Standard CVRPSPD-列生成尝试

原始模型

集合

名称	集合体	描述
N	$\{n1, n2, ...\}$	客户点的集合
V	$\{0, n1, n2, ...\}$	所有结点（客户 $+$ $depot$）的集合
K	$\{k1, k2, ...\}$	车辆的集合

参数

名称	单位	描述
q_i^+	$[-]$	客户i的取货需求
q_i^-	$[-]$	客户i的送货需求
c_{ij}	$[-]$	节点i到j的行驶成本（距离/时间）
Q	$[-]$	车辆的最大载重量（先不考虑异构）
v	$[-]$	车辆速度
st_i	$[-]$	车辆在客户i的服务时间
tm	$[-]$	车辆的最长在途时间

变量

名称	类型	描述
$x_{i,j,k}$	$Binary$	若车辆k从节点i行驶到节点j，则为1，否则为0 (注：车辆可以从仓库出发再直接回到仓库，即该车辆不使用；但在客户节点不可以)
$u_{i,k}$	$NonNegativeReals$	车辆k在节点i时的载货量（已完成取配）

约束

1. 每个客户被访问一次

$$\sum_{k\in K, j\in V, j\neq i} x_{ijk} = 1 \quad \forall i \in N \tag{1}$$

2. 车辆流平衡

$$\sum_{j \in V} x_{ijk} = \sum_{j \in V} x_{jik} \quad \forall i \in V, k \in K \tag{2}$$

3. 每辆车从depot出来，回到depot

$$\sum_{j \in V} x_{0jk} = \sum_{j \in V} x_{j0k} \quad \forall k \in K \tag{3}$$

4. 货量递推公式

$$u_{j,k} \geq u_{i,k} + q_j^+ - q_j^- - M \cdot (1 - x_{i,j,k}) \quad \forall k \in K, i \in V, j \in N, i \neq j \tag{4}$$

$$u_{j,k} \leq u_{i,k} + q_j^+ - q_j^- + M \cdot (1 - x_{i,j,k}) \quad \forall k \in K, i \in V, j \in N, i \neq j \tag{5}$$

5. **初始载货量**：车辆从车场出发时需携带足够货物：

$$u_{0,k} = \sum_{i \in N} q_i^- \cdot \sum_{j \in V, i \neq j} x_{i,i,k} \quad \forall k \in K \tag{6}$$

6. 车辆载重约束

$$u_{i,k} \leq Q \quad \forall k \in K, i \in V \tag{7}$$

7. 车辆最长在途时间约束

$$\sum_{i \in V, j \in V} c_{i,j} \cdot x_{i,j,k} + \sum_{i \in N} st_i \left(\sum_{j \in V} x_{i,j,k}\right) \leq tm \quad \forall k \in K \tag{8}$$

目标

最小化车辆总行驶成本：

$$min \sum_{i \in V, j \in V, k \in K} c_{i,j} \cdot x_{i,j,k} \tag{9}$$

cg-列生成

集合

名称	集合体	描述
M	$\{m1, m2, ...\}$	车型集合
K	$\{k1, k2, ...\}$	车辆集合

参数

名称	单位	描述
Q	$[-]$	车辆的最大载重量

主问题

子问题提供的集合

名称	集合体	描述
P	$\{p1, p2, ...\}$	子问题提供的方案集合

子问题提供的参数

名称	单位	描述
$x_{p,i}$	$[-]$	p方案中是否经过客户i；如果是则为1，否则为0
$u_{p,i}$	$[-]$	车辆在p方案中在节点i处的载货量
c_p	$[-]$	方案p的距离总成本

变量

名称	类型	描述
S_p	$Binary$	是否采取方案p(此处不区分不同车型的方案池)

约束

1. 每个客户被访问一次

$$\sum_{p \in P} x_{p,i} \cdot S_p = 1 \quad \forall i \in N \tag{10}$$

2. 车辆总数约束

$$\sum_{p \in P} S_p \leq |K| \tag{11}$$

目标

最小化车辆总行驶成本：

$$min \sum_{p \in P} c_p \cdot S_p \tag{12}$$

子问题

（此处只考虑相同车型，用同一个子问题提供的方案池）

变量

名称	类型	描述
$x_{i,j}$	$Binary$	若车辆从节点i行驶到节点j，则为1，否则为0
u_i	$NonNegativeReals$	车辆在节点i处的载货量

约束

1. 每个节点最多访问一次

$$\sum_{j \in V, j \neq i} x_{ij} \leq 1 \quad \forall i \in V \tag{13}$$

2. 车辆流平衡

$$\sum_{j \in V} x_{ij} = \sum_{j \in V} x_{ji} \quad \forall i \in N \tag{14}$$

3. 每辆车从depot出来，回到depot

$$\sum_{j \in N} x_{0j} = \sum_{j \in N} x_{j0} = 1 \tag{15}$$

注：这里规定一定要从depot出发去一个客户点，避免生成直接从depot到depot的无效路径

4. 货量递推公式

$$u_j \geq u_i + q_j^+ - q_j^- - M \cdot (1 - x_{i,j}) \quad \forall i \in V, j \in N, i \neq j \tag{16}$$
$$u_j \leq u_i + q_j^+ - q_j^- + M \cdot (1 - x_{i,j}) \quad \forall i \in V, j \in N, i \neq j \tag{17}$$

5. **初始载货量**：车辆从车场出发时需携带足够货物：

$$u_0 = \sum_{i \in N} q_i^- \cdot \sum_{j \in V, i \neq j} x_{i,j} \tag{18}$$

6. 车辆载重约束

$$u_i \leq Q \quad \forall i \in V \tag{19}$$

7. 车辆最长在途时间约束

$$\sum_{i \in V, j \in V} c_{i,j} \cdot x_{i,j} + \sum_{i \in N} st_i (\sum_{j \in V} x_{i,j}) \leq tm \tag{20}$$

目标

设 π_i , θ 是 (9)(10)的对偶参数:

$$min \sum_{i \in V, j \in V} c_{i,j} \cdot x_{i,j} - \sum_{i \in N, j \in V} \pi_i \cdot x_{i,j} + \theta \tag{21}$$

子问题（标号法生成路径）

名词定义

优先队列（堆）：

- **用途**：存放一系列标签，标签按路径成本排序，之后从里面提取排第一的标签用于循环
- **元素**：元组 `(cost, id(label), label)`

标签：

表示从车场出发的部分路径状态，包含：

```
{
    "node": 当前节点,
    "path": 路径序列（如 已返回车厂 [0,1,3,0] 或者 未返回车厂 [0,1,3]），
    "visited": 已访问节点集合,
    "initial_load": 初始载货量（出发时装载量），
    "remaining_load": 当前剩余载货量（走完path之后车上留存的载货量），
    "total_delivery": 路径总配送需求,
    "total_pickup": 路径总取货需求,
    "cost": 路径累计成本
}
```

支配规则：

现有标签 `existing` 支配新标签 `new` 当且仅当：

1. 到达同一节点（`existing["node"] == new["node"]`）
2. 现有标签成本更低（`existing["cost"] <= new["cost"]`）
3. 初始载货量更少（`existing["initial_load"] <= new["initial_load"]`）
4. 剩余载货量更多（`existing["remaining_load"] >= new["remaining_load"]`）
5. 访问节点是子集（`existing["visited"] ⊆ new["visited"]`）

步骤 1：初始化

- **创建优先队列**：存入初始标签（从车场0出发）
- **初始化支配字典**：用于记录每个节点的已有的标签

步骤 2：主循环

```
while heap不为空:
    1. 弹出当前成本最低的标签
    2. 剪枝检查:
        - 若标签被支配 → 跳过
        - 否则 → 加入支配字典
    3. 遍历所有可能的下一个节点:
        a. 跳过已访问节点和当前节点
        b. 扩展新标签（调用extend_label）
        c. 若扩展失败（容量超限） → 跳过
        d. 若回到车场:
            - 检查初始载货量是否满足总配送需求
            - 计算缩减成本，若为负则加入可行路径
        e. 否则 → 检验该新标签是否已经被支配，如果未被支配则假如heap
```

步骤 3：返回结果

返回所有找到的可行路径及其缩减成本

测试结果

data.zip
55.93KB

数据集	车辆数量	车辆容量	原模型 整数解	cg模型 松弛解	cg模型 整数解
cap_80	3	80	464.03	429.11	500.06
cap_90	3	90	412.70	378.22	513.37
cap_100	3	100	412.70	378.22	513.37
cap_150	3	150	317.34	317.34	317.34
cap_200	3	200	317.34	317.34	317.34