基于整数规划的订单协同配送方案研究

摘　　要

近年来，随着电商行业的蓬勃发展，快递物流等车辆路径调度问题的求解和优化逐渐成为一个重要的议题。本文建立了 0-1 整数规划模型，并使用动态规划的贪心算法和优化后的分支定界算法来优化运输方案。

针对问题一，基于所给订单和每辆运输车最多同时携带的订单件数不超过 1 件的约束条件，构造目标函数为最小停车次数的整数规划模型。并且类比背包问题模型，采用贪心算法进行求解，引入车次价值等概念，通过建立以总车次价值为最高的最优化目标函数，来权衡对于同一辆车次的订单数和停车次数。通过模型的计算求解，附录中实例 1 至 4，运输车最小停车次数分别为 82、159、162、636。

针对问题二，基于所给订单和每辆运输车最多同时携带的订单件数不超过 2 件的约束条件，构造目标函数为最小停车次数的整数规划模型。使用精确算法分支定界法求解，并在求解前通过问题一确定车辆数上限和固定订单数为 0 的变量来减少分支定界法的规模，从而提高分支定界法求解效率。通过模型的计算求解，附录中实例 1 至 4，运输车最小停车次数分别为 51、114、129、516。

针对问题三，基于所给订单、每辆运输车最多同时携带的订单件数不超过 2 件和每辆车在运输过程中不能空载的约束条件，构造目标函数为最小车辆数的整数规划模型。使用优化后的分支定界法求解。通过模型的计算求解，附录中实例 1 至 4，最小运输车车辆数分别为 16、28、29、104。

针对问题四，根据题意使用线性加权法构造运输车辆总数和总停车次数的评价函数，基于所给订单和每辆运输车最多同时携带的订单件数不超过 2 件的约束条件，构造以评价得分为目标函数的整数规划模型。使用优化后的分支定界法求解。通过模型的计算求解，附录中实例 1 至 4，运输车最小停车次数分别为 58、120、134、531；运输车车辆数分别为 16、30、35、122。

关键词：整数规划　贪心算法　分支定界法　线性加权法

一、 问题介绍

1.1 研究背景

随着网购和物流服务的发展，物流或快递公司的配送站点已经遍布各个城市，以满足人们取件寄件的需要。为了向大众提供优质的物流服务，企业在降低物流成本的同时，也要设计最优的订单协同配送方案，提高物流的配送速度。

对订单的不同的车次选择，会对物流公司造成不同的影响，合理的订单分配运输让顾客更早的接受自己的订单能使物流公司享有更高的声誉；而不合理甚至是错误的运输则会使物流公司蒙受损失，或是经济利益上的损失抑或声誉等。因此，如何合理地调用资源、解决订单协同配送问题，已经成为了一个研究的重要课题。

1.2 问题重述

快递公司 K 在城市 H 内有 n 个站点，在某时间段内有 m 件订单需要在这些站点之间进行运输。所有站点从小到大排列在一条直线上并有各自的标号，每个订单都有自己的初始站点和结束站点，并且每个订单的起始站点编号小于其结束站点编号。

每个订单从其初始站点出发通过运输车配送到其结束站点后，运输车停车，订单被搬运下车。若此时这个站点是另一个订单的初始站点，则可以将另一个订单搬运上车继续进行配送。但是每辆运输车的容量有限且相同，最多可以携带 C 个订单。

基于这些信息，建立相应的数学优化模型解决以下的订单协同配送问题。

问题一：

在快递公司运输车个数无限的情况下，针对 C=1 的情形，建立订单协同配送优化模型，极小化所有运输车辆的总停车次数，并利用所给数据进行验证。

问题二：

在快递公司运输车个数无限的情况下，针对 C=2 的情形，建立订单协同配送优化模型，极小化所有运输车辆的总停车次数，并利用所给数据进行验证。

问题三：

在快递公司运输车个数有限的情况下，针对 C=2 的情形，运输车中途不能空载，建立订单协同配送优化模型，极小化运输车总个数，并利用所给数据进行验证。

问题四：

在快递公司运输车个数有限的情况下，针对 C=2 的情形，运输车中途不能空载，同时考虑使用的运输车数量和运输车总停车次数两个优化目标极小化，建立订单协同配送优化模型，并利用所给数据进行验证。

二、 问题分析

问题一分析：

对于其车辆载货量仅为 1 单位订单的情况下，通过 0-1 背包问题来进行求解。订单的数量和车辆的停车次数可以综合考虑为单个背包的最求总价值，而对于这个问题可以转化为对于有限个背包，每个背包中有多个订单，但是背包的限制条件为同一个站点区间内最多只有 1 个单位的背包。所以可以通过调整订单数量和车辆停车次数的加权参数来更好的追求多个背包的总体价值。

问题二分析：

问题二分析，对于每辆运输车最多同时携带的订单件数不超过 2 件的情况下，以最小停车次数为目标构建整数规划模型。再使用精确算法分支定界法求解。

问题三分析：

问题三分析，对于每辆运输车最多同时携带的订单件数不超过 2 件的情况下，以最小停车次数为目标构建整数规划模型。使用优化后的分支定界法求解。

问题四分析：

问题四分析，根据题意构造以极小化所使用运输车辆总个数和极小化所有运输车辆的总停车次数的双目标规划评价函数，对于每辆运输车最多同时携带的订单件数不超过 2 件的情况下，构造目标函数为最小评价分数的整数规划模型。使用优化后的分支定界法求解。

三、 模型假设

(1) 每个订单都有自己的初始站点和结束站点。
(2) 每个订单的初始站点标号小于其结束站点标号。
(3) 多个订单可以具有相同的初始站点和结束站点。
(4) 每辆运输车的容量相同且有限。
(5) 当快递公司 K 的运输车有限时，运输车将其最后一个订单配送到结束站点后，会结束其配送任务。
(6) 所有运输车的行驶方向都是从左往右，不允许调头。

四、 符号说明

符号	文字说明
n	配送站点数
m	物流订单数
l	运输车车辆数
d_{ij}	从第 i 个站点送到第 j 个站点的订单数
$C_{\text{载}}$	每辆运输车的最大携带订单数
T	总停车次数

五、 问题一：C=1 的最小停车数模型

5.1 模型准备

由题意，总订单数为 m，总站点数为 n。本文根据所给订单，构造 $n \times n$ 的订单矩阵 D，d_{ij} 来存储所给数据中从 i 站点上货到 j 站点下货的订单数。矩阵如下：

$$d_{ij} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{bmatrix} \tag{1}$$

对于运输方案，共需 l 辆运输车辆来运输所有订单，故构造 $l \times n \times n$ 的数组 f_{kij} 来存储第 k 辆车负责从 i 站点上货到 j 站点下货的订单数。

5.2 模型建立

5.2.1 目标函数

题目 1 要求，在运输车辆无限多的情况下，极小化所有运输车辆的总停车次数。则目标函数为：

$$T = \sum_{k=1}^{l} \sum_{i=1}^{n} t_{ki} \ (k = 1, 2, \cdots, l \quad i = 1, 2, \cdots, n) \tag{2}$$

其中 t_{ki} 表示第 k 辆车是否需要在 i 站点停车，公式如下：

$$t(k,i) = \begin{cases} 1, t_{\text{上}ki} + t_{\text{下}ki} > 0 \\ 0, t_{\text{上}ki} + t_{\text{下}ki} = 0 \end{cases} \tag{3}$$

其中 $t_{\text{上}ki}$ 表示第 k 辆车是否需要在 i 站点停车上货，$t_{\text{下}ki}$ 表示第 k 辆车是否需要在 i 站点停车下货，公式如下：

$$t_{\text{上}ki} = \begin{cases} 1, \sum_{j=1}^{n} f_{kij} > 0 \\ 0, \sum_{j=i}^{n} f_{kij} = 0 \end{cases} \tag{4}$$

$$t_{\text{下}ki} = \begin{cases} 1, \sum_{i=1}^{n} f_{kij} > 0 \\ 0, \sum_{i=i}^{n} f_{kij} = 0 \end{cases} \tag{5}$$

其中 $\sum_{j=1}^{n} f_{kij}$ 表示第 k 辆车从 i 站点的上货数，$\sum_{i=1}^{n} f_{kij}$ 表示第 k 辆车从 j 站点的上货数。

5.2.2 约束条件

订单限制：

所有车辆运输从 i 站上货到 j 站下货的订单数之和要等于所给数据中从 i 站点上货到 j 站点下货的订单数，表示为：

$$\sum_{k=1}^{l} f_{kij} = d_{ij} \ (k = 1, 2, \cdots, l \quad i = 1, 2, \cdots, n \quad j = 1, 2, \cdots, n) \tag{6}$$

车辆限制：

(1) 车辆能接受运输从 i 站上货到 j 站下货的订单数要小于等于车最大载量 c 载，表示为：

$$f_{kij} \in \{0, 1\} \tag{7}$$

(2) 在运输过程中，车辆上货物的订单数要小于等于车最大载量 c 载，表示为：

$$C_{出发kx} \in \{0, 1\} (k = 1, 2, \cdots, l \quad x = 1, 2, \cdots, n) \tag{8}$$

C_{kx} 表示从 x 站点出发时车车辆上货物的订单数，公式如下：

$$C_{出发kx} = C_{上kx} - C_{下kx} \tag{9}$$

其中 $C_{上ki}$ 表示第 k 辆车是否需要在 i 站点停车上货，$C_{下ki}$ 表示第 k 辆车是否需要在 i 站点停车下货，公式如下：

$$C_{上kx} = \sum_{i=1}^{x} \sum_{j=1}^{n} f_{kij} \tag{10}$$

$$C_{下kx} = \sum_{i=1}^{n} \sum_{j=1}^{x} f_{kij} \tag{11}$$

5.2.3 C=1 的最小停车次数模型

基于以上分析，可以写出 $C_{载} = 1$ 时，基于整数规划的最小停车次数模型：

$$\textbf{min}\ T = \sum_{k=1}^{l}\sum_{i=1}^{n} t_{ki}$$

$$s.t.\begin{cases}
\sum_{k=1}^{l} f_{kij} = d_{ij} \\
C_{\text{上}kx} = \sum_{i=1}^{x}\sum_{j=1}^{n} f_{kij} \\
C_{\text{下}kx} = \sum_{i=1}^{n}\sum_{j=1}^{x} f_{kij}) \\
C_{\text{出发}kx} = C_{\text{上}kx} - C_{\text{下}kx} \\
t_{\text{上}ki} = \begin{cases} 1, \sum_{j=1}^{n} f_{kij} > 0 \\ 0, \sum_{j=i}^{n} f_{kij} = 0 \end{cases} \\
t_{\text{下}ki} = \begin{cases} 1, \sum_{i=1}^{n} f_{kij} > 0 \\ 0, \sum_{i=i}^{n} f_{kij} = 0 \end{cases} \\
t_{ki} = \begin{cases} 1, t_{\text{上}ki} + t_{\text{下}ki} > 0 \\ 0, t_{\text{上}ki} + t_{\text{下}ki} = 0 \end{cases} \\
f_{kij} \in \{0,1\} \\
C_{\text{出发}kx} \in \{0,1\} \\
i = 1,2,\cdots,n \\
j = 1,2,\cdots,n \\
k = 1,2,\cdots,l \\
x = 1,2,\cdots,n
\end{cases} \tag{12}$$

5.3 算法设计

定义 1 (车次价值) 问题一优化问题的目标函数，其具体的等式为：

$$w_k = \alpha_1 C_{order\ count} + \alpha_2 C_{stop\ time} \tag{13}$$

定义 2 (两订单的连续) 对于两订单 $d_{i_1 j_1}, d_{i_2 j_2}$，满足不完全相等且只满足 $i_1 = j_2, i_2 = j_1$ 其中的一个条件。

定义 3 (车次 C_k) 对于一辆车，再开始第一个订单的上货直至在完成所有连续的订单中的行车称为 C_k 车次，其中要求对于一辆车所接受的所有订单是连续的。

定义 4 (停车数) 具体的车辆在站点中停止进行上下货的次数总和，若车辆停止于某一特定站点进行多次上下货，均视为一次停车。

定义 5 (订单数) 具体的车次车辆中不同的订单的数目总和。

可以认为 0-1 背包即每个订单的数量为 1，对于车次 C_k 而言，可以选择放或者不放。

对于 m 个订单，每个订单 a_ij 占据了车辆从站点 d_i 至 d_j 的 1 单位容量，我们另 a_ij 为一个 1 维向量，满足：

$$d(i,j)=\begin{cases}0, x\in[i,j]\\1, x\notin[i,j]\end{cases} \tag{14}$$

而每个订单对于背包（即车次 C_k）的价值的贡献在于其提供的 1 单位订单数目和加入假如时所能改变的停车时间。

若用 x_kij 表示对于车次 C_k 订单 a_ij 是否放入背包中（假如放入的都是符合背包限制的），则多维背包问题[1] 的数学模型为：

$$\textbf{max}\sum_k w_k$$
$$s.t.\begin{cases}\|\sum_i^m\sum_j^m f_{kij}a_{ij}\|_1\le 1\\\|\sum_k f_{kij}a\|_1\le 1\\f_{kij}\in\{0,1\}\\k=1,2,\cdots,m\\i=1,2,\cdots,n\\j=1,2,\cdots,n\end{cases} \tag{15}$$

5.4 模型求解

有了以上模型之后，可以通过贪心算法来进行问题的求解：

利用贪心算法求出每个背包的最大权值，来求出所有背包总体的最大权值。假设背包在取第 i 个物品时获得的权值 P_{iw}，（此时 $1\le i\le m, 0\le P_{iw}\le w_ij$）则对于该背包，考虑第 i 个物品，一定有两种可能，选或是不选：

- 若不选，背包容量不变，改变为问题 $P_{(i-1)w}$。
- 若选，背包容量变小，改变为问题 $P_{(i-1)(w-w_i)}$。

对于第 i 次可以写成 $P_{iw}=max\{P_{(i-1)w},P_{(i-1)(w-w_i)}\}$

整体公式如下：

$$P_{iw}=\begin{cases}0, i=0\\0, w=0\\P_{(i-1)w}, w_i>w\\max\{P_{(i-1)w},P_{(i-1)(w-w_i)}\}\end{cases} \tag{16}$$

计算得出实例 1 至 4，运输车最小停车次数分别为 82，159，162，636，具体方案见附录。

六、 问题二：C=2 的最小停车数模型

6.1 模型准备

对于问题 2，$C_{\text{载}} = 2$，故改变问题 1 中基于整数规划的最小停车次数模型的车载限制约束：

$$f_{kij} \in \{0,1,2\}\ (k = 1,2,\cdots,l \quad i = 1,2,\cdots,n \quad j = 1,2,\cdots,n) \tag{17}$$

$$C_{\text{出发}kx} \in \{0,1,2\}\ (x = 1,2,\cdots,n) \tag{18}$$

6.2 模型建立

基于以上分析，可以写出 $C_{\text{载}} = 2$ 时，基于整数规划的最小停车次数模型：

$$\textbf{min}\ T=\sum_{k=1}^{l}\sum_{i=1}^{n}t_{ki}$$

$$s.t.\begin{cases}\sum\limits_{k=1}^{l}f_{kij}=d_{ij}\\ C_{\text{上}kx}=\sum\limits_{i=1}^{x}\sum\limits_{j=1}^{n}f_{kij}\\ C_{\text{下}kx}=\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{x}f_{kij}\\ C_{\text{出发}kx}=C_{\text{上}kx}-C_{\text{下}kx}\\ t_{\text{上}ki}=\begin{cases}1,\sum\limits_{j=1}^{n}f_{kij}>0.5\\ 0,\sum\limits_{j=i}^{n}f_{kij}=0\end{cases}\\ t_{\text{下}ki}=\begin{cases}1,\sum\limits_{i=1}^{n}f_{kij}>0.5\\ 0,\sum\limits_{i=i}^{n}f_{kij}=0\end{cases}\\ t_{ki}=\begin{cases}1,t_{\text{上}ki}+t_{\text{下}ki}>0\\ 0,t_{\text{上}ki}+t_{\text{下}ki}=0\end{cases}\\ f_{kij}\in\{0,1,2\}\\ C_{\text{出发}kx}\in\{0,1,2\}\\ i=1,2,\cdots,n\\ j=1,2,\cdots,n\\ k=1,2,\cdots,l\\ x=1,2,\cdots,n\end{cases}\tag{19}$$

6.3 算法设计

本文为了使得目标尽可能的优化，故采用组合优化问题的精确算法，是整数规划模型下的优化算法，然后用分支定界法求解。

分支定界法[2] 是一种广义搜索穷举算法，它由“分支”和“定界”两部分组成，其基本思想如下：

(1) 首先不考虑整数约束，求解（IP）的松弛问题（LP），可能得到以下情况：

① 若（LP）没有可行解，则（IP）也没有可行解，迭代停止；

② 若（LP）有最优解，并符合（IP）的整数条件，则（LP）的最优解就是（IP）的最优解，迭代停止；

③ 若（LP）有最优解，但不符合（IP）的整数条件，设（LP）的最优解为 $Z^{(0)}$。

(2) 定界。记（IP）的最优解为 Z^*，以 $Z^{(0)}$ 作为 Z^* 的上界，记为 $\overline{Z} = Z^{(0)}$。再用视察法找（IP）的一个整数可行解 Z' 作为 Z^* 的下界，记为 $\underline{Z} = Z'$（也可以为-∞），则

$$\underline{Z} \leq Z^* \leq \overline{Z} \tag{20}$$

(3) 分支。在（LP）的最优解 $X^{(0)}$ 中，任选一个不符号整数条件的变量 $x = b'$，$[b]$ 为不超过 b' 的最大整数，构造两个约束条件 $x \leq [b]$ 和 $x \geq [b]+1$ 分别加入问题（IP），形成两个子问题（IP1）和（IP2），求解其相关的松弛问题（LP1）和（LP2）。

(4) 修改上下界。规则如下：

① 在各分支问题中，找出目标函数值最大者作为新的上界；

② 从已符合整数条件的分支中，找出目标函数值最大者作为新的下界；

(5) 比较和剪枝。各分枝的目标函数值中，若有小于 $\underline{Z}$ 者，则剪掉此枝，表明此子问题不必继续分支；否则，还要继续分支。如此反复进行，直到 $\underline{Z} = Z^* = \overline{Z}$ 停止，即为整数最优解。

为了使算法更加直观，这里给出对模型的求解图示过程：

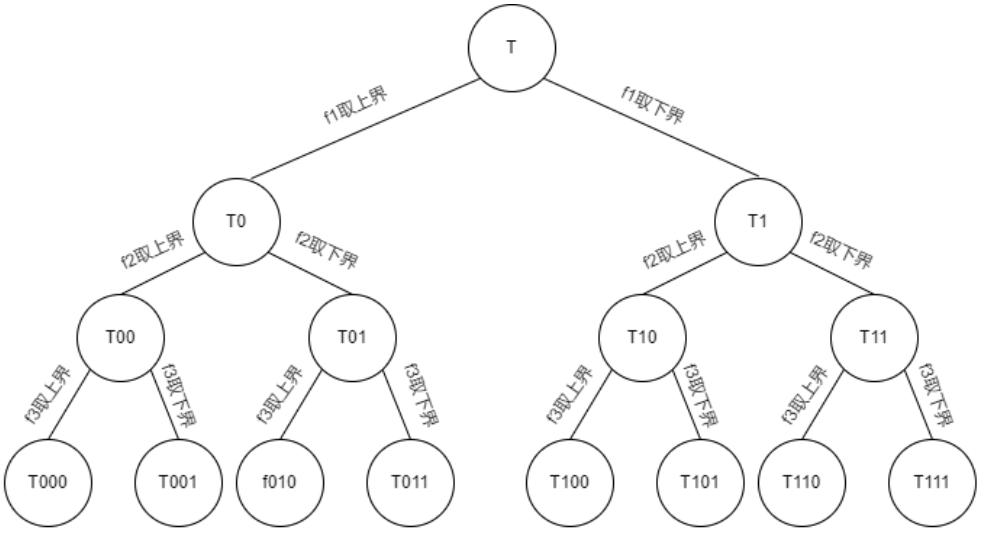

图 1　分支定界法求解示例图

6.4 模型求解

为了加快分支定界法效率，可适当增加前置条件：

当 $d_{ij} = 0$ 时，固定变量 $f_{kij} = 0$ $(k = 1, 2, \cdots, l \quad i = 1, 2, \cdots, n \quad j = 1, 2, \cdots, n)$。

同时通过问题一的动态规划模型求解出完成所有订单的最大车辆数作为1，减少规模进而加快速度，通过分支定界法进行求解，得出四个实例运输车的总停车次数分别为51、122、129、516，具体方案见附录。

七、 问题三：C=2 的最小车辆数模型

7.1 模型建立

7.1.1 目标函数

$$C_{car} = \sum_{k=1}^{l} car(k) \tag{21}$$

$car(k)$ 表示车辆 k 上是否存在订单，公式如下：

$$car(k) = \begin{cases} 1, \sum_{k=1}^{l}\sum_{i=1}^{n}\sum_{j=1}^{n} f_{kij} > 0 \\ 0, \sum_{k=1}^{l}\sum_{i=1}^{n}\sum_{j=1}^{n} f_{kij} = 0 \end{cases} \tag{22}$$

7.1.2 约束条件

$$\sum_{i=1}^{n} Y_{ki} = 0 \tag{23}$$

y_{ki} 表示车辆 k 在 i 站点的行为是否违背车辆从出发站点开始到结束站点间不空载，违背为 1，不违背为 0，公式如下：

$$Y_{ki} = \begin{cases} 1, C_{ki} > 0, C_{k(i-1)} = 0, \sum_{j=1}^{n} f_{kij} > 0 \\ 0, \text{其他} \end{cases} \tag{24}$$

7.1.3 C=2 的最小车辆数模型

基于以上分析，可以写出 $C_{\text{载}} = 2$ 时，基于整数规划的最小车辆数模型：

$$
\begin{aligned}
&\textbf{min}\ C_{car} = \sum_{k=1}^{l} car(k) \\
&s.t. \begin{cases}
car(k) = \begin{cases} 1, \sum_{k=1}^{l}\sum_{i=1}^{n}\sum_{j=1}^{n} f_{kij} > 0 \\ 0, \sum_{k=1}^{l}\sum_{i=1}^{n}\sum_{j=1}^{n} f_{kij} = 0 \end{cases} \\
Y_{ki} = \begin{cases} 1, C_{ki} > 0, C_{k(i-1)} = 0, \sum_{j=1}^{n} f_{kij} > 0 \\ 0, \text{其他} \end{cases} \\
\sum_{i=1}^{n} Y_{ki} = 0 \\
\sum_{k=1}^{l} f_{kij} = d_{ij} \\
C_{\text{上}kx} = \sum_{i=1}^{x}\sum_{j=1}^{n} f_{kij} \\
C_{\text{下}kx} = \sum_{i=1}^{n}\sum_{j=1}^{x} f_{kij} \\
C_{kx} = C_{\text{上}kx} - C_{\text{下}kx} \\
C_{kx} \in \{0,1,2\} \\
f_{kij} \in \{0,1,2\} \\
i = 1,2,\cdots,n \\
j = 1,2,\cdots,n \\
k = 1,2,\cdots,l \\
x = 1,2,\cdots,n
\end{cases}
\end{aligned} \tag{25}
$$

7.2 模型求解

同样使用分支定界法求解，求解得出运输车车辆数分别为 16、28、29、104。

八、 问题四：C=2 的单目标评估模型

8.1 模型准备

本问为了同时考虑极小化所使用运输车辆总个数和极小化所有运输车辆的总停车次数的两个目标，建立双目标评估模型如下：

$$P_{总} = P_{总车辆数} + P_{总停车次数} \tag{26}$$

使用极值差法使得 $P_{总车辆数}$，$P_{总停车次数}$ 无量纲化。

$$P_{总车辆数} = \frac{C_{car}}{m} \tag{27}$$

$$P_{总停车次数} = \frac{T}{2m} \tag{28}$$

8.2 模型建立

评价函数：

$$P_{总} = \frac{C_{car}}{m} + \frac{T}{2m} \tag{29}$$

基于以上分析，可以写出 $C_{载} = 2$ 时，基于整数规划的单目标模型：

$$
\begin{aligned}
&\textbf{min}\ P_{总} = \frac{C_{car}}{m} + \frac{T}{2m} \\
s.t.\ &\begin{cases}
T = \sum_{k=1}^{l}\sum_{i=1}^{n} t_{ki} \qquad C_{car} = \sum_{k=1}^{l} car(k) \\
car(k) = \begin{cases} 1, \sum_{k=1}^{l}\sum_{i=1}^{n}\sum_{j=1}^{n} f_{kij} > 0 \\ 0, \sum_{k=1}^{l}\sum_{i=1}^{n}\sum_{j=1}^{n} f_{kij} = 0 \end{cases} \\
Y_{ki} = \begin{cases} 1, C_{ki} > 0, C_{k(i-1)} = 0, \sum_{j=1}^{n} f_{kij} > 0 \\ 0, 其他 \end{cases} \\
\sum_{i=1}^{n} Y_{ki} = 0 \\
\sum_{k=1}^{l} f_{kij} = d_{ij} \\
t_{上ki} = \begin{cases} 1, \sum_{j=1}^{n} f_{kij} > 0.5 \\ 0, \sum_{j=i}^{n} f_{kij} = 0 \end{cases} \\
t_{下ki} = \begin{cases} 1, \sum_{i=1}^{n} f_{kij} > 0.5 \\ 0, \sum_{i=i}^{n} f_{kij} = 0 \end{cases} \\
t_{ki} = \begin{cases} 1, t_{上ki} + t_{下ki} > 0 \\ 0, t_{上ki} + t_{下ki} = 0 \end{cases} \\
C_{上kx} = \sum_{i=1}^{x}\sum_{j=1}^{n} f_{kij} \\
C_{下kx} = \sum_{i=1}^{n}\sum_{j=1}^{x} f_{kij} \\
C_{kx} = C_{上kx} - C_{下kx} \\
C_{kx} \in \{0,1,2\} \qquad f_{kij} \in \{0,1,2\} \\
i = 1,2,\cdots,n \quad j = 1,2,\cdots,n \\
k = 1,2,\cdots,l \quad x = 1,2,\cdots,n
\end{cases}
\end{aligned}
\tag{30}
$$

8.3 模型求解

同样使用分支定界法求解，求解结果：运输车停车次数分别为 58、120、134、531，运输车车辆数分别为 16、30、35、122。

九、 模型的评价与改进

9.1 模型评价

在当代电商平台日益发展壮大，从网上购物已经变得十分常见并且成为我们日常生活中的重要一部分，而对于货物从商家批发商手里交货至买家手里、P2P 服务中，重要的一环就是物流。物流服务特别是车辆路径调度日益成为社会中一个重要的议题。

本文在理想情况下，建立了整数规划模型求解订单的协同配送问题，算法方面：问题一利用多重背包问题模型和贪心算法得到了局部最优解；问题二至问题四则利用分支定界法进行求解，由于时间复杂度过高，算法并不能高效求解出最优解，因此选择了局部最优解结果。

由于本文情境过于理想化，在运用到实际情况中时，具有一定的局限性和低效性，因此面对实际问题，仍需考虑其他影响因素。

9.2 模型改进

(1) 问题一的 0-1 整数规划模型使用动态规划的贪心算法可以快速求出最优解。
(2) 使用优化后的分支定界法求解可以加快求解出结果。
(3) 分支定界法的时间复杂度为 $O(2^n)$. 随着数据规模上升，求解效率急剧减慢。

参考文献

[1] Jiang.G.F. 动态规划之背包模型 [OL].https://gfjiangly.github.io/algorithm/dp_bag_model.html.2020 年 03 月 03 日

[2] 邓成梁. 运筹学的原理和方法 [M]. 武汉：华中科技大学出版社,2016 年 6 月：233-234

十、 附录

10.1 Python 实现问题一

10.1.1 得到 d_{ij} 矩阵

```
import numpy as np
import pandas as pd


file = "/Users/smh/Desktop/数学建模集训/数模七月短学期集训模型2/集训模型2 数据.xlsx"
maxValues = [20, 40, 60, 160]
for i in range(4):
    sheet = pd.read_excel(file, sheet_name=i, usecols=[0, 1]).to_numpy()
    maxValue = maxValues[i]
    table = np.zeros(shape=(maxValue, maxValue)).astype(dtype=int)
    for row in sheet:
        table[row[0]][row[1]] += 1
    np.savetxt('集训模型2 数据 实例%d.csv'%(i), table, delimiter=',')
    print(table.shape)
```

10.1.2 贪心算法求解

```
# question 1
import pandas as pd
import numpy as np


file = "/Users/smh/Desktop/数学建模集训/数模七月短学期集训模型2/集训模型2 数据.xlsx"
for i in range(5):
    sheet = pd.read_excel(file, sheet_name=i, usecols=[0, 1])
    sheet = sheet.to_numpy()
    allOrder = list()
    while len(sheet):
        order = list()
        reachedIdxs = list()
        startIdx = sheet[:, 0].tolist().index(min(sheet[:, 0]))
        reachedIdxs.append(startIdx)
        order.append(sheet[startIdx, 0])

        sIdx = startIdx
        while(True):
            if sheet[sIdx, 1] in sheet[:, 0]:
                eIdx = (sheet[:, 0].tolist()).index(sheet[sIdx, 1])
                order.append(sheet[eIdx, 1])
                sIdx = eIdx
                reachedIdxs.append(eIdx)
                continue
            else:
```

```
            order.append(sheet[sIdx, 1])
            break

    allOrder.append(order)
    reachedIdxs.sort(reverse=True)
    for idx in reachedIdxs:
        sheet = np.delete(sheet, idx, axis=0)
print(allOrder)
stopSum = 0
for order in allOrder:
    stopSum += len(order) + 1
print(stopSum)
```

10.1.3 问题一具体方案

实例一：

```
[{'weight': 6.8, 'stopTime': 5, 'orderIdxs': {40, 36, 37, 6}, 'orderCount': 4},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {24, 1, 39}, 'orderCount': 3},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {32, 42, 13}, 'orderCount': 3},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {16, 17, 43}, 'orderCount': 3},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {2, 44}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {8, 45}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {49, 11}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {48, 12}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {34, 15}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {19, 47}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {38, 23}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {41, 26}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {29, 46}, 'orderCount': 2},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {0}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {33}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {3}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {35}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {4}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {5}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {7}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {9}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {10}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {14}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {18}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {20}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {21}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {22}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {25}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {27}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {28}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {30}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {31}, 'orderCount': 1}]
total stop time: 82
```

实例二:


```
[{'weight': 9.5, 'stopTime': 10, 'orderIdxs': {1, 99, 37, 40, 46, 14, 81, 22, 93},
    'orderCount': 9},
{'weight': 7.5, 'stopTime': 8, 'orderIdxs': {67, 69, 55, 75, 60, 13, 78}, 'orderCount': 7},
{'weight': 4.5, 'stopTime': 5, 'orderIdxs': {3, 90, 11, 15}, 'orderCount': 4},
{'weight': 4.5, 'stopTime': 5, 'orderIdxs': {56, 8, 27, 92}, 'orderCount': 4},
{'weight': 4.5, 'stopTime': 5, 'orderIdxs': {41, 33, 85, 63}, 'orderCount': 4},
{'weight': 4.5, 'stopTime': 5, 'orderIdxs': {54, 65, 62, 86}, 'orderCount': 4},
{'weight': 3.5, 'stopTime': 4, 'orderIdxs': {2, 76, 38}, 'orderCount': 3},
{'weight': 3.5, 'stopTime': 4, 'orderIdxs': {34, 91, 5}, 'orderCount': 3},
{'weight': 2.5, 'stopTime': 3, 'orderIdxs': {0, 68}, 'orderCount': 2},
{'weight': 2.5, 'stopTime': 3, 'orderIdxs': {64, 4}, 'orderCount': 2},
{'weight': 2.5, 'stopTime': 3, 'orderIdxs': {16, 77}, 'orderCount': 2},
{'weight': 2.5, 'stopTime': 3, 'orderIdxs': {73, 19}, 'orderCount': 2},
{'weight': 2.5, 'stopTime': 3, 'orderIdxs': {59, 23}, 'orderCount': 2},
{'weight': 2.5, 'stopTime': 3, 'orderIdxs': {52, 29}, 'orderCount': 2},
{'weight': 2.5, 'stopTime': 3, 'orderIdxs': {57, 30}, 'orderCount': 2},
{'weight': 2.5, 'stopTime': 3, 'orderIdxs': {96, 31}, 'orderCount': 2},
{'weight': 2.5, 'stopTime': 3, 'orderIdxs': {35, 53}, 'orderCount': 2},
{'weight': 2.5, 'stopTime': 3, 'orderIdxs': {84, 45}, 'orderCount': 2},
{'weight': 2.5, 'stopTime': 3, 'orderIdxs': {98, 61}, 'orderCount': 2},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {6}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {7}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {9}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {10}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {12}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {17}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {18}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {20}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {21}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {24}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {25}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {26}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {28}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {32}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {36}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {39}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {42}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {43}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {44}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {47}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {48}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {49}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {97}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {66}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {70}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {71}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {72}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {74}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {79}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {80}, 'orderCount': 1},
```

```
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {50}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {82}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {83}, 'orderCount': 1},
 {'weight': 1.5, 'stopTime': 2, 'orderIdxs': {51}, 'orderCount': 1},
 {'weight': 1.5, 'stopTime': 2, 'orderIdxs': {87}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {88}, 'orderCount': 1},
 {'weight': 1.5, 'stopTime': 2, 'orderIdxs': {89}, 'orderCount': 1},
 {'weight': 1.5, 'stopTime': 2, 'orderIdxs': {58}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {94}, 'orderCount': 1},
{'weight': 1.5, 'stopTime': 2, 'orderIdxs': {95}, 'orderCount': 1}]
total stop time: 159
```

实例三:

```
[{'weight': 11.0, 'stopTime': 8, 'orderIdxs': {48, 18, 85, 90, 15, 29, 79}, 'orderCount': 7},
{'weight': 6.8,'stopTime': 5, 'orderIdxs': {80, 89, 54, 1}, 'orderCount': 4},
{'weight': 6.8, 'stopTime': 5, 'orderIdxs': {24, 65, 76, 8},'orderCount': 4},
{'weight': 6.8, 'stopTime': 5, 'orderIdxs': {99, 36, 37, 23}, 'orderCount': 4},
{'weight': 6.8, 'stopTime':5, 'orderIdxs': {32, 33, 28, 93}, 'orderCount': 4},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {97, 74, 3}, 'orderCount': 3},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {26, 12, 87}, 'orderCount': 3},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {19,83, 51}, 'orderCount': 3},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {49, 22, 31}, 'orderCount': 3},
{'weight':3.9999999999999996, 'stopTime': 3, 'orderIdxs': {0, 50}, 'orderCount': 2},
{'weight': 3.9999999999999996,'stopTime': 3, 'orderIdxs': {5, 62}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {45,7}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {10, 52}, 'orderCount': 2},
{'weight':3.9999999999999996, 'stopTime': 3, 'orderIdxs': {64, 13}, 'orderCount': 2},
{'weight': 3.9999999999999996,'stopTime': 3, 'orderIdxs': {16, 38}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {72,21}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {39, 55}, 'orderCount': 2},
{'weight':3.9999999999999996, 'stopTime': 3, 'orderIdxs': {41, 98}, 'orderCount': 2},
{'weight': 3.9999999999999996,'stopTime': 3, 'orderIdxs': {44, 63}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {56,81}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {70, 95}, 'orderCount': 2},
{'weight':2.6, 'stopTime': 2, 'orderIdxs': {2}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {4}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {6}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {9},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {11}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {14}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {17}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {20}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {25}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {27}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {30},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {34}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {35}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {40}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {42}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {43}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {46}, 'orderCount': 1},
```

```
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {47},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {53}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {57}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {58}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {59}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {60}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {96}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {66},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {67}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {68}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {69}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {71}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {73}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {75}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {77},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {78}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {82}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {84}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {86}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {88}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {91}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {92},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {61}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {94}, 'orderCount': 1}]
total stop time: 162
```

实例四:

```
[{'weight': 15.2, 'stopTime': 11, 'orderIdxs': {384, 35, 292, 262, 304, 337, 18, 213, 254,
    286}, 'orderCount':10},
{'weight': 12.399999999999999, 'stopTime': 9, 'orderIdxs': {98, 146, 259, 99, 7, 297, 283,
    61}, 'orderCount': 8},
{'weight': 12.399999999999999, 'stopTime': 9, 'orderIdxs': {65, 151, 27, 202, 315, 316, 78,
    95}, 'orderCount': 8},
{'weight': 11.0, 'stopTime': 8, 'orderIdxs': {280, 34, 227, 183, 88, 138, 120}, 'orderCount':
    7},
{'weight': 11.0,'stopTime': 8, 'orderIdxs': {64, 354, 37, 200, 121, 90, 270}, 'orderCount': 7},
{'weight': 11.0, 'stopTime': 8, 'orderIdxs':{129, 387, 343, 57, 203, 172, 77}, 'orderCount':
    7},
{'weight': 8.2, 'stopTime': 6, 'orderIdxs': {1, 324, 91, 28, 367},'orderCount': 5},
{'weight': 8.2, 'stopTime': 6, 'orderIdxs': {321, 178, 13, 44, 333}, 'orderCount': 5},
{'weight': 8.2,'stopTime': 6, 'orderIdxs': {352, 273, 42, 55, 218}, 'orderCount': 5},
{'weight': 8.2, 'stopTime': 6, 'orderIdxs': {320,201, 186, 43, 156}, 'orderCount': 5},
{'weight': 8.2, 'stopTime': 6, 'orderIdxs': {310, 70, 379, 140, 109}, 'orderCount':5},
{'weight': 8.2, 'stopTime': 6, 'orderIdxs': {208, 261, 245, 251, 205}, 'orderCount': 5},
{'weight': 6.8, 'stopTime': 5,'orderIdxs': {161, 10, 19, 180}, 'orderCount': 4},
{'weight': 6.8, 'stopTime': 5, 'orderIdxs': {83, 30, 334, 207},'orderCount': 4},
{'weight': 6.8, 'stopTime': 5, 'orderIdxs': {36, 349, 150, 71}, 'orderCount': 4},
{'weight': 6.8,'stopTime': 5, 'orderIdxs': {274, 378, 243, 46}, 'orderCount': 4},
{'weight': 6.8, 'stopTime': 5, 'orderIdxs': {394, 190,74, 134}, 'orderCount': 4},
{'weight': 6.8, 'stopTime': 5, 'orderIdxs': {371, 281, 131, 112}, 'orderCount': 4},
{'weight': 6.8,'stopTime': 5, 'orderIdxs': {344, 369, 356, 157}, 'orderCount': 4},
```

```
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {336, 11,107}, 'orderCount': 3},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {376, 17, 330}, 'orderCount': 3},
{'weight': 5.4,'stopTime': 4, 'orderIdxs': {242, 342, 23}, 'orderCount': 3},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {348, 117, 38},'orderCount': 3},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {103, 39, 111}, 'orderCount': 3},
{'weight': 5.4, 'stopTime': 4,'orderIdxs': {40, 93, 382}, 'orderCount': 3},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {49, 133, 383}, 'orderCount': 3},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {68, 53, 295}, 'orderCount': 3},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {67,189, 159}, 'orderCount': 3},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {73, 225, 250}, 'orderCount': 3},
{'weight': 5.4,'stopTime': 4, 'orderIdxs': {322, 75, 399}, 'orderCount': 3},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {80, 361, 198},'orderCount': 3},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {393, 341, 118}, 'orderCount': 3},
{'weight': 5.4, 'stopTime':4, 'orderIdxs': {296, 169, 122}, 'orderCount': 3},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {288, 125, 149},'orderCount': 3},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {257, 165, 353}, 'orderCount': 3},
{'weight': 5.4, 'stopTime':4, 'orderIdxs': {241, 228, 173}, 'orderCount': 3},
{'weight': 5.4, 'stopTime': 4, 'orderIdxs': {240, 199, 216}, 'orderCount':3},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {8, 182}, 'orderCount': 2},
{'weight':3.9999999999999996, 'stopTime': 3, 'orderIdxs': {234, 12}, 'orderCount': 2},
{'weight': 3.9999999999999996,'stopTime': 3, 'orderIdxs': {285, 15}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs':{276, 21}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {368, 24}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {188, 29}, 'orderCount': 2},
{'weight':3.9999999999999996, 'stopTime': 3, 'orderIdxs': {194, 31}, 'orderCount': 2},
{'weight': 3.9999999999999996,'stopTime': 3, 'orderIdxs': {211, 45}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs':{184, 50}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {96, 52}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {59, 302}, 'orderCount': 2},
{'weight':3.9999999999999996, 'stopTime': 3, 'orderIdxs': {66, 355}, 'orderCount': 2},
{'weight': 3.9999999999999996,'stopTime': 3, 'orderIdxs': {236, 69}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs':{72, 267}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {76, 366}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {128, 79}, 'orderCount': 2},
{'weight':3.9999999999999996, 'stopTime': 3, 'orderIdxs': {84, 374}, 'orderCount': 2},
{'weight': 3.9999999999999996,'stopTime': 3, 'orderIdxs': {89, 187}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {94,143}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {265, 105}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {110, 398}, 'orderCount': 2},
{'weight':3.9999999999999996, 'stopTime': 3, 'orderIdxs': {113, 221}, 'orderCount': 2},
{'weight': 3.9999999999999996,'stopTime': 3, 'orderIdxs': {114, 326}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs':{298, 115}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {392, 124}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {289, 127}, 'orderCount': 2},
{'weight':3.9999999999999996, 'stopTime': 3, 'orderIdxs': {308, 135}, 'orderCount': 2},
{'weight': 3.9999999999999996,'stopTime': 3, 'orderIdxs': {137, 220}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs':{141, 247}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {153, 290}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {154, 263}, 'orderCount': 2},
{'weight':3.9999999999999996, 'stopTime': 3, 'orderIdxs': {155, 396}, 'orderCount': 2},
{'weight': 3.9999999999999996,'stopTime': 3, 'orderIdxs': {377, 162}, 'orderCount': 2},
```

```
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs':{177, 340}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {192, 185}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {191, 319}, 'orderCount': 2},
{'weight':3.9999999999999996, 'stopTime': 3, 'orderIdxs': {193, 293}, 'orderCount': 2},
{'weight': 3.9999999999999996,'stopTime': 3, 'orderIdxs': {282, 212}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs':{299, 239}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {360, 255}, 'orderCount': 2},
{'weight': 3.9999999999999996, 'stopTime': 3, 'orderIdxs': {291, 357}, 'orderCount': 2},
{'weight':3.9999999999999996, 'stopTime': 3, 'orderIdxs': {338, 364}, 'orderCount': 2},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {0}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {2}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {3}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {4}, 'orderCount': 1},
{'weight':2.6, 'stopTime': 2, 'orderIdxs': {5}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {6}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {9}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {14},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {16}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {20}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {22}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {25}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {26}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {32}, 'orderCount': 1},
 {'weight': 2.6, 'stopTime': 2, 'orderIdxs': {33},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {41}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {47}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {48}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {51}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {54}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {56}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {58},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {60}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {62}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {63}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {81}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {82}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {85}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {86},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {87}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {92}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {97}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {100}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {101}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {102}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {104},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {106}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {108}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {116}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {119}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {123}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {126}, 'orderCount': 1},
```

```
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {130},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {132}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {136}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {139}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {142}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {144}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {145}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {147},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {148}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,
'orderIdxs': {152}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {158}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {160}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {163}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {164}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {166},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {167}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {168}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {170}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {171}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {174}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {175}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {176},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {179}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {181}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {195}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {196}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {197}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {204}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {206},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {209}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {210}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {214}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {215}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {217}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {219}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {222},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {223}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,
'orderIdxs': {224}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {226}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {256}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {385}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {258}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {386},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {260}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {388}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {389}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {390}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {391}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {264}, 'orderCount': 1},
```

```
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {266},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {395}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {268}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {269}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {397}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {271}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {272}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {275},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {277}, 'orderCount': 1},
 {'weight': 2.6, 'stopTime': 2,'orderIdxs': {278}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {279}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {284}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {287}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {294}, 'orderCount': 1},
 {'weight': 2.6, 'stopTime': 2, 'orderIdxs': {300},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {301}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {303}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {305}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {306}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {307}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {309}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {311},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {312}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {313}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {314}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {317}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {318}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {323}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {325},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {327}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {328}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {329}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {331}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {332}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {335}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {339},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {345}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {380}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {346}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {347}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {350}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {351}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {375},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {229}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {230}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {358}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {231}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {253}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {359}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {232},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {233}, 'orderCount': 1},
```

```
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {362}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {235}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {363}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {237}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {365}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {238},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {370}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {244}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {372}, 'orderCount': 1},
{'weight': 2.6,'stopTime': 2, 'orderIdxs': {373}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {246}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {248}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {249},'orderCount': 1},
{'weight': 2.6, 'stopTime': 2, 'orderIdxs': {252}, 'orderCount': 1},
{'weight': 2.6, 'stopTime': 2,'orderIdxs': {381}, 'orderCount': 1}]
total stop time: 636
```

10.2 Python 实现问题二

```python
import pandas as pd
import numpy as np


sheet: np.ndarray
allSelectedIdxs: set = set()
allIdxOccurence: dict = dict()
allConnectedIdxs: dict = dict()
allSortedIdxs: list = list()
maxStopTime: int = 0
C: int = 0


def isConnected(idxToSelect: int, selectedIdxs: set) -> bool:
    global sheet
    s0, e0 = sheet[idxToSelect, :]
    for idx in selectedIdxs:
        s, e = sheet[idx, :]
        flag = (s==s0) + (s==e0) + (e==s0) + (e==e0)
        if flag:
            return True
    return False


def calculateWeight(info: dict) -> float:
    orderCountW = 0.2
    stopTimeW = 1.2
    weight = orderCountW * info['orderCount'] + stopTimeW * info['stopTime']
    return weight
```

```
def rangeIntersection(idx1: int, idx2: int) -> tuple:
    global sheet
    (s1, e1) = sheet[idx1, :]
    (s2, e2) = sheet[idx2, :]
    if e1 - s1 < e2 - s2:
        (s1, e1, s2, e2) = (s2, e2, s1, e1)

    if s1 <= s2 <= e2 <= e1:
        return s2, e2
    elif s2 <= s1 <= e2 <= e1:
        return s1, e2
    elif s1 <= s2 <= e1 <= e2:
        return s2, e1
    else:
        return 0, 0


def isTruckToBeFull(idxToSelect: int, selectedIdxs: set) -> bool:
    global C
    # start, end = sheet[idxToSelect, :]
    count: int = 0
    for idx in selectedIdxs:
        r = rangeIntersection(idxToSelect, idx)
        if r[1] - r[0]:
            count += 1
    return count >= C


def calculateDeltaStopTime(idxToSelect: int, selectedIdxs: set) -> int:
    global sheet
    start, end = sheet[idxToSelect, :]
    flagS: bool = False
    flagE: bool = False
    for idx in selectedIdxs:
        if start == sheet[idx, 0] or start == sheet[idx, 1]:
            flagS = True
        if end == sheet[idx, 0] or end == sheet[idx, 1]:
            flagE = True
        if flagS and flagE:
            break
    return 2 - flagE - flagS


def calculateDeltaWeight(idxToSelect: int, infoOld: dict):
    oldWeight = infoOld['weight']
    infoOld['stopTime'] += calculateDeltaStopTime(idxToSelect, infoOld['orderIdxs'])
    infoOld['orderCount'] += 1
    newWeight = calculateWeight(infoOld)
    return newWeight - oldWeight
```

```
def fullScan():
    global sheet, allConnectedIdxs, allIdxOccurence
    for idx in range(len(sheet)):
        allConnectedIdxs[idx]: set = set()
        for idx2 in range(len(sheet)):
            if idx2 == idx:
                continue
            if isConnected(idx2, {idx}):
                allConnectedIdxs[idx].add(idx2)

                if idx2 in allIdxOccurence:
                    allIdxOccurence[idx2] += 1
                else:
                    allIdxOccurence[idx2] = 1


def selectBestRoute() -> set:
    global allSelectedIdxs, allIdxOccurence
    maxOrderCount = 0
    maxOrders = set()
    remainingIdxs = set(range(len(sheet))).difference(allSelectedIdxs)
    # 根据出现次数的多少对remainingIdxs进行倒序排序
    for idx in dict(sorted({key: allIdxOccurence[key] for key in remainingIdxs}.items(),
        key=lambda item: item[1])).keys():
        orderPoolSet: set = {idx}
        orderPoolList: list = list(orderPoolSet)
        orderPoolSize = 0
        pointer = 0
        while not (orderPoolSize == len(orderPoolSet)):
            orderPoolSize = len(orderPoolSet)
            for i in range(pointer, orderPoolSize):
                idx2 = orderPoolList[i]
                if idx2 in allSelectedIdxs:
                    continue
                # for idx3 in allConnectedIdxs[idx2]:
                for idx3 in dict(sorted({key: allIdxOccurence[key] for key in
                    allConnectedIdxs[idx2]}.items(), key=lambda item: item[1])).keys():
                    if idx3 in orderPoolSet or idx3 in allSelectedIdxs:
                        continue

                    if not isTruckToBeFull(idx3, orderPoolSet):
                        orderPoolSet.add(idx3)
                        orderPoolList.append(idx3)
            pointer = orderPoolSize
        orderCount = len(orderPoolSet)
        if orderCount > maxOrderCount:
            # print("greater,", orderCount, orderPoolSet)
            maxOrderCount = orderCount
            maxOrders = orderPoolSet
    return maxOrders
```

```
file = "/Users/smh/Desktop/数学建模集训/数模七月短学期集训模型2/集训模型2 数据.xlsx"
C = 2
i = 1
sheet = pd.read_excel(file, sheet_name=i, usecols=[0, 1]).to_numpy()
fullScan()
print("all connected idxs:", allConnectedIdxs)
print("all occurance:", allIdxOccurence)

allOrders: list = []
while len(allSelectedIdxs) < len(sheet):
    order = selectBestRoute()
    print(order)
    allOrders.append(order)
    allSelectedIdxs = allSelectedIdxs.union(order)

print("all orders:", allOrders)
print("total stop time:", sum([len(iter)+1 for iter in allOrders]))
print("total car count:", len(allOrders))
```

10.2.1 问题二具体方案

```
实例一:
[{1, 37, 6, 40, 11, 49},
{32, 38, 43, 13, 16, 17},
{0, 36, 39, 46, 24},
{8, 34, 35, 45},
{48, 33, 12, 47},
{26, 27, 44, 7},
{42, 14, 23},
{2, 3},
{25, 4},
{9, 5},
{41, 15},
{10, 28},
{18, 21},
{19, 29},
{20, 30},
{22, 31}]
total stop time: 66

实例二:
 [{65, 1, 67, 99, 69, 37, 75, 11, 13, 46, 55, 60, 63},
{33, 34, 2, 5, 8, 40, 76, 45, 78, 56, 93},
{0, 68, 81, 54, 86, 89, 30}, {35, 4, 41, 15, 16, 53},
{96, 98, 14, 52, 90}, {73, 66, 19, 22},
{36, 38, 7, 23, 59},
{80, 27, 84, 92},
{57, 82, 91, 85},
```

```
{72, 3, 61},
{47, 29, 79},
{51, 43, 31},
{64, 6},
{9, 42},
{25, 10},
{12, 28},
{17, 44},
{18, 74},
{83, 20},
{32, 71},
{97, 26},
{77, 39},
{48, 49},
{50, 94},
{24, 88},
{58, 95},
{70},
{21},
{62},
{87}]
total stop time: 130
```

实例三:

```
[{32, 1, 33, 68, 12, 15, 18, 22, 55, 87, 89},
{97, 3, 74, 79, 48, 83, 85, 90, 29},
{65, 7, 8, 72, 10, 76, 24},
{99, 36, 37, 80, 54, 23, 26},
{0, 38, 44, 50, 30},
{64, 5, 60, 13},
{41, 98, 28, 93},
{2, 51, 66},
{56, 82, 35},
{49, 31, 39},
{45, 52, 4},
{73, 69, 63},
{84, 70, 95},
{75, 6},
{9, 71},
{11, 20},
{77, 14},
{16, 46},
{17, 47},
{88, 21},
{25, 86},
{34, 27},
{67, 94},
{40, 78},
{59, 43},
{58, 53},
{96},
```

```
{42},
{81},
{19},
{57},
{91},
{92},
{61},
{62}]
total stop time: 135
```

实例四:

```
[{98, 259, 99, 354, 38, 7, 67, 265, 200, 297, 37, 334, 207, 146, 26},
{34, 227, 357, 199, 360, 138, 109, 120, 111, 245, 183, 280, 28, 216},
{384, 292, 133, 262, 135, 304, 337, 49, 308, 213, 244, 254, 286},
{224, 64, 161, 72, 105, 301, 180, 157, 90, 283, 61},
{352, 1, 129, 387, 194, 70, 203, 268, 367, 57, 91},
{320, 68, 196, 201, 363, 43, 140, 53, 186, 156},
{65, 264, 393, 296, 202, 78, 151, 313, 315, 95},
{257, 353, 165, 326, 42, 172, 77, 273, 55},
{193, 321, 293, 44, 13, 333, 178, 88},
{322, 261, 11, 107, 336, 208, 243, 251},
{35, 36, 71, 18, 150, 154, 383},
{289, 386, 332, 46, 241, 19, 316},
{131, 134, 232, 394, 74, 112, 190},
{40, 10, 267, 302, 93, 382},
{356, 298, 15, 369, 344, 285},
{355, 170, 399, 83, 277, 63},
{96, 101, 205, 274, 182, 378},
{225, 73, 343, 89, 250, 187},
{128, 137, 276, 21, 220, 191},
{5, 168, 299, 239, 117},
{359, 362, 149, 184, 188},
{69, 104, 236, 177, 340},
{331, 371, 84, 281, 255},
{0, 211, 45, 230},
{24, 368, 253, 310},
{361, 162, 27, 212},
{323, 103, 76, 39},
{58, 3, 164, 270},
{338, 398, 364, 110},
{153, 114, 179, 290},
{376, 17, 139, 269},
{240, 288, 379, 287},
{242, 23, 342, 319},
{9, 132, 228},
{234, 12, 263},
{32, 396, 20},
{192, 33, 185},
{50, 124, 66},
{256, 52, 160},
{171, 237, 62},
```

```
{80, 198, 390},
{82, 260, 189},
{56, 325, 86},
{210, 94, 143},
{126, 108, 206},
{314, 116, 324},
{358, 341, 118},
{97, 122, 169},
{247, 141, 231},
{377, 155, 4},
{392, 163, 349},
{113, 106, 221},
{248, 370, 317},
{41, 2},
{197, 6},
{8, 328},
{14, 167},
{16, 372},
{226, 22},
{25, 266},
{29, 279},
{300, 30},
{291, 31},
{130, 47},
{48, 388},
{51, 375},
{54, 271},
{385, 59},
{136, 60},
{81, 79},
{152, 85},
{102, 87},
{258, 294},
{389, 391},
{395, 195},
{209, 397},
{214, 142},
{144, 345},
 {272, 303},
{145, 166},
{147, 318},
{275, 350},
{339, 148},
{284, 278},
{330, 158},
{365, 159},
{115, 295},
{235, 174},
{305, 249},
{233, 306},
{307, 238},
```

```
{347, 181},
{366, 311},
{229, 327},
{125, 335},
{346, 351},
{348, 246},
{123, 374},
{282},
{173},
{175},
{176},
{309},
{312},
{223},
{100},
{217},
{218},
{329},
{75},
{204},
{119},
{252},
{373},
{215},
{380},
{121},
{219},
{92},
{381},
{222},
{127}]
total stop time: 522
```

10.3 Python 实现问题三

```python
import pandas as pd
import numpy as np


sheet: np.ndarray
allSelectedIdxs: set = set()
allIdxOccurence: dict = dict()
allConnectedIdxs: dict = dict()
allSortedIdxs: list = list()
maxStopTime: int = 0
C: int = 0


def isConnected(idxToSelect: int, selectedIdxs: set) -> bool:
    global sheet
```

```
    s0, e0 = sheet[idxToSelect, :]
    for idx in selectedIdxs:
        s, e = sheet[idx, :]
        flag = (s==s0) + (s==e0) + (e==s0) + (e==e0)
        if flag:
            return True
    return False


def calculateWeight(info: dict) -> float:
    orderCountW = 0.2
    stopTimeW = 1.2
    weight = orderCountW * info['orderCount'] + stopTimeW * info['stopTime']
    return weight


def rangeIntersection(idx1: int, idx2: int) -> tuple:
    global sheet
    (s1, e1) = sheet[idx1, :]
    (s2, e2) = sheet[idx2, :]
    if e1 - s1 < e2 - s2:
        (s1, e1, s2, e2) = (s2, e2, s1, e1)

    if s1 <= s2 <= e2 <= e1:
        return s2, e2
    elif s2 <= s1 <= e2 <= e1:
        return s1, e2
    elif s1 <= s2 <= e1 <= e2:
        return s2, e1
    else:
        return 0, 0


def isTruckToBeFull(idxToSelect: int, selectedIdxs: set) -> bool:
    global C
    # start, end = sheet[idxToSelect, :]
    count: int = 0
    for idx in selectedIdxs:
        r = rangeIntersection(idxToSelect, idx)
        if r[1] - r[0]:
            count += 1
    return count >= C


def calculateDeltaStopTime(idxToSelect: int, selectedIdxs: set) -> int:
    global sheet
    start, end = sheet[idxToSelect, :]
    flagS: bool = False
    flagE: bool = False
    for idx in selectedIdxs:
        if start == sheet[idx, 0] or start == sheet[idx, 1]:
```

```
            flagS = True
        if end == sheet[idx, 0] or end == sheet[idx, 1]:
            flagE = True
        if flagS and flagE:
            break
    return 2 - flagE - flagS


def calculateDeltaWeight(idxToSelect: int, infoOld: dict):
    oldWeight = infoOld['weight']
    infoOld['stopTime'] += calculateDeltaStopTime(idxToSelect, infoOld['orderIdxs'])
    infoOld['orderCount'] += 1
    newWeight = calculateWeight(infoOld)
    return newWeight - oldWeight


def fullScan():
    global sheet, allConnectedIdxs, allIdxOccurence
    for idx in range(len(sheet)):
        allConnectedIdxs[idx]: set = set()
        for idx2 in range(len(sheet)):
            if idx2 == idx:
                continue
            if isConnected(idx2, {idx}):
                allConnectedIdxs[idx].add(idx2)

                if idx2 in allIdxOccurence:
                    allIdxOccurence[idx2] += 1
                else:
                    allIdxOccurence[idx2] = 1


def selectBestRoute(initIdx: int) -> set:
    global allSelectedIdxs, allIdxOccurence
    maxOrderCount = 0
    maxOrders = set()
    remainingIdxs = set(range(len(sheet))).difference(allSelectedIdxs.union({initIdx}))
    # 根据出现次数的多少对remainingIdxs进行倒序排序
    for idx in dict({key: allIdxOccurence[key] for key in remainingIdxs}.items(), key=lambda
        item: item[1]).keys():
        orderPoolSet: set = {idx, initIdx}
        orderPoolList: list = list(orderPoolSet)
        orderPoolSize = 0
        pointer = 0
        while not (orderPoolSize == len(orderPoolSet)):
            orderPoolSize = len(orderPoolSet)
            for i in range(pointer, orderPoolSize):
                idx2 = orderPoolList[i]
                if idx2 in allSelectedIdxs:
                    continue
                # for idx3 in allConnectedIdxs[idx2]:
```

```
            for idx3 in dict(sorted({key1: allIdxOccurence[key1] for key1 in
                allConnectedIdxs[idx2]}.items(), key=lambda item: item[1])).keys():
                if idx3 in orderPoolSet or idx3 in allSelectedIdxs:
                    continue

                if not isTruckToBeFull(idx3, orderPoolSet):
                    orderPoolSet.add(idx3)
                    orderPoolList.append(idx3)
        pointer = orderPoolSize
    orderCount = len(orderPoolSet)
    if orderCount > maxOrderCount:
        # print("greater,", orderCount, orderPoolSet)
        maxOrderCount = orderCount
        maxOrders = orderPoolSet
return maxOrders



file = "/Users/smh/Desktop/数学建模集训/数模七月短学期集训模型2/集训模型2 数据.xlsx"
C = 2
i = 1
sheet = pd.read_excel(file, sheet_name=i, usecols=[0, 1]).to_numpy()
fullScan()
print("all connected idxs:", allConnectedIdxs)
print("all occurance:", allIdxOccurence)

initCarCount = 18
minWeight = 0
while initCarCount > 2:
    initCarCount -= 1
    carLine: list = []
    sortedIdxs: list = list(dict(allIdxOccurence.items(), key=lambda item: item[1]).keys())
    for i in range(initCarCount):
        initIdx = sortedIdxs[i]
        res = selectBestRoute(initIdx)
        carLine.append(res)
        allSelectedIdxs = allSelectedIdxs.union(res)
    if len(sheet) > len(allSelectedIdxs):
        print(set(range(len(sheet))).difference(allSelectedIdxs))
        exit()
```

10.4 lingo 实现分支定界法

```
model:
sets:
  m/1../:car;
  n/1..60/:;
  c/1..59/:;
  a(n,n):d;
  b(m,n):t,t1,t2,s;
```

```
  z(m,n,n):f;
endsets

data:
  d=@ole('D:/桌面/集训模型2/d3.xlsx','datad');
enddata


min=@sum(b(k,i):t(k,i));

@for(n(i):@for(n(j):@sum(m(k):f(k,i,j))=d(i,j)));


@for(m(k):s(k,1)=@sum(n(j):f(k,1,j))-@sum(n(i):f(k,i,1)));
@for(m(k):@for(c(x):s(k,x+1)=s(k,x)+@sum(n(j):f(k,x+1,j))-@sum(n(i):f(k,i,x+1))));

@for(b(k,i):s(k,i)<=2);
!@for(z(k,i,j):f(k,i,j)<=2);
@for(b(k,i):s(k,i)>=0);
!@for(z(k,i,j):f(k,i,j)>=0);

@for(b(k,i):t1(k,i)=@if(@sum(n(j):f(k,i,j))#gt#0.5,1,0));
@for(b(k,j):t2(k,j)=@if(@sum(n(i):f(k,i,j))#gt#0.5,1,0));
@for(b(k,i):t(k,i)=@if((t1(k,i)+t2(k,i))#gt#0.5,1,0));
@for(m(k):car(k)=@if(@sum(a(i,j):f(k,i,j))#gt#0,1,0));

@for(z(k,i,j):f(k,i,1)=0);
@for(z(k,i,j):f(k,i,j)=@if(d(i,j)#eq#0,0,f(k,i,j)));


@for(z(k,i,j):@gin(f(k,i,j)));
@for(z(k,i,j):@bnd(0,f(k,i,j),2));
!@for(b(k,i):@gin(s(k,i)));
!@for(b(k,i):@bnd(0,s(k,i),2));

!@for(b(k,i):@bin(t(k,i)));
!@for(b(k,i):@bin(t1(k,i)));
!@for(b(k,i):@bin(t2(k,i)));




end
```