两阶段方法解决带时间窗的多油种多车型成品油运输问题

> 您正在阅读的是 2025 Spring RUC 运筹学建模与算法大作业的报告， 全文共 3852 字，预计需要 15 分钟阅读。
>
> 目录：
>
> - 问题分析
> - 问题建模与算法设计
> - 算法实现细节
> - 实验

问题分析

问题引入与难点分析

> 某地区有2个油库（A、B）和 25 个加油站，需每日根据加油站需求量为3种油品制定配送计划。已知各个加油站每日的需求量、配送开始时加油站与油库的库存量、油库与加油站间的运距、加油站之间的运距、油罐车相关参数（包括容量、车速等），目标是为所有加油站提供配送服务，最小化配送成本并满足各项约束：
>
> 1. 油库A拥有数量有限的油罐车用于配送运输；油库B作为存储点，不配备配送车辆。
> 2. 区域内有大量的加油站，每天每个加油站对每种油品的需求量上下限与最可能值已知。
> 3. 每个加油站拥有多个油罐，分别存储多种油品，加油站油罐在配送开始时拥有一定量的库存。
> 4. 每辆车由隔板划分为两个容积相等的储油仓，每个仓油品不能混装。每个仓对应某个加油站一个油罐的配送。
> 5. 加油站的配送时间既不能早于不容纳时间点（即加油站因油罐的容量限制无法继续接收油品的时间点），又不能晚于断油时间点（即加油站的油品库存量归零）。加油站的油品匀速消耗，每天消耗的油品总量为每天的油品需求量。
> 6. 每个加油站在同一时间只能由一辆车提供配送服务。
> 7. 油罐车允许分载,在同一行程内可以联程配送多个加油站,配送总量不超过油罐车运载能力。
> 8. 油罐车一天内可进行两趟配送，配送开始时间为上午8点（允许延迟出发），结束时间为下午5点。
> 9. 油罐车在每天配送开始时从油库A出发，可在第一趟配送结束后返回油库A或B作为第二趟配送的起点，一天的配送完成后最终返回油库A 。

原问题是一个极其复杂的成品油运输问题，属于车辆路径问题的变种。其复杂的点在于：

1. 油有多个品种，每个加油站都会有自己需求的油量。
2. 油罐车有多个品种，且油罐车还有两个隔舱可以同时运输两种油。
3. 存在中转机制，即一天可以跑两趟，但是必须要回到油仓补给。

- 2、3 难点是极其关键的，传统的建模方法往往难以处理这种可以跑一趟也可以跑两趟，每趟中可以运一次也可以运两次这样复杂的选择，强行处理需要额外引入更高维度的决策变量和更复杂的约束条件，使得传统方法的优化难度越来越高。

4. 加油站存在库存，并且有油罐大小，既不能过晚的到达以至于可能存在真空期，也不能过早的到达，如果过早到达导致没有地方当下油。
    - 这一变量的引入使得问题从简单的整数规划问题变成了混合整数问题，而且由于到达时间可能影响运几趟，在哪里中转的考量，这一变量往往需要全局的统筹，并非简单的约束条件可以描述。

传统方法

传统的建模方法一般是 **混合整数规划** 的建模方式，典型的一种思路是：

以 x_{ijk}^t 表示 k 车辆在第 t 趟运次中是否要从 i 地点到达 j 地点。

然后引入一些辅助变量例如每次运的油种类，在何处中转等等。

最终优化一个如下的目标函数：

$$min \sum_i \sum_j \sum_k \sum_t x_{ijk}^t D_{ij} C_k \tag{2}$$

其中 D 表示两地的距离，C 表示 k 车单位距离的运价

需要引入的关键约束是：

1. 车辆路径的连续性约束
2. 时间线约束和时间窗约束
3. 运量和每次小于两仓的逻辑约束
4. 加油站需求必须满足的约束

然而，传统方法的劣势在于处理车辆路径的时候，每一趟运几次以及有几趟都是不确定的，如果要引入额外的变量会使的复杂度显著提升，这一部分的内容已经在问题的难点分析中提到。

新的两阶段方案

先前的分析中已经提到，各种多趟，每趟多少次的不确定带来难点的关键在于路径连续性、时间线的约束不能简单保持原样。换而言之，由于任务总量的不固定，使得处理路径规划的约束变得难以统一的表述。很自然的，我们提出了一种两阶段的方法，如果每个车辆的任务数量不固定，那么我们就先确定每辆车辆先分配到任务，再根据分配到的任务单独的寻路。

具体而言，在准备阶段，我们将所有的需求全部抽象成独立的任务，每个任务由三个关键参数唯一确定：**加油站**、**配送油种**、**时间窗**。至于油量，基于数据的洞察表明，所有任务都可以在一个油舱内满足，只不过有些任务不可以被大油舱的车辆运输，这一部分我们可以通过时间窗表示；在第二阶段，我们根据分配到的任务，规划出在所有可能的路径中的最优的路线，需要同时满足到达时间的时间窗约束，配送路线的可行性。

问题建模和算法设计

阶段一：确定任务分配

在两阶段的方案中，在第一阶段，我们优化一个以下的 **带约束的指派问题** 任务：

$$
\begin{aligned}
&min \sum_{v \in V} cost(x_v) \\
s.t &\begin{cases} \sum_{t \in T} x_{tv} \leq 4, \forall v \in V \\ \sum_{v \in V} x_{tv} = 1, \forall t \in T \\ x_{tv} = 0, 1 \end{cases}
\end{aligned} \tag{3}
$$

其中 x 为决策变量，也就是一个指派矩阵，x_{tv} 表示 t 任务是否由 v 车辆来做，cost 为优化后每辆车在获得所有的任务后规划得到的最优代价。

在第二阶段，由于已经确定任务总量最多唯一四个，实际上我们可以穷举所有可能的路径，也就是任务的全排列，并且在其中插入可行的中转点，在至多 96 种组合中选择一个最小的（事实上由于大部分路径可能完全不可行所以计算成本并不高）。

阶段二：确定任务下的最优寻路

由于第一阶段优化任务的代价函数必须在整体方案确定后才可以实现（我们不能假定一个车辆之后不再会收到任务），因此两阶段任务必须在同一轮迭代中分别优化。

第二阶段的优化相对简单，只需要设计穷举即可。第一阶段是一个整数规划问题，但是传统的 0-1 规划并不能很好的解决这个问题，这是因为传统的 0-1 规划本质上还是穷举法，难以适应指派问题这样的搜索空间巨大的问题，因此我们引入了一种 **遗传算法** 思路：

1. 首先编码分配方案为一个向量，长度等于任务数，每个位置表示一个任务归属的车辆。
2. 生成优秀的亲代，为了保证亲代大概率可行，我们先给每个车辆至多安排两个任务，如此无论两个任务节点之间有无通路，经过一次中转永远可以走通，切时间窗重叠的概率也较小。
3. 在选择策略上，我们将任务分配完后，由每一个车辆单独运行寻路算法，得到最小的代价后叠加，取相反数作为适应度（我们要最小化代价）。至于选择的策略，我们采用了 **锦标赛选择** 的方法，即随机抽样多组个体，每组选举其中适应度最高的个体。
4. 在子代生成的策略上，我们采用精英保留和随机交叉结合的思想，由于搜索空间较大，我们不希望下一代比前一代还要差，所以保留适应程度最好的固定个亲代，并由固定的交叉方法生成新的子代。
5. 为了保证种群的多样性，我们采用了一个小概率的基因变异率，这保证我们不会过度局限于局部极小值。
6. 最后，为了保证每辆车最多分到 4 个任务，我们采用了一种 **修复性策略**，每当监测到亲代中有一辆车被分配了超过四个任务，将这个个体进行单独修复，将超额任务随机转移到没有满任务的车辆上。

基于以上的遗传算法设计，循环一个大的轮数，可以基本保证达到最优。

伪代码

具体而言，我们的算法将以如下伪代码的思路呈现：

```
// STAGE 1. PREPARATION
Tasks <- Initilize by split all requirement into independent task
Vehicles <- Initilize by reading from sheet
Individuals <- InitIndividuals()
best_cost <- INTINITY
Constant Population

// Main Loop
Iteration:
    best_cost, best_chromosome <- min(min(Cost(for each Individual)), best_cost)
    While size of next_generation < Population:
        Select 2 parents use Tournament Selection
        child <- Repair(Mutate(CrossOver(parents)))
        next_generation.append(child)
```

算法实现细节

由于上文已经完整描述了算法工作过程，并且所有代码都已经上传，我们不再展开介绍算法的完整实现，我们主要关注在实现层面中总体的思想和一些需要指出的细节。

为了实现这个两阶段方法，我们将每个任务和每个车辆都封装成了单独的实体类，每个车辆都有独立的任务列表，并且车辆类内部由规划路线这一对外接口。

我们将主要的第一阶段实现封装到了遗传算法类中，这个类接受必要的数据和超参数，并且提供对外的启动接口。

如此我们才用了分层的策略，这与两阶段算法本身契合。

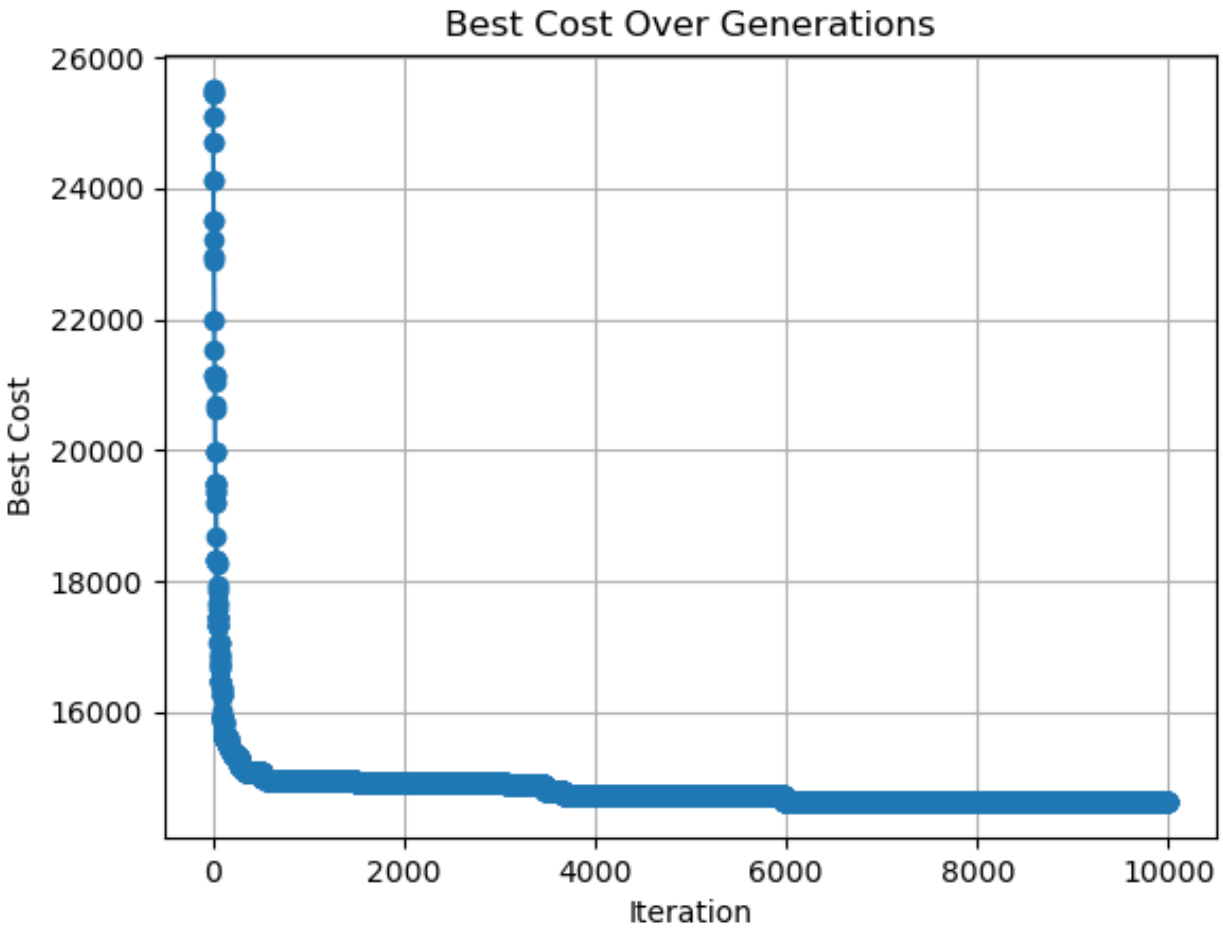
以下是一些上文中未提及但是在设计中着重注意的一些细节：

1. 我们在计算时间窗的时候对于最早和最晚到达的时间计算采取了动态的策略，也就是需要结合具体需求量来决定，这个需求量是比较弹性的，我们在计算最早到达时间是假设需求量极大，计算最晚到达时间的时候假设需求量极小。
2. 如果车辆早于最早时间抵达，我们还需要额外等待一个固定的时间，为了最小化这个时间，我们假设车辆运输量刚好等于需求量。

实验

根据我们设计的算法和代码实现，我们进行了实验得到如下结果：

经过 10000 轮迭代，以下是最优代价随迭代轮数的变化趋势，可以看到，最优代价呈现出了较好和较快的收敛趋势。

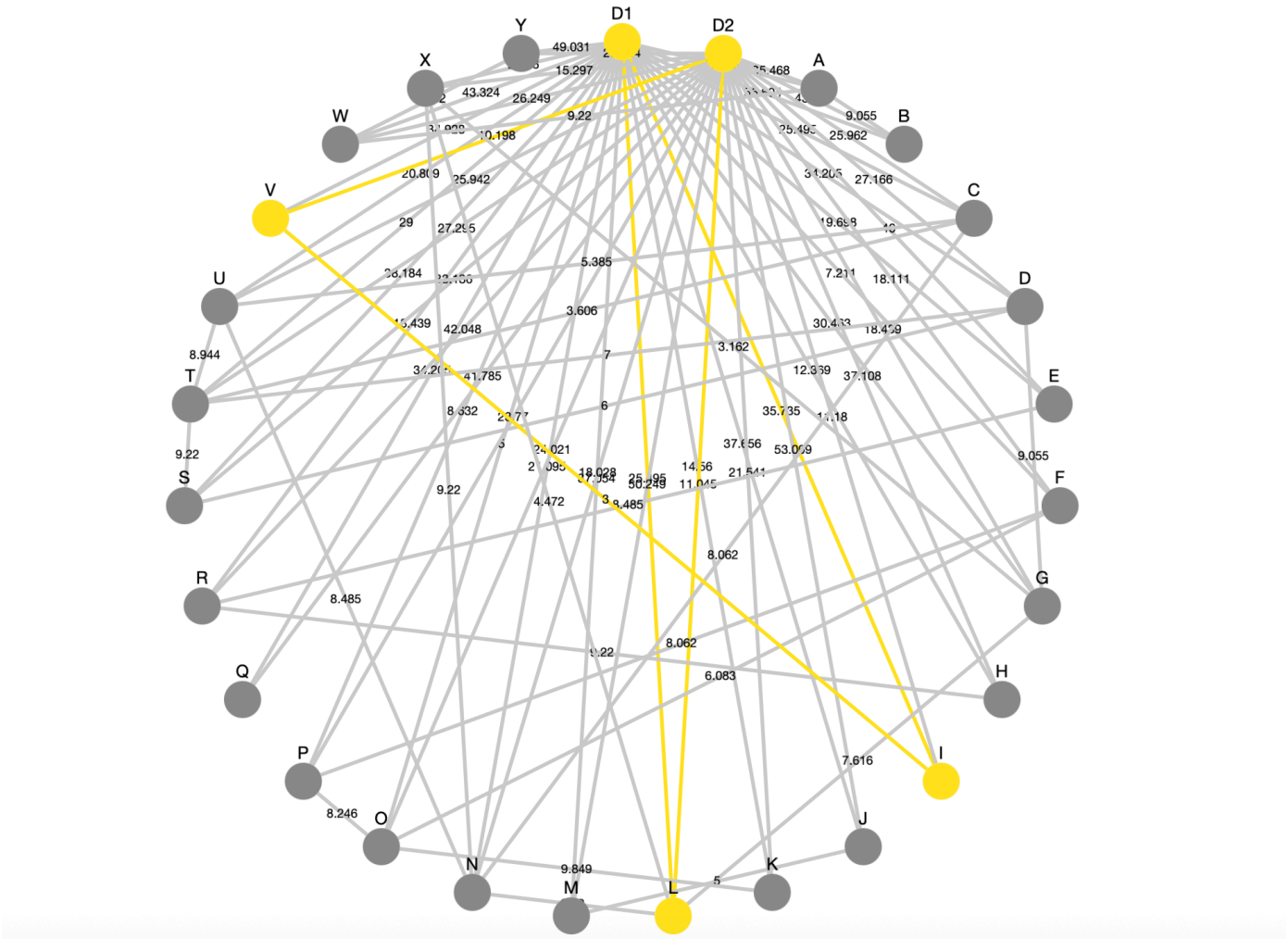


以下是最优的规划方案的描述版本，最优代价为：**14606.59**。

```
车辆 0: 3 个任务, 任务顺序: 17 (油种: 2) 17 (油种: 3) A (油种: 0) 8 (油种: 2)
车辆 1: 4 个任务, 任务顺序: 23 (油种: 3) 25 (油种: 2) B (油种: 0) 20 (油种: 1) 3 (油种: 2)
车辆 5: 2 个任务, 任务顺序: 6 (油种: 1) 6 (油种: 3)
车辆 6: 4 个任务, 任务顺序: 2 (油种: 1) 1 (油种: 1) B (油种: 0) 24 (油种: 1) 7 (油种: 3)
车辆 7: 4 个任务, 任务顺序: 3 (油种: 1) 3 (油种: 3) B (油种: 0) 23 (油种: 1) 23 (油种: 2)
车辆 8: 4 个任务, 任务顺序: 13 (油种: 2) 13 (油种: 3) A (油种: 0) 5 (油种: 1) 18 (油种: 3)
车辆 9: 4 个任务, 任务顺序: 15 (油种: 1) 15 (油种: 2) A (油种: 0) 19 (油种: 2) 19 (油种: 3)
车辆 10: 4 个任务, 任务顺序: 21 (油种: 2) 21 (油种: 3) B (油种: 0) 25 (油种: 1) 25 (油种: 3)
车辆 17: 4 个任务, 任务顺序: 4 (油种: 1) 4 (油种: 3) B (油种: 0) 22 (油种: 1) 22 (油种: 3)
车辆 20: 4 个任务, 任务顺序: 19 (油种: 1) 4 (油种: 2) B (油种: 0) 12 (油种: 2) 12 (油种: 3)
车辆 22: 4 个任务, 任务顺序: 1 (油种: 2) 1 (油种: 3) B (油种: 0) 7 (油种: 1) 7 (油种: 2)
车辆 25: 2 个任务, 任务顺序: 18 (油种: 1) 18 (油种: 2)
车辆 27: 4 个任务, 任务顺序: 9 (油种: 1) 9 (油种: 3) B (油种: 0) 6 (油种: 2) 16 (油种: 2)
车辆 28: 4 个任务, 任务顺序: 14 (油种: 2) 14 (油种: 3) B (油种: 0) 24 (油种: 2) 24 (油种: 3)
车辆 29: 3 个任务, 任务顺序: 17 (油种: 1) A (油种: 0) 11 (油种: 1) 11 (油种: 3)
车辆 30: 4 个任务, 任务顺序: 2 (油种: 2) 2 (油种: 3) B (油种: 0) 14 (油种: 1) 21 (油种: 1)
车辆 31: 2 个任务, 任务顺序: 10 (油种: 3) 13 (油种: 1)
车辆 32: 2 个任务, 任务顺序: 16 (油种: 1) 16 (油种: 3)
车辆 33: 4 个任务, 任务顺序: 8 (油种: 1) 8 (油种: 3) A (油种: 0) 20 (油种: 2) 20 (油种: 3)
车辆 34: 2 个任务, 任务顺序: 5 (油种: 2) 5 (油种: 3)
车辆 35: 2 个任务, 任务顺序: 10 (油种: 1) 10 (油种: 2)
车辆 38: 2 个任务, 任务顺序: 11 (油种: 2) 15 (油种: 3)
```

```
车辆 39：3 个任务，任务顺序：9 (油种：2) 22 (油种：2) B (油种：0) 12 (油种：1)
```

以下是最优方案的可视化版本，如图展现了车辆 39 的行动路线，D1 代表油舱 A，加油站编码按编号顺序排列：



所有实验所用的数据和代码均可在 https://github.com/YichenShen0103/VRPTW-MC 访问。