|  |  |
| --- | --- |
| 队伍编号 | MC25006414 |
| 题号 | D |

基于XGBoost和启发式算法的货量预测与车辆调度模型

**摘 要**

随着国内外消费市场对物流运输需求的增加，短途运输作为物流末端环节，在物流运输中至关重要。对短途运输活动进行合理化规划，精准预测短途运输的货量，并进行合理的车辆调度，能提升运输效率，降低成本，增强物流企业的市场竞争力。本文旨在通过时间序列预测模型，实现对未来一天各线路货量的有效预测，同时使用启发式算法对所预测的各线路货量的车辆调度进行科学合理的安排。

**针对问题一**，为了预测未来一天内各个线路的日、分钟货量，首先整理11月30日至12月15日的历史数据，各个时间段货量以同一线路为分组，整合日期和具体时间点为后续操作做准备。考虑到对货量缺失值的补零操作对模型预测效果的不良影响，选择不对无货量时段补零。模型选择**XGBoost**提升树模型，采用训练集为12月1日-14日数据（90%），测试集为12月14日-15日数据（10%）。利用**XGBoost**模型对货量进行初步预测，可以得到“场地3- 站点83–0600”线路历史数据上展现出21：00货量最少，5：50货量达到峰值的特点；而“场地3- 站点83–1400”也具有明显货量变化趋势。通过构建**XGBoost结合STL时间序列季节性分解方法**和**ARIMA自回归预测方法**等，得到未来一天内各个线路的日、分钟粒度货量，详见结果表1、结果表2。

**针对问题二**，为了基于问题1的预测结果，规划发车量、发运时间、串点及车辆调度，最小化总成本获得最佳车辆调度方案，在确定了：1.车辆调度需满足可用性(自有车周转率优先)和容量限制;2.合并需求时，我们设置等待时间不超过1个时间片(10分钟)等约束条件、合并（“串点”）策略和车辆优先级后，制定了取**最小成本的目标函数**，并采用**启发式算法**来获得各线路需求的车辆调度方案，详见结果表3。

**针对问题三**，由于引入无限量标准容器，装卸时间缩短至10分钟，但车辆装载量降至800包裹，我们对于问题2的策略进行了调整：1.需求合并优先级:标准容器需求 >串点需求 >普通需求；2.车辆调度优先级:可用车辆>新车辆>标准容器车>外车，并沿用了前面的串点策略和时间片考虑，同样采用**启发式算法**得到了使用标准容器下的车辆调度优化方案，详见结果表4.

**针对问题四**，我们设置偏差范围为[-30%,30%]，并将评估指标设置为外部车辆数、自有车周转次数、车辆均包裹量、总成本。我们将预测货量调整为基准值的70%~130%的7种情景和预测货量结果表输入到问题3模型中。采用**多指标**对比图来量化偏差影响，详情见图9。

**关键词**： XGBoost 时间序列季节性分解 自回归预测 启发式算法

目录

一、问题重述 1

1.1 背景知识 3

1.2 问题描述 3

二、问题分析 4

2.1对问题一的分析 4

2.2对问题二的分析 4

2.3对问题三的分析 4

2.4对问题四的分析 4

三、模型假设 5

四、符号说明 5

五、模型的建立及求解 6

5.1 问题一： 6

5.1.1基于XGBoost的分钟预测模型 6

5.1.2 分钟预测模型的求解 6

5.2 问题二： 10

5.2.1基于规则与启发式算法的调度优化 10

5.2.2车辆调度优化的求解 13

5.2.3 结果数据可视化 14

5.3 问题三： 15

5.3.1基于规则与启发式算法的调度优化 15

5.3.2使用标准容器的车辆调度优化的求解 16

5.3.3 结果数据可视化 19

5.4 问题四： 20

5.4.1不同偏差情况对于优化调度的影响体现 20

六、模型的评价、改进与推广 22

6.1 模型的优点 22

6.2 模型的缺点 22

6.3 模型的改进 22

6.1 模型的推广 22

七、参考文献 23

八、附录 24

**一、问题重述**

**1.1 背景知识**

习近平总书记精准指出，“物流是实体经济的‘筋络’，连接生产与消费、内贸与外贸”，明确了物流在经济体系里的重要位置。随着消费者购买能力提升以及物流业的发展，消费者对国内外市场商品的需求持续高涨，这使得物流运输的重要性愈发突出。短途运输作为物流配送的核心部分，尽管行程较短，却有着高频率的特点，其直接关系到货物能否迅速、准确地交付到消费者手中，在整个物流体系中起着不可或缺的基础性作用。

面对市场对物流需求的不断增长，提升物流效率已成为当务之急。借助科学合理的车辆调度，结合大数据分析对短途运输业务量进行精准预测，不仅能够优化运输资源的配置，缩短货物运输时间，极大地提升消费者的购物体验与幸福感，还能助力物流企业在激烈的市场竞争中提升自身竞争力，进而拓展市场份额。高效的物流体系对于促进国内国际消费市场的良性循环具有积极意义，它有效地串联起生产、流通和消费各个环节，为构建国内大循环和国内国际双循环相互促进的新发展格局提供了有力支撑。

**1.2问题描述**

**问题1**：建立货量预测模型，预测各个线路未来一天的货量，并以10分钟的颗粒度来拆分每条线路的总货量，其中结果的时间从为12 月15日14:00开始，并在至12月16日14:00结束。预测结果需填入结果表1和结果表2，并且在论文中需提供线路编码为“场地3- 站点83–0600”和“场 地3- 站点83–1400”的预测结果。

**问题2**：以问题一输出的结果为根据，确定当天的运输需求，涵盖各个线路的发车量、预计发运时间以及串点规划，要求任意需求的发运时间在该线路的发运节点之前。以上述需求的所得结果，确定各个需求的承运车辆；其中自有车的周转率和各车辆携带包裹数量越高越好 总成本越低越好。将包括：线路编码、预计发运时间、承运车辆的结果写入结果表3中，并且在论文中需提供线路编码为“场地3- 站点83–0600”和“场地3- 站点83–1400”的调度方案。

**问题3**：使用一种标准容器，可以将装车及卸车时长卓著缩短至10分钟，进一步利用自有车；但有降低车辆装载量至800个的缺陷。以改容器数量无限的前提，根据问题一输出的结果，保留问题2的优化指标，再次确定运输需求。在问题2输出结果之上，另外输出各运输需求是否使用此容器。结果填入结果表4，并且在论文中需提供线路编码为“场地3- 站点83–0600”和“场地3- 站点83–1400”的调度方案。

**问题4**：评估当与实际结果对比的问题1的货量预测结果出现偏差时，在问题3的条件下，该预测对优化后调度模型的影响。

**二、问题分析**

**2.1对问题一的分析**

数据清洗步骤对于各线路不发货时间段的货量缺失值进行补全为0的操作。因此将附件二的数据以日期排序，对于各个线路编码进行从11.30到12.15的货量统计。我们使用XGBoost提升树模型以10分钟为颗粒度，以12月1日至12月14日货量数据以10分钟颗粒度作为训练数据对模型进行训练，并将12月14日到15日一天的数据作为测试集检验模型的性能，最终模型训练集占90%，测试集占10%。然后利用训练好的模型对每个分拣中心分别预测未来一天的日货量。

2.2 对问题二的分析

在数据准备方面，先将可配对站点整理为集合，将车队-规模信息转化为字典，以便后续查找问题。二需要确定预测的未来一天的各需求发运方案，首先以10分钟为颗粒度划分时间片统计货量，确定在累积时间片的运输需求下确定最终发运时间。以下为约束条件分析：若要合并超出原卡车容量的i需求与j需求（相加不超过总容量），我们设定等待i需求等待j需求的时间不超过一个时间片，并且线路可串点，则将他们看作是一个新需求，以i和j需求到达时间的最大值为最终记录数据。考虑车辆调度的约束，若调度的自有车辆在发运该需求前已被调度，要确保该车是可用的，且其到达起点时间早于该需求发运时间；若需调度空闲车辆，则要确保已用车辆少于该车队容量。以上条件均不符则承运外车。考虑时间约束，一是首次使用某车辆，则考虑需求到达站点时间；若是重复使用某车辆，则考虑其上一次返回站点的时间。通过以上约束，我们建立目标函数最小化总成本来找到最优的决策方案。

2.3 对问题三的分析

在问题2的求解基础上，需要加上对数量无限制的标准容器的考虑。该容器优缺点明显，优点是装卸货时间大大减少，缺点是单次包裹装载量有所降低。因此，在设计第三问的优化方案时，对于线路需求，我们采取初始化标准容器需求、串点需求合并最后是采用普通方案的顺序来调配；对于车辆调度，则采取可用车辆、新车辆、标准容器和外部车辆的优先级来调度。通过题目二的约束和本问新加条件，们建立目标函数最小化总成本来找到最优的决策方案。

2.4 对问题四的分析

首先明确预测货量出现偏差有高估和低估两种情况，我们将模型预测情况分为-30%、-20%、-10%、基准、10%、20%、30%这六种情况来观察预测偏差对于优化调度的影响。将这六种情景以系数为0.7、0.8、0.9、1、1.1、1.2、1.3的预测数据输入模型并得到以外部车辆数、自有车周转次数、车辆均包裹量和总成本为考量的影响可视化图。

**三、 模型假设**

本文将作如下假设，以便于模型的建立与求解：

1、假设所给的历史货物数据均为真实数据。

2、假设货量的变化受地域差异、促销活动、站点收发时间差异等因素的影响。

3、假设对数据中缺失值的处理方式不会对预测结果造成太大的误差。

4、假设历史数据符合时间序列，存在一定周期性。

5、假设货量预测与所在线路的变化相关，且这种关系可通过历史数据量化。

6、假设外部车辆和标准容器的使用不受限制。

7、假设自有变动成本和外部承运商成本数据为真实数据。

8、假设预测模型使用的随机游走不影响数据的预测。

**四、 符号说明**

符号 说明

前  棵决策树一起对样本 的预测值

正则化项，惩罚更复杂模型

超参数，控制叶子节点数量增加带来的惩罚力度

超参数，控制叶子节点预测值权重的 L2正则化强度

叶子节点数量  
 正则项中第棵树第个叶子节点的预测值

线路合集

时间片（10分钟间隔）

线路L在时间片内的货量

卡车容量

需求的属性

线路最终发车时间

标准容器的容量

普通车辆的装载容量

时间窗口大小（用于时间分桶）

五、 问题模型的建立与求解

**5.1问题一：**

5.1.1 基于XGBoost的分钟预测模型

考虑到以10分钟为颗粒度的各线路货量具有时间序列的季节性和趋势信息[8][9]，我们选择XGBoost[1]模型的XGBRegressor预测模型结合时间序列季节性分解[3]对未来一日的货量进行预测。

极致梯度提升树（eXtreme Gradient Boosting) 是一种提升树模型，即它将许多CART回归树模型集成在一起，形成一个很强的分类器。组成XGBoost的决策树之间具有先后顺序：后一棵决策树的生成会考虑前一棵决策树的预测结果，即将前一棵决策树的偏差考虑在内，使得先前决策树做错的训练样本在后续受到更多的关注，然后基于调整后的样本分布来训练下一棵决策树，以此不断减小预测值和真实值的差距。

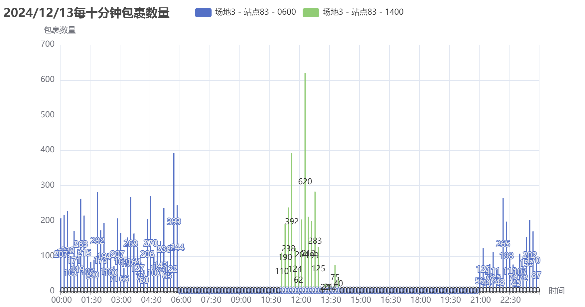
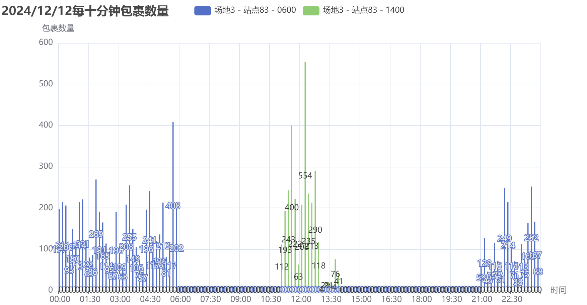
其构建从目标函数出发，选择特征作为节点并确定左右子树权值以建立当前决策树，不断进行上述算法，直至所有特征都被使用或者已经达到限定的层数，则完整的决策树构建完成。

XGBoost的目标函数、正则项以及第i轮加法模型如下：

其中 为损失函数，可衡量预测值与真实值间的差异； 用于控制模型复杂度以防过拟合情况； 和 是超参数，代表叶子节点数量，为第节点预测值。

5.1.2分钟预测模型的求解

在对附件2的数据进行以时间为基准的排序之后我们得到了“场地3- 站点83–0600”和“场地3- 站点83–1400”从12月1日到14日每天的货运量。其中数据以十分钟为单位。



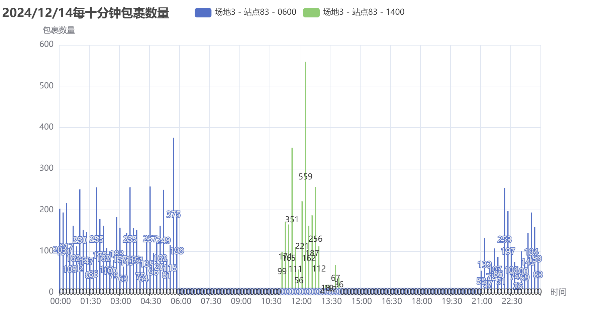


图1 “场地3- 站点83–0600”和“场地3- 站点83–1400”12月12日到12月14日货运量

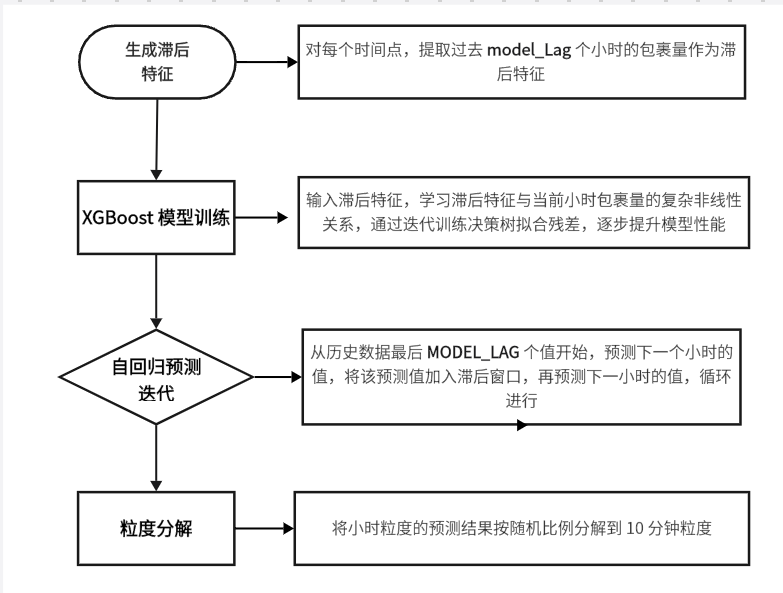


图2 问题一思路

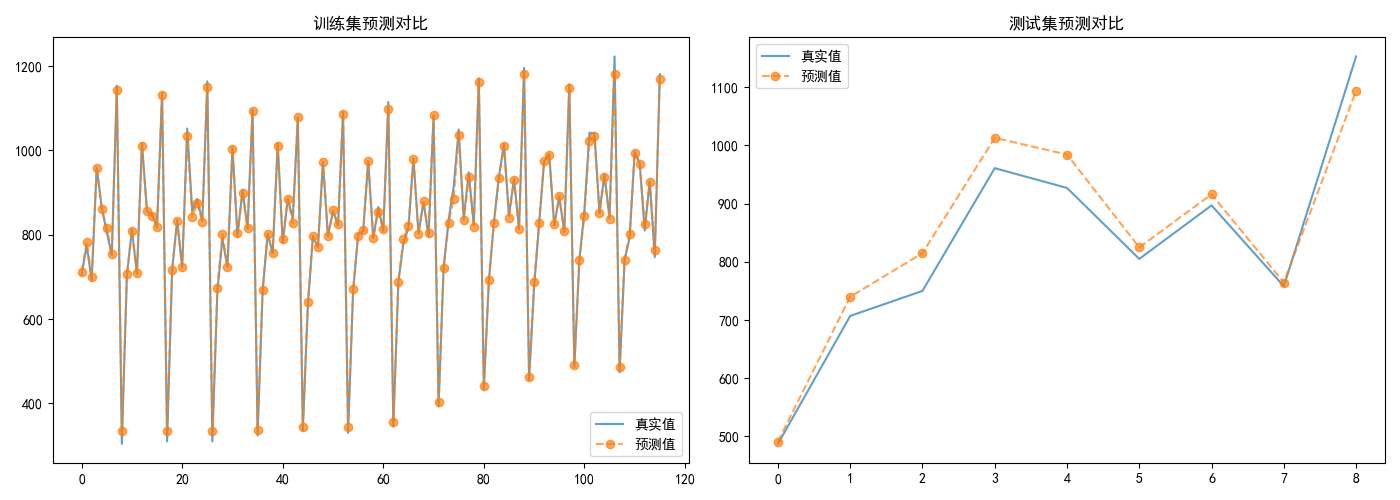


图3 XGBoost模型训练与测试的预测效果展示

通过模型的不断优化，我们最终使用在训练集和测试集上分别为0.964和0.893的决定系数，我们得出线路“场地3- 站点83–0600”和“场地3- 站点83–1400”未来一天的货运量如下图所示

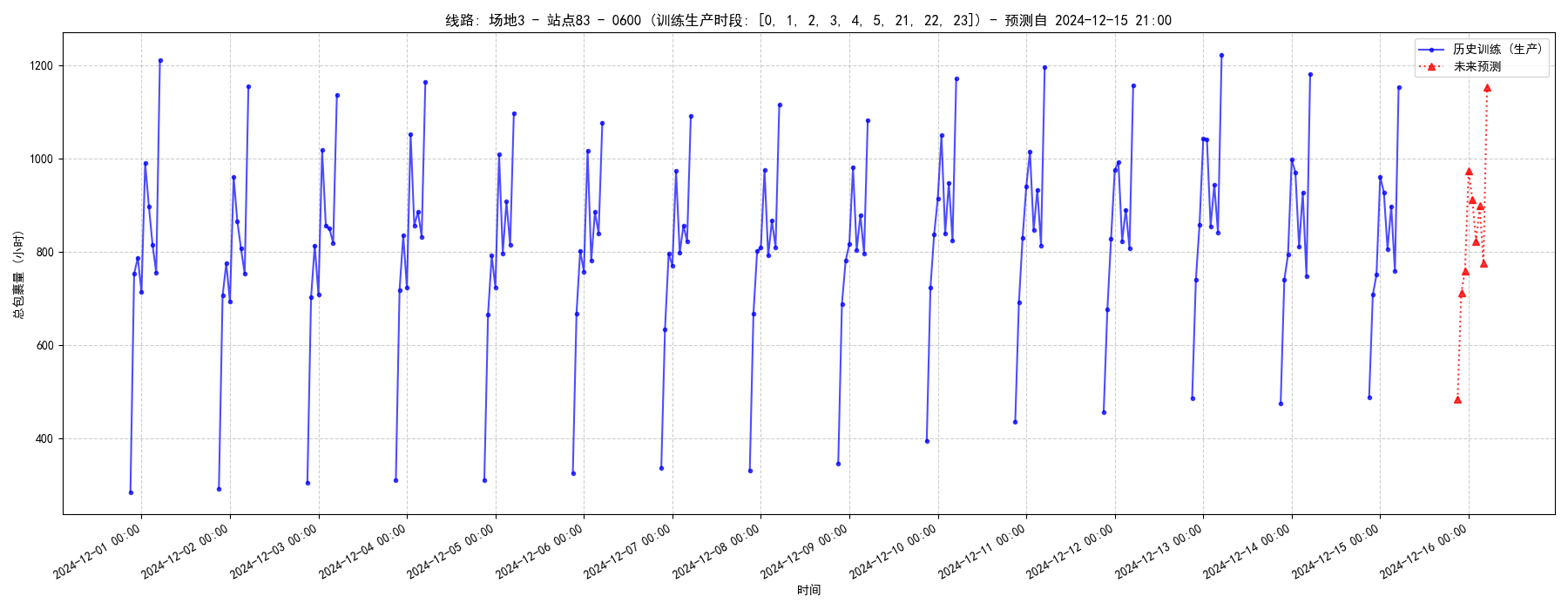


图4“场地3- 站点83–0600”货量变化及预测图

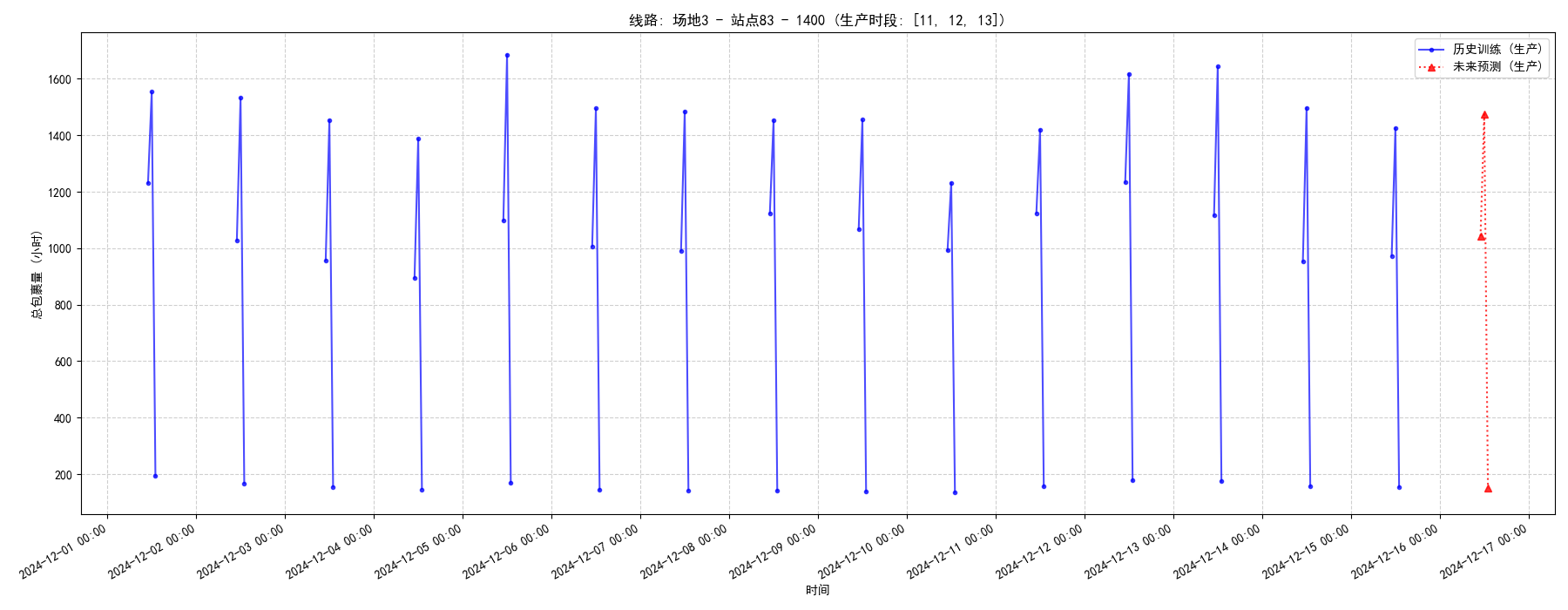


图5“场地3- 站点83–1400”货量变化及预测图

利用XGboost模型[2]，我们得到了了未来一天两条线路日货量的变化情况。由图1.4及图1.5可以看出包裹量变化趋势与历史数据基本吻合，符合生活中的真实情况。数量的波动不仅反映了周期性，也反映了货量受不确定因素的影响。我们将预测结果填入结果表1和结果表2中，其中“场地3- 站点83–0600”和“场地3- 站点83–1400”的预测结果如表所示。

线路编号 日期 货量

|  |  |  |
| --- | --- | --- |
| 场地3 - 站点83 - 0600 | 2024/12/16 | 5531 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 2668 |

表1 “场地3- 站点83–0600”和“场地3- 站点83–1400”日颗粒度预测货量

线路编号 日期 分钟起始 货量

|  |  |  |  |
| --- | --- | --- | --- |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 21:00:00 | 4 |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 21:10:00 | 4 |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 21:20:00 | 8 |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 21:30:00 | 8 |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 21:40:00 | 3 |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 21:50:00 | 9 |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 22:00:00 | 1 |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 22:10:00 | 0 |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 22:20:00 | 0 |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 22:30:00 | 1 |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 22:40:00 | 1 |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 22:50:00 | 1 |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 23:00:00 | 0 |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 23:10:00 | 1 |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 23:20:00 | 1 |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 23:30:00 | 1 |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 23:40:00 | 0 |
| 场地1 - 站点3 - 0600 | 2024/12/15 | 23:50:00 | 2 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 0:00:00 | 14 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 0:10:00 | 17 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 0:20:00 | 17 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 0:30:00 | 15 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 0:40:00 | 9 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 0:50:00 | 12 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 1:00:00 | 12 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 1:10:00 | 49 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 1:20:00 | 13 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 1:30:00 | 50 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 1:40:00 | 17 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 1:50:00 | 8 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 2:00:00 | 19 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 2:10:00 | 0 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 2:20:00 | 84 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 2:30:00 | 85 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 2:40:00 | 60 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 2:50:00 | 19 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 3:00:00 | 17 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 3:10:00 | 4 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 3:20:00 | 34 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 3:30:00 | 182 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 3:40:00 | 49 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 3:50:00 | 72 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 4:00:00 | 26 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 4:10:00 | 55 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 4:20:00 | 129 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 4:30:00 | 152 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 4:40:00 | 61 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 4:50:00 | 14 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 5:00:00 | 94 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 5:10:00 | 57 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 5:20:00 | 53 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 5:30:00 | 232 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 5:40:00 | 149 |
| 场地1 - 站点3 - 0600 | 2024/12/16 | 5:50:00 | 134 |

表2 “场地3- 站点83–0600” 货量分钟颗粒度预测

线路编号 日期 分钟起始 货量

|  |  |  |  |
| --- | --- | --- | --- |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 11:00:00 | 379 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 11:10:00 | 210 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 11:20:00 | 149 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 11:30:00 | 236 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 11:40:00 | 47 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 11:50:00 | 23 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 12:00:00 | 58 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 12:10:00 | 98 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 12:20:00 | 263 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 12:30:00 | 444 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 12:40:00 | 123 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 12:50:00 | 489 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 13:00:00 | 38 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 13:10:00 | 38 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 13:20:00 | 33 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 13:30:00 | 14 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 13:40:00 | 3 |
| 场地3 - 站点83 - 1400 | 2024/12/16 | 13:50:00 | 23 |

表3 “场地3- 站点83–1400” 货量分钟颗粒度预测

**5.2问题二**：

5.2.1 基于规则和启发式算法的调度优化

根据题目已知条件，我们需要选择满足**自有车周转率尽可能高**、**所有车辆均包裹尽可能高**、**总成本尽可能低**的最优方案来为线路调度车辆[5]。我们通过启发式算法[7]来求解这一问题。

模型构建与求解过程

1. 确定决策变量

在车辆调度问题中，决策变量是影响调度方案的核心因素。我们定义以下决策变量：

为可用内部车辆集合，表示当前可以用于调度的自有车辆。

为可新调配的内部车辆集合。

若需求和满足合并条件，则决策生成新需求

其中， 和 分别为需求 和 的发运时间， 和 分别为需求 和 的包裹量， 和 分别为需求 和 *涉及*的站点集合。

**2、确定目标函数**

调度优化的目标是最小化总成本，包括自有车辆的变动成本和外部车辆的成本。目标函数定义为：

其中：

为所有运输需求的集合

为指示函数，表示需求 *d* 是否由内部车辆承运：

为内部车辆承运需求 *d* 的成本

为外部车辆承运需求 *d* 的成本

**3、确定约束条件**

调度方案需要满足以下约束条件：

**时间约束**：车辆的可用时间需满足装卸及运输时间要求。具体计算如下：

其中， 为发运时间，为运输时间，为装卸时间。

**容量约束**：单次运输包裹量不能超过车辆容量：

其中， 为线路 在时间 内的包裹量， 为车辆容量

**站点兼容性约束**：合并需求的站点需可配对。即对于需求 和 ，所有站点对 ( ,) 必须属于允许配对的集合 。

**4、 关键公式解释**

1.需求生成公式**：**

初始需求生成考虑线路在时间片内的包裹量及车辆容量:

其中，

为时间片 的起始时间

表示根据累计包裹量计算的满载[6]车次数

为线路最晚发车时间

表示剩余未满载的包裹量

2.需求合并公式：

当两个需求满足时间窗口、站点兼容性及容量约束时，合并生成新需求：

其中，

为允许的时间窗口

和 分别为需求 和 涉及的站点集合

和 分别为需求 和 对应的车队。

3.车辆分配公式

根据车辆可用性和需求时间分配车辆：

其中，

为可用内部车辆集合。

为可新调配的内部车辆集合。

为车队的最大车辆数。

表示外部承运商。

5.2.2 车辆调度优化的求解

根据上述模型，在遍历站点匹配表、各车队数量、附件一各线路数据以及问题1得出的结果表1和结果表2后，部分线路车辆调度如表所示

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 线路编码 | 日期 | | 预计发运时间 | | | 发运车辆 | |
| 场地1 - 站点1 - 0600 | | 2024-12-16 | | 02:50 | 车队1-V05 | |
| 场地1 - 站点1 - 0600 | | 2024-12-16 | | 04:10 | 车队1-V29 | |
| 场地1 - 站点1 - 0600 | | 2024-12-16 | | 05:30 | 外部 | |
| 场地1 - 站点1 - 0600 | | 2024-12-16 | | 06:00 | 车队1-V15 | |
| 场地1 - 站点1 - 1400 | | 2024-12-16 | | 13:10 | 车队1-V25 | |
| 场地1 - 站点1 - 1400 | | 2024-12-16 | | 14:00 | 车队1-V29 | |
| 场地1 - 站点10 - 0600 | | 2024-12-16 | | 05:30 | 外部 | |
| 场地1 - 站点10 - 0600 + 场地1 - 站点2 - 0600 | | 2024-12-16 | | 06:00 | 外部 | |
| 场地1 - 站点10 - 1400 | | 2024-12-16 | | 14:00 | 车队1-V19 | |
| 场地1 - 站点11 - 0600 | | 2024-12-16 | | 03:30 | 车队1-V19 | |
| 场地1 - 站点11 - 0600 | | 2024-12-16 | | 05:10 | 外部 | |
| 场地1 - 站点11 - 0600 | | 2024-12-16 | | 06:00 | 外部 | |
| 场地1 - 站点11 - 1400 | | 2024-12-16 | | 14:00 | 车队1-V22 | |
| 场地1 - 站点12 - 0600 | | 2024-12-16 | | 02:40 | 车队1-V01 | |
| 场地1 - 站点12 - 0600 | | 2024-12-16 | | 04:30 | 外部 | |
| 场地1 - 站点12 - 0600 | | 2024-12-16 | | 05:30 | 外部 | |
| 场地1 - 站点12 - 0600 | | 2024-12-16 | | 06:00 | 外部 | |
| 场地1 - 站点12 - 1400 | | 2024-12-16 | | 13:50 | 车队1-V01 | |
| 场地1 - 站点12 - 1400 + 场地1 - 站点13 - 1400 | | 2024-12-16 | | 14:00 | 车队1-V08 | |
| 场地1 - 站点13 - 0600 | | 2024-12-16 | | 06:00 | 外部 | |
| 场地1 - 站点14 - 0600 | | 2024-12-16 | | 04:00 | 车队1-V28 | |
| 场地1 - 站点14 - 0600 | | 2024-12-16 | | 06:00 | 外部 | |
| 场地1 - 站点14 - 1400 | | 2024-12-16 | | 14:00 | 车队1-V26 | |
| 场地1 - 站点15 - 0600 | | 2024-12-16 | | 03:30 | 车队1-V20 | |
| 场地1 - 站点15 - 0600 | | 2024-12-16 | | 05:10 | 外部 | |
| 场地1 - 站点15 - 0600 | | 2024-12-16 | | 06:00 | 外部 | |
| 场地1 - 站点15 - 1400 | | 2024-12-16 | | 13:30 | 车队1-V05 | |

表4 部分车辆调度结果

5.2.3 结果数据可视化

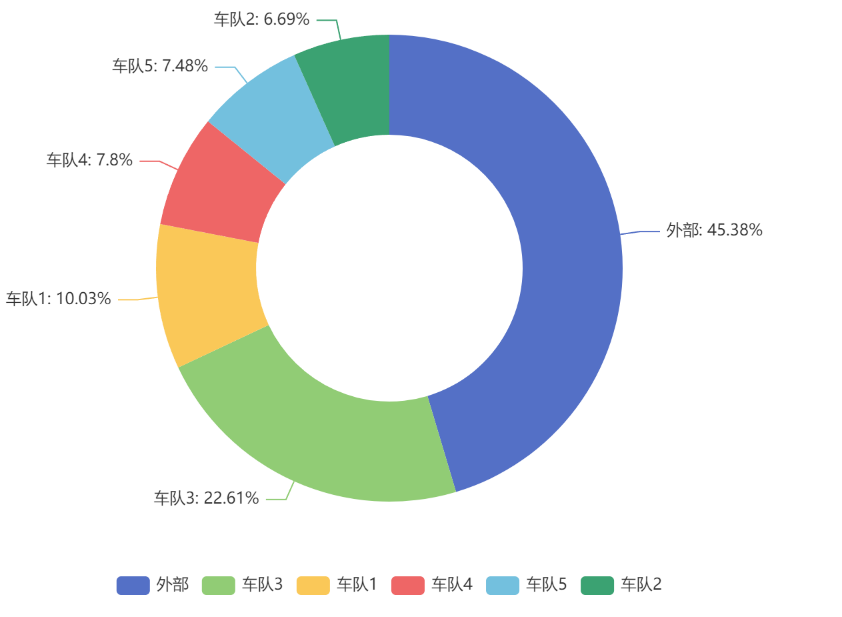


图6 发运车辆来源比例分析

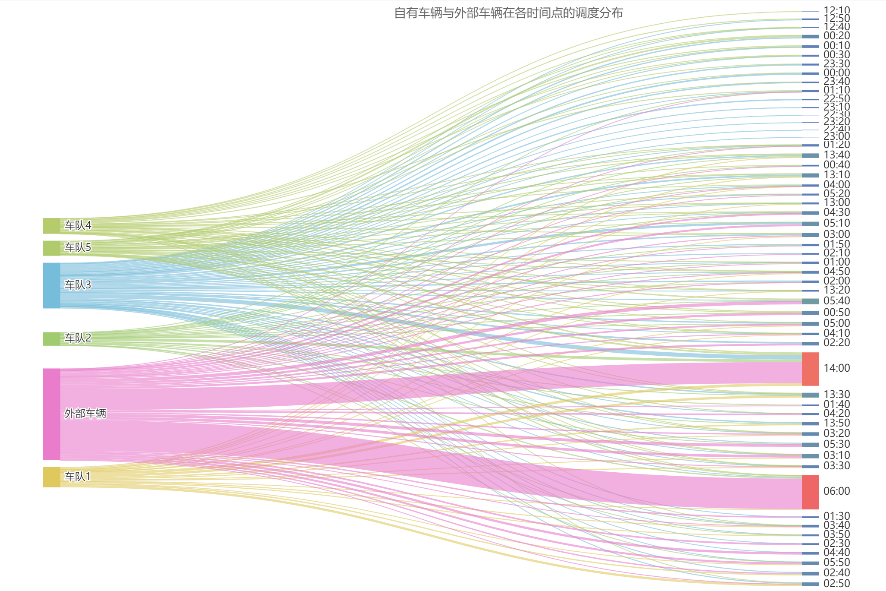


图7 调度资源流动桑基图

由上图可知，自有车辆占比 54.62%，外部车辆占比 45.38%。自有车辆依 “前需求发运时间 + 装车 / 卸车时间 + 往返在途时间＜后需求发运时间” 规则高频周转，车队 3 货量集中、串点效率高，周转率达 3.1 次 / 天，优化成效显著；外部车辆补充串点剩余货量及周转不足场景，平衡成本与效率。调度资源流动桑基图表明，6:00 和 14:00 发运节点集中 58% 调度量，自有车辆占 82%，保障核心时效；00:00-6:00 等非节点时段，外部车辆占 65%，填补周转盲区。串点方案合并 2-3 个线路节点发车，降低空载率，符合 “串点不超 3 个线路” 规则。

**5.3问题三**：

5.3.1基于规则和启发式算法的调度优化

根据题目已知条件，我们需要在问题2的基础上，增加使用标准容器的方案，进一步优化车辆调度[4]方案。我们通过启发式算法对这个问题进行求解。我们首先确定线路使用标准容器的需求列表，对各个需求进行需求匹配以减少车辆使用，然后对使用标准容器的需求尝试去标准容器化以优化车辆配置。在车辆配置阶段，优先分配可用车辆和新车辆的规则与问题2相同，上述分配方式均失败后考虑使用标准容器，否则使用外用车辆。

**1.确定决策变量**

在车辆调度问题中，决策变量是影响调度方案的核心因素。我们定义以下决策变量：

: 是否使用标准容器，取值为布尔类型

：可用车辆集合

：新分配车辆集合

2调度规则优化

我们首先确定线路使用标准容器的需求列表，对各个需求进行需求匹配以减少车辆使用，然后对使用标准容器的需求尝试去标准容器化以优化车辆配置。

**初始化需求**：根据附件中的数据，计算每条线路的总货量，并将其拆解到10分钟颗粒度。基于10分钟颗粒度的数据，生成初始需求列表。初始化需求的公式为：

当时，生成一个集装箱需求。

**需求匹配**：对初始需求进行匹配，方法与问题二相同。对需求进行排序，然后尝试将相邻的需求进行匹配，以减少车辆的使用数量。

**去集装箱化与合并**：如果两个需求满足以下条件，则可以合并：

使用相同的容器: ，

出发时间属于同一个时间窗口:

合并后的包裹量不超过普通车辆的装载量:

**动态容器调整**：在调度过程中，若当前需求未使用容器且其包裹量不超过容器容量，尝试切换至使用容器，以缩短装车和卸车时间，进而提高车辆周转率。若切换后成功分配到可用车辆，则更新车辆的返回时间；否则回退容器使用状态。

**3调度规则：**

在车辆调度时，遵循以下规则：

车辆的返回时间计算公式为：

选择规则为：

即选择返回时间最早的车辆进行分配

**4时间分桶**

为了更高效地进行需求匹配和调度，我们使用时间分桶技术。时间分桶的公式为：

其中， 是时间窗口的大小，这里设置为10分钟。

5.3.2 使用标准容器的车辆调度优化的求解

根据上述模型，在遍历站点匹配表、各车队数量、附件一各线路数据以及问题1得出的结果表1和结果表2后，部分线路车辆调度如表所

线路编码 日期 发运时间 是否使用容器 发运车辆

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 场地1 - 站点5 - 1400 | 2024-12-16 | 14:00 | Y | 外部 |
| 场地1 - 站点6 - 0600 | 2024-12-16 | 02:20 | Y | 车队1-V06 |
| 场地1 - 站点6 - 0600 | 2024-12-16 | 03:50 | Y | 外部 |
| 场地1 - 站点6 - 0600 | 2024-12-16 | 04:50 | Y | 外部 |
| 场地1 - 站点6 - 0600 | 2024-12-16 | 05:30 | Y | 外部 |
| 场地1 - 站点6 - 0600 + 场地1 - 站点25 - 0600 + 场地1 - 站点34 - 0600 | 2024-12-16 | 06:00 | N | 外部 |
| 场地1 - 站点6 - 1400 | 2024-12-16 | 13:40 | Y | 车队1-V23 |
| 场地1 - 站点6 - 1400 + 场地1 - 站点3 - 1400 | 2024-12-16 | 14:00 | N | 外部 |
| 场地1 - 站点7 - 0600 | 2024-12-16 | 03:00 | Y | 车队1-V19 |
| 场地1 - 站点7 - 0600 | 2024-12-16 | 04:40 | Y | 外部 |
| 场地1 - 站点7 - 0600 | 2024-12-16 | 05:50 | Y | 外部 |
| 场地1 - 站点7 - 1400 | 2024-12-16 | 13:30 | Y | 车队1-V28 |
| 场地1 - 站点8 - 0600 | 2024-12-16 | 03:10 | Y | 车队1-V22 |
| 场地1 - 站点8 - 0600 | 2024-12-16 | 04:30 | Y | 车队1-V22 |
| 场地1 - 站点8 - 0600 | 2024-12-16 | 05:30 | Y | 外部 |
| 场地1 - 站点8 - 1400 | 2024-12-16 | 13:20 | Y | 车队1-V10 |
| 场地1 - 站点9 - 0600 | 2024-12-16 | 04:10 | Y | 外部 |
| 场地1 - 站点9 - 0600 | 2024-12-16 | 05:50 | Y | 外部 |
| 场地1 - 站点9 - 1400 | 2024-12-16 | 14:00 | Y | 外部 |
| 场地2 - 站点38 - 1400 + 场地2 - 站点37 - 1400 + 场地2 - 站点39 - 1400 | 2024-12-16 | 14:00 | N | 车队2-V11 |
| 场地2 - 站点39 - 0600 + 场地2 - 站点52 - 0600 | 2024-12-16 | 06:00 | N | 车队2-V12 |
| 场地2 - 站点40 - 0600 | 2024-12-16 | 02:00 | Y | 车队2-V01 |
| 场地2 - 站点40 - 0600 | 2024-12-16 | 04:50 | Y | 车队2-V05 |
| 场地2 - 站点40 - 0600 + 场地2 - 站点37 - 0600 | 2024-12-16 | 06:00 | N | 车队2-V03 |
| 场地2 - 站点41 - 0600 | 2024-12-16 | 01:30 | Y | 车队2-V14 |
| 场地2 - 站点41 - 0600 | 2024-12-16 | 04:30 | Y | 车队2-V06 |
| 场地2 - 站点41 - 1400 + 场地2 - 站点50 - 1400 | 2024-12-16 | 14:00 | Y | 车队2-V09 |
| 场地2 - 站点42 - 0600 | 2024-12-16 | 01:10 | Y | 车队2-V12 |
| 场地2 - 站点42 - 0600 | 2024-12-16 | 03:50 | Y | 车队2-V10 |
| 场地2 - 站点42 - 1400 + 场地2 - 站点48 - 1400 | 2024-12-16 | 14:00 | N | 车队2-V10 |

表5 采用标准容器部分车辆调度结果

（注：是否使用容器列中Y表示是，N表示否）

5.3.3 结果数据可视化

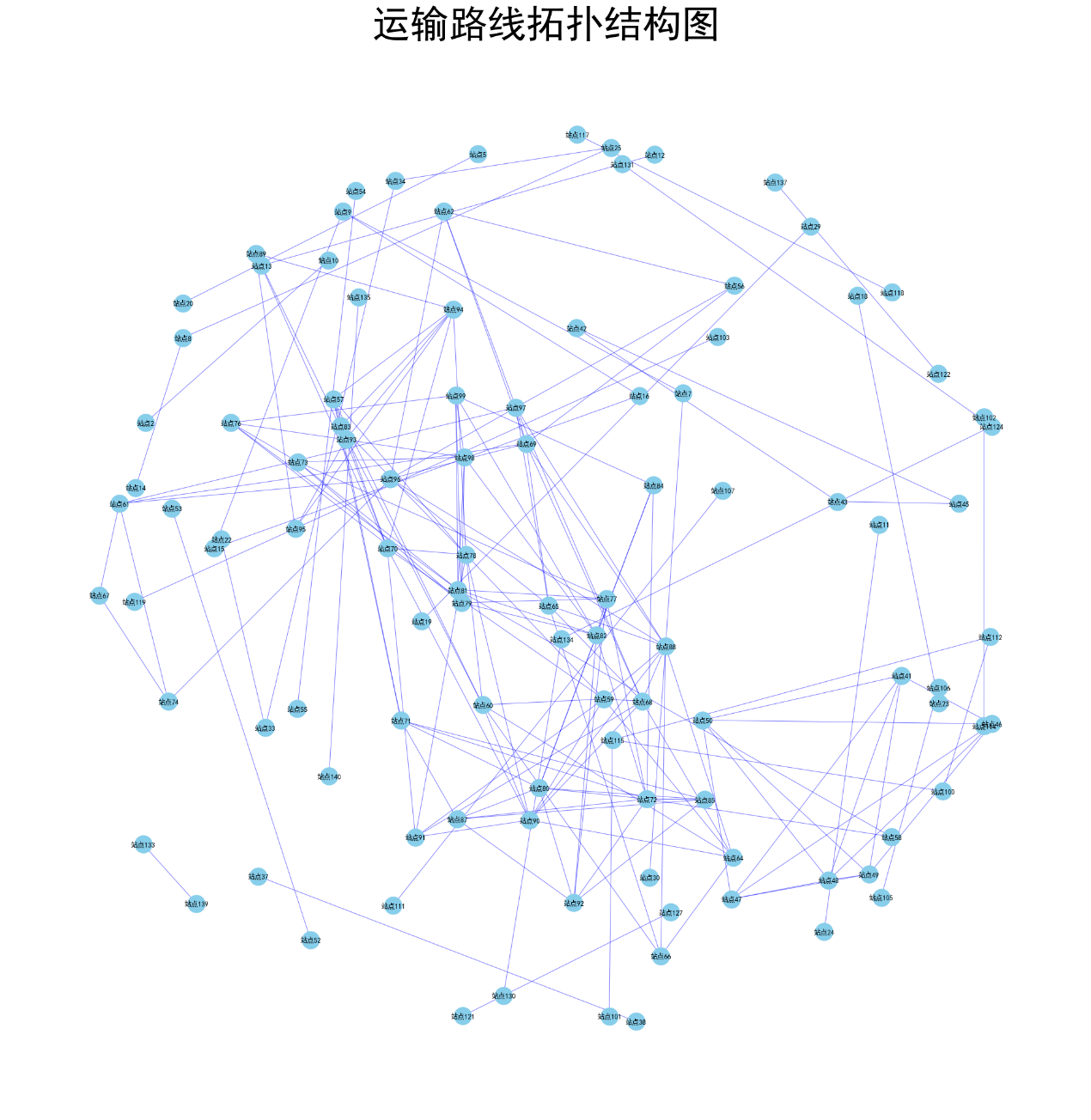


图8 运输线路拓补结构图

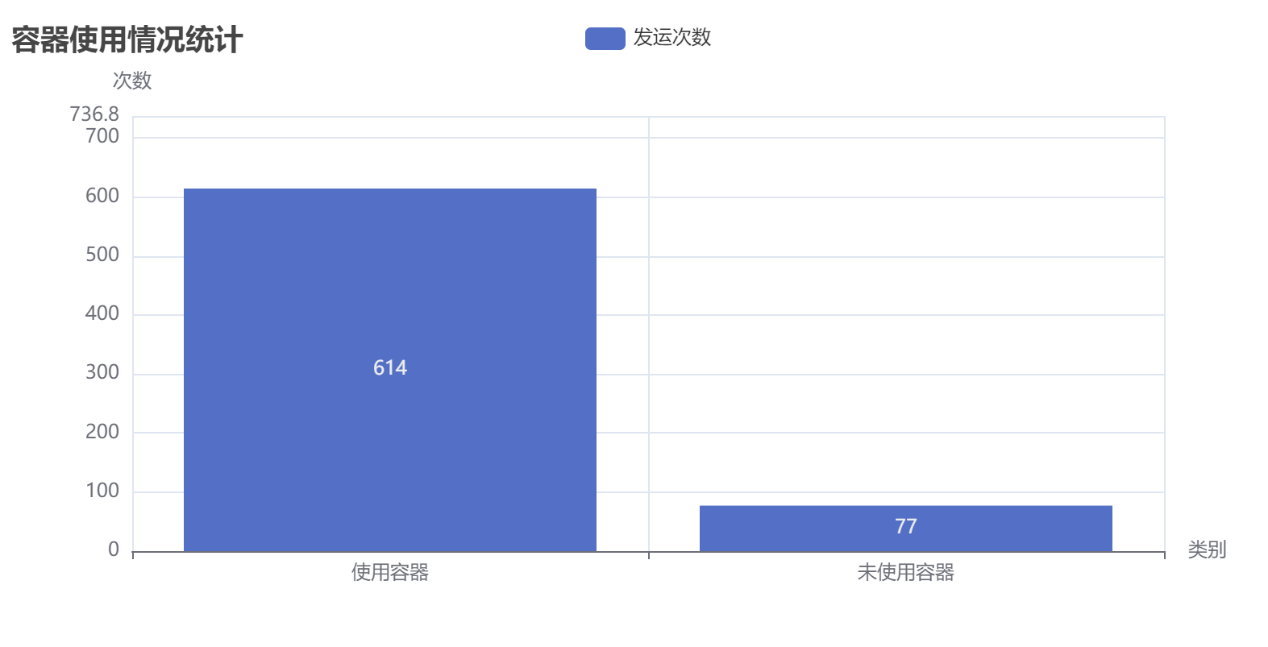


图9 容器使用情况统计图

运输线路拓扑结构图（图8）直观呈现各站点间的连接关系，反映运输网络复杂性，为优化调度策略提供基础。容器使用情况统计图（图9）则清晰对比使用与未使用容器的发运次数，显示使用容器次数（614次）显著高于未使用容器次数（77次），有力证明容器在提升运输效率和优化调度中的关键作用。

**5.4问题四**：

5.4.1不同偏差情况对于优化调度的影响体现:

根据问题需求，最终决定以从低估到高估-30%、-20%、-10%、基准、10%、20%、30%这七种情况来观察预测偏差对于优化调度的影响，将scenarios = {"低估30%": 0.7, "低估20%": 0.8, "低估10%": 0.9,"基准": 1.0, "高估10%": 1.1, "高估20%": 1.2，"高估30%": 1.3}与问题1的预测结果一同作为输入传给问题3优化模型，得到以下将外部车辆数、自有车周转次数、车辆均包裹量和总成本作为考量结果的柱形图。

5.4.2 结果数据可视化

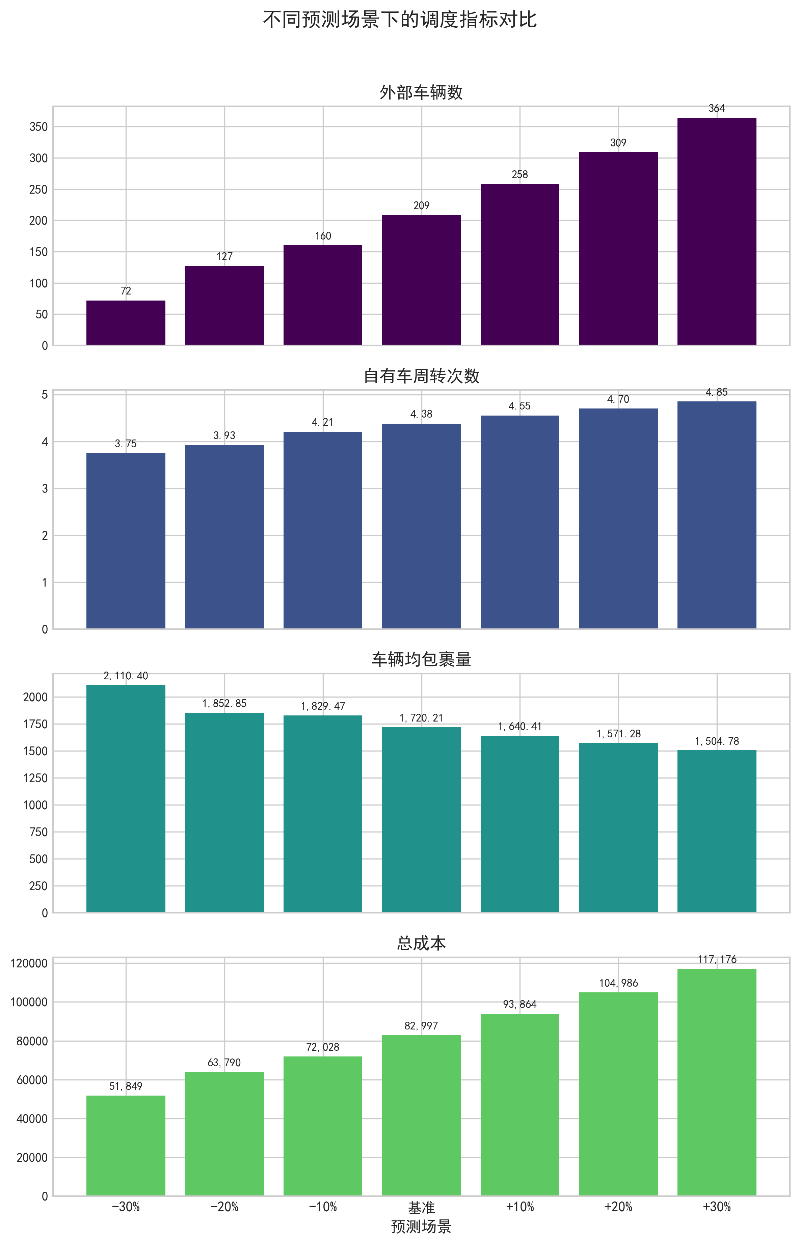


图10不同偏差下预测模型的多个指标结果

从图10中我们可以得出：车辆均包裹量随着模型的预估量增大而不断减少，呈现负相关趋势。说明当预估包裹量较低时，系统为了满足运输需求，可能会相对精准地调配车辆，使得每辆车尽量满载，从而车辆均包裹量较高。随着预估包裹量升高，为了确保所有包裹都能及时运输，可能会优先增加车辆数量来应对需求，而不是等待车辆满载，导致车辆均包裹量下降。

对于外部车辆数来说，在低估场景下，基于相对较少的包裹量预测，依靠自有车辆基本就能满足运输需求，所以对外部车辆的调用较少。随着预估包裹量从基准到高估逐渐增多，自有车辆运力逐渐不足，为了确保所有包裹都能按时运输，就需要调用更多外部车辆来补充运力，因此外部车辆数呈现不断增多的正相关趋势。

而对于自有车周转次数，当预估包裹量处于低估状态时，自有车承担的运输任务量相对较少，在一定时间内完成运输任务后不需要频繁再次投入运输，周转次数低。而随着预估包裹量升高，自有车需要不断往返运输更多包裹，才能应对持续增长的运输需求，这就导致其在相同时间内的周转次数不断增加，呈现正相关。

相应的，总成本主要由车辆使用成本（包括自有车的运营成本、外部车辆的租赁成本等）、人力成本等构成。低估场景下，运输需求小，车辆使用少，相应成本较低。随着预估包裹量从基准到高估，一方面需要投入更多车辆（包括自有车增加周转和调用更多外部车辆），车辆相关成本上升；另一方面，可能还需要增加人力来配合运输、装卸等工作，人力成本也会增加，综合起来使得总成本不断升高，呈现正相关。

**六、模型的评价、改进与推广**

6.1模型的优点

对于问题1的货量预测需求，我们使用梯度提升算法XGBoost，它能够捕捉到数据中的复杂非线性关系，对于货量预测这种可能受到多种因素影响的问题，能够更好地拟合数据，从而提高预测的准确性。并且它可以灵活增加决策树的数量、调整学习率等，来优化模型的性能。

为了更好捕捉线性关系，使用自回归预测方法，有效地利用时间序列的历史信息。我们的模型结合了时间序列季节性分解，避免模型在学习过程中受到季节性和趋势性的干扰，从而提高模型的准确性和稳定性。并在数据预处理层面提取日、

小时和分钟颗粒度的时间相关特征。同时还使用了时间序列交叉验证的模型评估方法，以12月的前14天作为训练集，12月15日作为测试集来训练模型，更准确地评估了模型在时间序列数据上的性能，避免因数据泄露而导致的过拟合问题。

6.2模型的缺点

该模型仅采用包裹量自身的时间滞后值作为特征，未纳入诸如星期、节假日、特殊活动、天气、促销信息等可能影响包裹量的外生变量，这使其难以全面捕捉各类影响因素。此外，在将小时预测量分解至10分钟时，采用的是随机分配比例的方式，而未考量包裹量实际分配分布情况。例如，实际中一小时内包裹量往往在中间10分钟时段达到峰值，但当前方法无法学习到此类规律，仍进行随机分配，进而可能致使10分钟预测结果出现偏差。

6.3模型的改进

由于XGBoost 本身并非专门为时间序列设计的模型，我们在模型优化阶段对于数据预处理部分引入了特征工程强化方法和季节性分解方法来提高预测的准确性。但是更优模型还依赖模型训练阶段的参数调整： 如学习率和树的深度。

6.4模型的推广

考虑到实际运输中不同噪声的影响和各种参数的不确定性，比如由天气导致道路状况不良、运输工具故障等容易导致交付不及时的多种因素，可以将这些随机因素进行全面梳理并确定其影响规律，并为他们赋予相应概率分布作为特征输入模型，让模型学习随机因素与货量之间的关系。还可以将我们的模型和ARIMA时间序列模型集成，加强对货量的基本趋势和季节性规律的捕获，提升预测稳健性。

**七、参考文献**

1. 郭洪鹏,刘斌,肖尧. 基于Bi-LSTM网络的铁路短期货运量预测研究[J]. 铁道货运,2022,40(2):52-58. DOI:10.16669/j.cnki.issn.1004-2024.2022.02.10.
2. 苏飞强. 优化XGBoost模型在城轨站点短期客流预测的应用[J]. 人民公交,2024(20):59-63. DOI:10.3969/j.issn.1674-8050.2024.20.018.
3. 夏进. 基于时间序列分解的短时交通流预测模型研究[D]. 江苏:扬州大学,2023.
4. 李春庆. 基于启发式算法的警用装备调度管理系统研究[D]. 江苏:南京理工大学,2023.
5. 李妍峰,李军,高自友. 动态规划启发式算法求解时变车辆调度问题[J]. 系统工程理论与实践,2012,32(8):1712-1718. DOI:10.3969/j.issn.1000-6788.2012.08.010.
6. 何曼洁,曾连荪. 基于一种特设启发式算法的车辆优化调度问题研究[J]. 电视技术,2019,43(4):1-5. DOI:10.16280/j.videoe.2019.04.001.
7. 李臻,雷定猷. 多车场车辆优化调度模型及算法[J]. 交通运输工程学报,2004,4(1):83-86. DOI:10.3321/j.issn:1671-1637.2004.01.021.
8. 智能供应链:预测算法理论与实战，庄晓天等，电子工业出版社，2023.10
9. 智能供应链:运筹优化理论与实战，庄晓天等，电子工业出版社，2024.10

**八、附录**

|  |
| --- |
| A 问题1 DMC25006414附件.zip\运输规划.py  # Updated TQDM desc  for route\_code in tqdm(all\_route\_codes, desc="处理线路 (固定预测开始时间)"):      try:          route\_data = raw\_data\_base[raw\_data\_base['线路编码'] == route\_code].copy();          if route\_data.empty: continue          # --- 预处理和小时聚合 ---          route\_data['包裹量'] = route\_data['包裹量'].fillna(0).astype(int)          route\_data['时间'] = pd.to\_datetime(route\_data["日期"].astype(str) + " " + route\_data["分钟起始"].astype(str), errors='coerce')          route\_data = route\_data.dropna(subset=['时间'])          if route\_data.empty: continue          valid\_time\_data = route\_data.set\_index('时间')[['包裹量']].sort\_index()          try:              hourly\_agg = valid\_time\_data['包裹量'].resample('H').sum().reset\_index()              orig\_stamps = valid\_time\_data.index.floor('H').unique()              hourly\_agg = hourly\_agg[hourly\_agg['时间'].isin(orig\_stamps)].rename(columns={'时间': '小时开始时间', '包裹量': '总包裹量'})          except Exception as e: print(f"线路 {route\_code}: 聚合失败: {e}"); continue          # --- 按路线过滤生产小时 (用于训练/评估) ---          production\_hours = get\_route\_production\_hours(route\_code)          # Only use production hours for training data          prod\_hourly\_hist = hourly\_agg[hourly\_agg['小时开始时间'].dt.hour.isin(production\_hours)].copy()          prod\_hourly\_hist = prod\_hourly\_hist.sort\_values('小时开始时间').reset\_index(drop=True)          if len(prod\_hourly\_hist) < MIN\_RECORDS\_FOR\_TRAINING:              print(f"  线路 {route\_code}: 生产时段历史数据不足 ({len(prod\_hourly\_hist)} 小时) 无法训练模型 (需要 {MIN\_RECORDS\_FOR\_TRAINING} 小时). 跳过.")              continue          prod\_hourly\_values = prod\_hourly\_hist["总包裹量"].values          # prod\_hourly\_timestamps = prod\_hourly\_hist["小时开始时间"].values # Not directly used for lag features anymore |
| B 问题2 DMC25006414附件.zip\运输规划.py  initial\_demands.sort(key=lambda d: d['depart\_dt']) # Sort by departure time for efficient windowing  merged\_demands: List[Dict[str, Any]] = []  demand\_processed: List[bool] = [False] \* len(initial\_demands)  merge\_time\_window = datetime.timedelta(minutes=MERGE\_TIME\_WINDOW\_MINUTES)  for i, current\_demand in enumerate(initial\_demands):      if demand\_processed[i] or current\_demand['volume'] >= TRUCK\_CAPACITY:          if not demand\_processed[i]: # Add unprocessed full trucks               merged\_demands.append(current\_demand)               demand\_processed[i] = True          continue      merged\_successfully = False      for j in range(i + 1, len(initial\_demands)):          candidate\_demand = initial\_demands[j]          # Early exit if candidate is too far in the future          if (candidate\_demand['depart\_dt'] - current\_demand['depart\_dt']) > merge\_time\_window:              break          if demand\_processed[j] or candidate\_demand['volume'] >= TRUCK\_CAPACITY: continue          if current\_demand['fleet'] != candidate\_demand['fleet']: continue          # Check station pairability          stations1 = {get\_station\_id(line) for line in current\_demand['lines']}          stations2 = {get\_station\_id(line) for line in candidate\_demand['lines']}          stations1.discard(None) # Remove None if get\_station\_id failed          stations2.discard(None)          is\_pairable = True          if not stations1 or not stations2: # Cannot merge if station IDs are missing               is\_pairable = False          else:              for s1, s2 in itertools.product(stations1, stations2):                  if (s1, s2) not in pairable\_station\_set:                      is\_pairable = False                      break          if not is\_pairable: continue          # Check combined volume          if current\_demand['volume'] + candidate\_demand['volume'] > TRUCK\_CAPACITY: continue |
| C 问题3 DMC25006414附件.zip\优化车辆利用率.py  demands\_by\_key = collections.defaultdict(list)  for idx, d in enumerate(demands\_after\_pairing):      # Create a time bucket key (floor to 10 minutes)      bucket\_time = d['depart\_datetime'].replace(minute=(d['depart\_datetime'].minute // 10) \* 10, second=0, microsecond=0)      key = (d['fleet\_id'], bucket\_time)      demands\_by\_key[key].append(idx)  final\_demands = []  processed\_indices\_stage3 = [False] \* len(demands\_after\_pairing)  for idx1, demand1 in enumerate(demands\_after\_pairing):      if processed\_indices\_stage3[idx1]: continue      merged\_in\_stage3 = False      # Try merging only if it's currently using a container and has room to grow      if demand1['use\_container'] and demand1['volume'] < CAPACITY\_CONTAINER:          bucket\_time = demand1['depart\_datetime'].replace(minute=(demand1['depart\_datetime'].minute // 10) \* 10, second=0, microsecond=0)          key = (demand1['fleet\_id'], bucket\_time)          candidate\_indices = demands\_by\_key.get(key, [])          for idx2 in candidate\_indices:              if idx1 == idx2 or processed\_indices\_stage3[idx2]: continue              demand2 = demands\_after\_pairing[idx2]              # --- Check Decontainerization Conditions ---              # 1. Both must currently use containers              if not demand2['use\_container']: continue              # 2. Combined volume must be <= NORMAL capacity              if demand1['volume'] + demand2['volume'] > CAPACITY\_NORMAL: continue              # 3. Assume fleets already match due to key structure |
| D 问题4 DMC25006414附件.zip\p评估影响多组对比.py  scenarios = {      "-30%": 0.7, "-20%": 0.8, "-10%": 0.9,      "基准":  1.0,      "+10%": 1.1, "+20%": 1.2, "+30%": 1.3,  }  scenario\_results = {}  # 3. Run Simulation for Each Scenario  for scenario\_name, multiplier in scenarios.items():      print(f"\n--- Running Scenario: {scenario\_name} (Multiplier: {multiplier:.1f}) ---")      # Stage 1: Generate initial demands      current\_demands = generate\_initial\_demands(          df\_ten\_min\_predictions, df\_route\_info, route\_base\_dates, multiplier      )      # Stage 2: Pair demands (multi-stop)      current\_demands = pair\_demands\_multi\_stop(          current\_demands, allowed\_station\_pairs\_ids # Pass allowed pair IDs      )      # Stage 3: Decontainerize and merge      current\_demands = merge\_to\_remove\_container(current\_demands) # Does not need allowed\_station\_pairs\_ids      # Stage 4: Assign vehicles      assignments = assign\_vehicles\_to\_demands(current\_demands, fleet\_capacities)      # Stage 5: Calculate metrics for the scenario      scenario\_results[scenario\_name] = calculate\_simulation\_metrics(assignments) # Metrics based on assignments      gc.collect() # Clean up memory between scenarios |