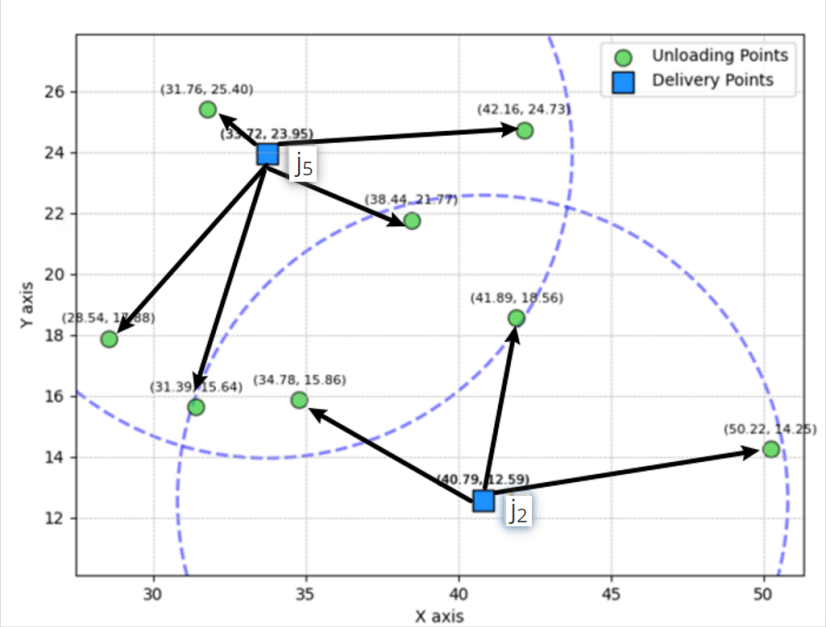

Figure 3: 对于 j_2 和 j_5 责任圆及其卸货点的最终划分

3.2 数学建模

在一个责任圆内，无人机的飞行路线是不受地面交通线路制约的。所以我们可以将责任圆的配送中心点与责任点是做图节点，并构造一个完全图（完全图的任意两个节点都有边）。因此，我们可以先将该问题视为一个旅行商问题。但是需要注意的是，我们研究的问题还受很多给定因素的制约。结合第 1 节的叙述，可能的影响因素 f 分别是假设条件（1）、(2)、(3)、(6)、(7)。

相应的，我们需要解决的问题主要有以下几点：

- **问题 1：**由于责任圆中的卸货点并非时刻都有订单，因此无人机的每次配送并非遍历责任圆中的所有卸货点。那么如何确定无人机每次需要送货的点？
- **问题 2：**无人机一次携带的物品数量有上限，我们确定无人机每次配送时携带的物品数量以及如何确定需要发送几架无人机？
- **问题 3：**虽然我们在 3.1 对问题进行了简化，依据无人机的最短路程 20 公里将所有卸货点进行了划分，但是无人机的整个配送路径长度是否会超过 20 公里？答案是可能的。因此，我们如何解决无人机配送过程中配送路径可能超过 20 公里的问题？

如果上述三个问题可以解决，我们就可以利用解决旅行商回路问题的算法对本问题进行求解。下面，我们给出解决上述三个问题的思路。为了叙述简单并不失一般性，以下的讨论我们将重点集中在一个配送中心及其责任圆，如 Figure 4 所示。

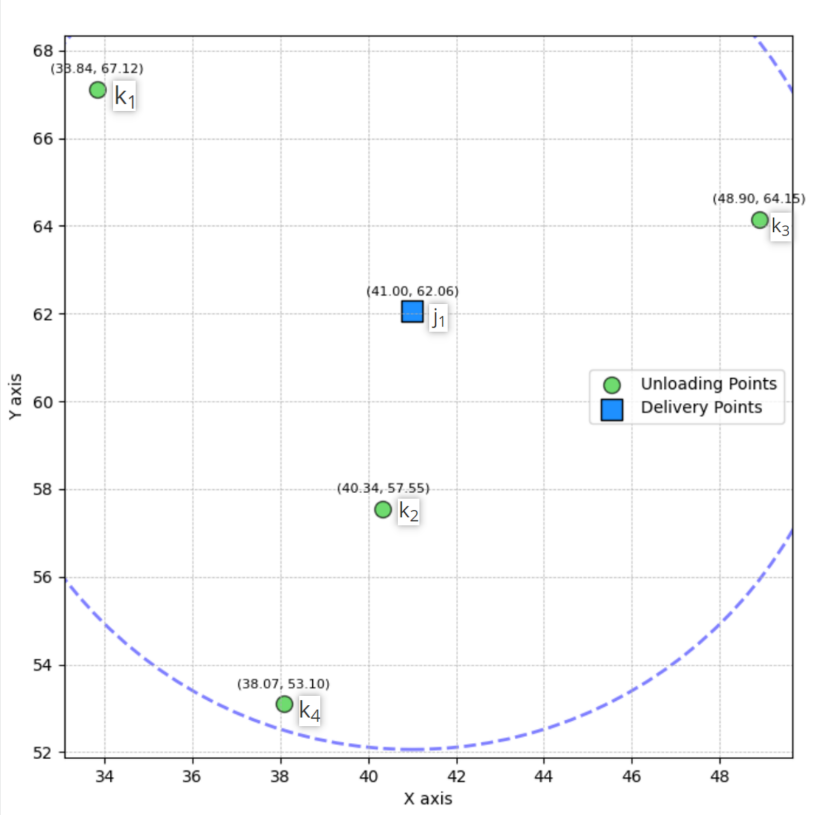

Figure 4: 配送中心 j_1 及其责任圆

3.2.1 问题 1 的解决思路

由第 1 节的假设条件（3）无人机的速度为 60 公里/小时、以及我们的配送中心 j_1 到其他卸货点 k_i 之间的距离，我们可以计算出无人机从配送中心到卸货点 $k_i, i = 1, 2, 3, 4$ 的最短时间。最短时间即无人机直达卸货点的时间。时间计算结果如下，单位为分钟。

```
[ 0.    8.77401456   4.56065279    8.16838906    9.4305167 ]
```

根据第 1 节的假设条件（6）和（7）有如下限制：

（6）每个卸货点每隔一段时间（假设为 t 分钟）会随机生成 m 个订单，一个订单只有一个商品;

（7）订单有优先级别，分为三个优先级别（用户下订单时，会选择优先级别，优先级别高的付费高）：一般：3 小时（180 分钟）内配送到即可；较急：1.5 小时（90 分钟内）配送到即可；紧急：0.5 小时（30 分钟内）配送到即可。

我们采用贪心算法的思路解决问题 1。我们设 m = 1，t=10。在生成订单后，无人机并不立即进行派送，而是等待至不得不进行配送的时刻。这样可以确保无人机每次配送都尽可能地携带更多地商品。我们将无人机不得不进行配送的时刻（即无人机发送的时刻记为 T_i,i 指无人机向卸货点 k_i 进行配送）。则

T_i= 某一商品的剩等待配送时间-无人机到该商品对应的卸货点所用的最短时间

3.2.2 问题 2 的解决思路

在确定了无人机需要发送的时刻后，我们需要解决问题 2，即无人机每次发送应该携带多少数量的商品? 我们依然采用贪心算法的思路，即当遇到 T_i 时，无人机总是携带其能携带的最大数量的商品。我们记当下需要携带的商品数量为 n_{now}，记无人机最多携带的商品数量为 n，记发送的无人机数量为 n_{luanch}。则有

$$n_{luanch} = \lceil n_{now}/n \rceil \tag{1}$$

3.2.3 问题 3 的解决思路

在解决了问题 2 以及问题 3 后，我们便得到了某一个无人机在当前时刻需要进行配送的商品以及该商品对应的卸货点。我们用这些卸货点和配送中心点建立一个完全子图，然后在该子图上处理一个旅行商回路问题。如果我们在该子图上得到的最短回路小于 20 公里，则发送无人机，否则将子图拆解为更小的包含配送中心的公共子图，在公共子图上继续迭代上述过程，直到满足限制条件。

3.3 算法描述

在 3.2 节建立好数学模型之后，关键问题是解一个旅行商回路问题。在本节，我们将采用动态规划的思想解决这样的问题，并基于 Figure 4 对问题进行实例化，得到一个实际的算法运行结果，并进行可视化。

3.3.1 动态规划解决旅行商回路问题

对旅行商问题的最简单解法是“暴力穷举法”，也就是罗列出所有可能的遍历方式。如果有 n 个城市，其中一个城市作为起点，于是要罗列其他 n-1 个城市的遍历次序就会产生 $(n-1)!$ 种可能。显然“暴力穷举法”不是解决该问题的有效方法，我们多次看到过，对于“最优化”问题，最好应用动态规划的思维来解决。首先需要思考的就是如何将问题分解成子问题。

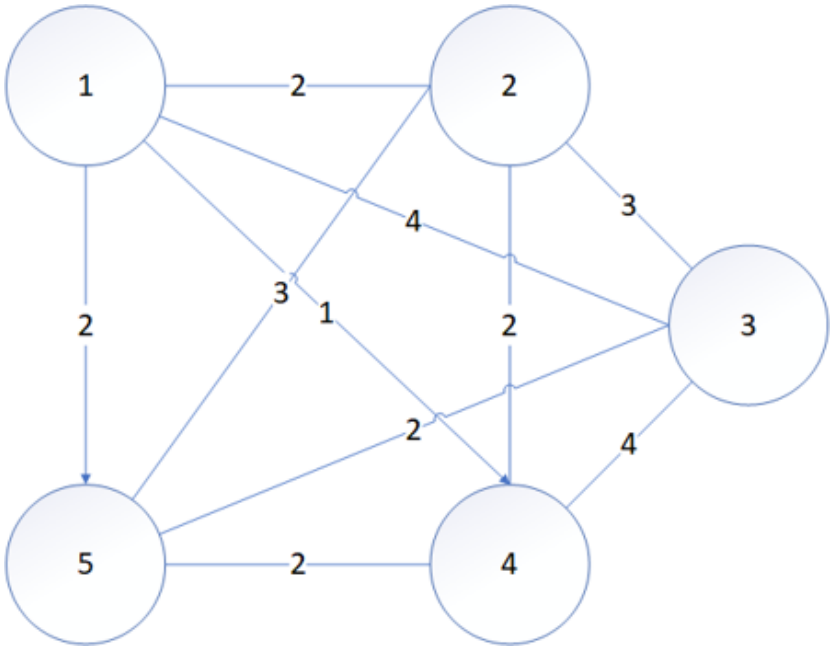

Figure 5: 一个简单的旅行商问题

假设旅行者从节点 1 出发最终回到节点 1，如 Figure 5，中间遍历的最短路径对应其他节点的某种排序情况。我们用集合 $S = 2, 3, 4, ..., n-1$ 表示除去起始节点外的其他节点集合，遍历的最短路径 $1 \rightarrow c_1 \rightarrow c_2 \rightarrow ... \rightarrow c_n \rightarrow 1, c_k \in S$，假设倒数第 2 个节点是 j，那么一定有路径 $1 \rightarrow c_1 \rightarrow c_2 \rightarrow ... \rightarrow j$ 是节点 1 到节点 j 的最短路径。那么节点 j 前面是哪个节点呢? 为了保证节点 1 到节点 j 的距离最短，同时找到节点 j 前面的节点，要遍历除了节点 1 到节点 j 之外所有节点的距离关并加上该节点到节点 j 的距离，哪个能使加总距离最小的节点就是位于节点 j 前面的节点，如果用 $C(S, j)$ 来表示节点 1 到节点 j 的最短路径，那么有

$$C(S, j) = \min_{i \in S, i \neq j} \{C(S - \{j\}, i)\} + d_{ij} \tag{2}$$

式中，d_{ij} 表示节点 i 与节点 j 之间的距离。显然，如果节点集合中只有一个节点，也就是旅行者的起始节点，那么它需要遍历的距离就是 0，于是有 $C(1, 1) = 0$。如果集合含有多个节点，因为最后路径要返回节点 1，因此需要考虑最短路径中从起始节点到倒数第 2 个节点 j 的最短路径，也就是找到节点 j，使得

$$C(S, 1) = \min_{i \in S, j \neq 1} \{C(S - \{1\}, j)\} + d_{j1} \tag{3}$$

然后继续确定节点 j 前面的节点，这样问题就可以不断分解成规模更小的子问题并根据式 (2) 不断递归下去。

3.3.2 算法流程归纳

```
t = 1 # t记录当前时刻，最初t=1，表示第1个10分钟，因为卸货点每隔10分钟生成随机订单
对于每一个配送中心及其责任圆:
    while t!=0 :
        卸货点生成t时刻的随机订单
```

```
        依据3.2节的数学模型进行t时刻配送中心完全子图的构建
            for i in range(len(所有的完全子图):
                依据3.3.1节所示的动态规划算法求解该完全子图的最短回路
        t = t + 1
```

3.3.3 结果展示

我们依据 Figure6 所示的情况进行算法演示。假设此时 k_1, k_2, k_3 生成了新的一个随机订单，k_1 当前存在不得不配送的一个订单。因此，总订单数为 4，k_1 卸货点有 2 个订单，k_2 卸货点有 1 个订单，k_3 卸货点有一个订单。

提示：订单有优先级别，分为三个优先级别（用户下订单时，会选择优先级别，优先级别高的付费高）：一般：3 小时（180 分钟）内配送到即可；较急：1.5 小时（90 分钟内）配送到即可；紧急：0.5 小时（30 分钟内）配送到即可。

假设无人机最多可以携带 4 件商品，则我们共需要派出 1 架无人机。我们先计算无人机遍历 k_1, k_2, k_3 卸货点的最短路径。我们得到的最短路径长度为 39.33，于是我们进行进一步的子图划分。

```
the shortest distance for visiting all points is : 39.3260558
the shortest path for visiting all points is : [3, 2, 1]
```

我们从 k_1, k_2, k_3 随机选择一个一个点设为 k_2，此时 k_1, k_2 构成完全子图。k_3 由于不含有必须进行配送的商品，故退出子图构建。然后，我们在该子图上计算最短路径长度，结果为 24.9046423，让然不满足条件。显然，最后只剩下 k_1，我们只需要对 k_1 进行配送即可。当下最短路径如 Figure6 所示。最后，迭代上述过程即可求出一定时间内所有配送中心的所有无人机的最短路径。

```
the shortest distance for visiting all points is : 24.9046423
the shortest path for visiting all points is : [2, 1]
```

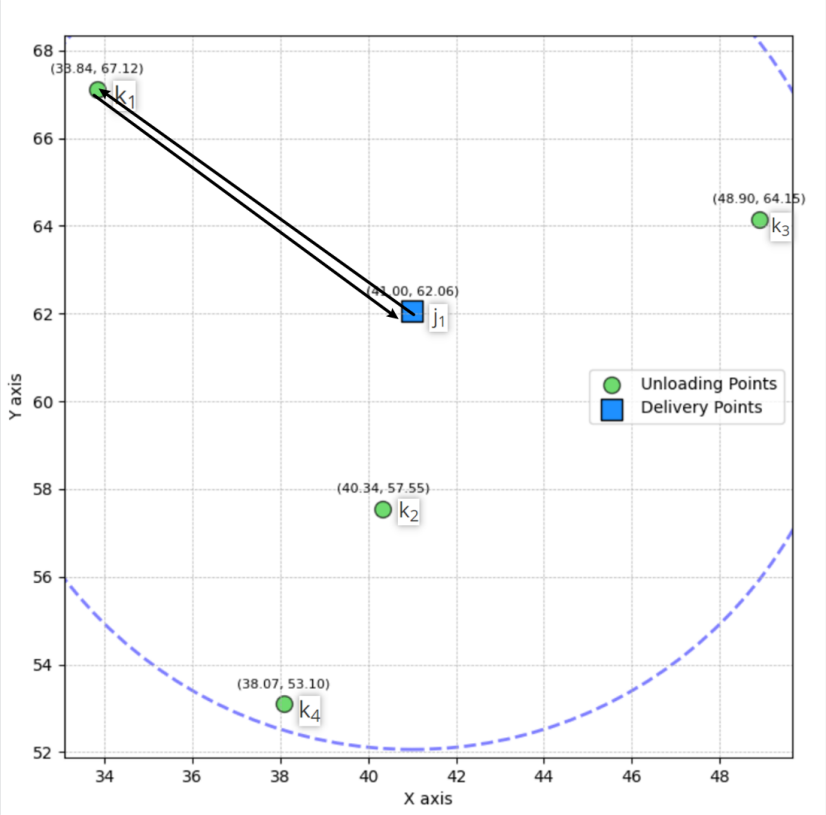

Figure 6: 当下 j1 的最优配送路径

4 总结

研究生阶段深入学习《高级算法设计与分析》这门课程，对我来说是一次理论与实践的深度交融之旅，它不仅拓宽了我的学术视野，更在很大程度上提升了我的问题解决能力和创新思维。以下是我从这门课程中获得的几大感悟与收获：

1. 理论与实践的桥梁高级算法设计与分析课程，将复杂的数学理论与实际问题解决巧妙结合，让我深刻理解到理论知识在实际应用中的重要性和实用性。通过对经典算法的学习，比如贪心算法、高级图算法、近似算法、在线算法，我学会了如何从数学的角度抽象问题，再利用这些算法框架去设计具体解决方案，这一过程极大地锻炼了我的逻辑思维和抽象能力。

2. 复杂性分析的重要性课程中，我们深入探讨了算法的时间复杂度和空间复杂度，以及 NP 完全问题等概念。这使我认识到，在设计算法时，不仅要追求正确性，还要兼顾效率。通过学习不同的复杂性类别，我开始学会评估算法的可行性，这对于未来研究方向的选择和问题的可行性分析具有重大意义。

3. 算法优化的艺术在解决实际问题时，往往需要在多种算法中做出选择，或者对现有算法进行优化以适应特定场景。课程中的案例研究和项目作业，让我亲身体验到了算法优化的过程，从最初的“暴力”解法到逐步引入剪枝、缓存等技巧，最终达到既高效又优雅的解决方案。这种从实践中学习的方式，极大增强了我对算法优化的理解和技能。

针对课程报告可能存在的问题，有以下几点总结：

1. 数学模型的不完善，我们直接将卸货点划分在以配送中心为圆心的责任圆内，这样虽然可以简化问题，但导致的结果是某一个配送中心的无人机必须返回其最初的配送中心。这与现实是

不相符合的。现实生活中，无人机应该可以返回任意一个配送中心。

2. 算法复杂度太高，

该算法的时间复杂度属于指数型，核心在于函数 $get\_subset()$，它要从给定集合中找到所有可能的子集，如果集合原来包含 n 个元素，那么它的所有可能子集个数为 2^n。

对于一个子集而言，给集合中每个元素对应 1 个变量，如果该元素属于子集，那么变量取值为 1; 如果不属于子集，那么变量取值为 0，于是所有子集个数相当于含有 n 个比特位的数值所能表达的整数个数，因此子集的数量为 2^n。于是函数 $get\_subset()$ 的时间复杂度为 $O(2^n)$，获得子集后函数 $find\_shortest\_path0$ 中外层有个 for 循环，次数对应集合中的点数，因此循环次数为 $0(n)$。综合两种情况，算法的时间复杂度为 $O(2^n)$。

在后续过程中，我也将尝试不同的算法对实验进行改进，如遗传算法等