基于评价类模型和时间序列分析法的原材料订购运输问题

摘要

生产企业原材料的订购与运输方案是生产成本控制所研究的重要课题。本文应用主流的 TOPSIS 评价模型和自定义的缺失评价模型，从历史订单和供货数据中给予不同供应商合理的评价，并根据评价指标选择出重要性最高的若干供应商。进一步地构造出多种基于时间序列分析法的供应商供货量预测模型、转运商运输方案决策模型，保障了企业在原材料订购和运输方面的最优成本控制，并为企业在现有条件下的原材料供给上限做出了合理预测，为企业提高自身产能等进一步发展提供了合理指导。

针对问题一，为研究不同供货商对于企业的重要性，通过建立基于客观数据分析的" 缺失评价模型"，即通过人工干预的方法去除某家供货商在历史上对于企业的供货量，计算该供货商供货量的缺失对企业产能造成的损失作为该供货商重要性评价的指标，并为总共 402 家供货商按照指标降序的方式排序，筛选出对企业最重要的 50 家供货商。同时建立主流 TOPSIS 评价模型对 402 家供货商进行评价，同样选出 50 家重要的供货商，用于和" 缺失评价模型" 的结果进行比对，最终发现二者选出的供货商仅有一家不同，说明本文建立的缺失评价模型在具有创新性的同时兼具了高度的可靠性。

针对问题二，根据附件 1 中的历史数据，通过构建长度为 24 周的滑动窗口模型，筛选出符合后续预测问题需要的可靠数据区间，通过 0-1 规划模型计算出最少需要 25 家供货商满足企业的产能，回答了第一小问。之后基于时间序列分析法和整数规划法构建了在该可靠数据区间上的供应商供货量预测模型，并根据概率论知识构建供货量和订单量的转换模型，为企业制定出了合理且最优的 24 周原材料采购方案，回答了第二小问。之后同样基于时间序列分析法对不同转运商的运损率进行分析，通过整数规划法构建转运商运输方案的决策模型，回答了第三小问。通过上述模型制定出的采购方案和运输方案效果均达到理论预期，具体方案详见附件 A，附件 B。

针对问题三，需要在问题二中构建的基于时间序列分析的规划模型中，增加题干中要求的多采购 A 类材料且少采购 C 类材料的约束条件，其他求解均可参照问题二。具体方案详见附件 A，附件 B。

针对问题四，通过上述问题二、三建立的分析模型，对供货商和转运商进行分析发现，在当前条件下该企业每周所能入库的原材料的最大值受限于 8 家转运商的最高运力，即 $6000\times 8 = 48000m^3$，根据此约束条件，运用整数规划模型，制定了相应的原材料采购模型和转运模型，进一步求解出相关方案，详见附件 A，附件 B。计算出该企业每周产能的上限为 $35747.06m^3$，相较于之前可以提高 26.76%。

综上所述，本文在多种不同尺度的约束条件下，均构建出基于时间序列分析的订购和转运方案制定模型，并求出相应约束条件下生产成本最低的方案。该套模型具备极优的规划能力和极强的适应性，可以将其推广至面向实体制造的全部工业界，并为企业估测自身产能上限，制定发展战略提供了重要的参考价值。

关键词: TOPSIS 评价模型　时间序列分析法　Python 线性规划　概率论　机器学习

目录

问题提出

生活中存在的工业产品种类丰富，其中许多都是由相关企业从简单的原材料经过各种步骤逐步加工而来。对于一家生产企业而言，如何制定原料采购和转运方案是控制生产成本的重要课题。

1.1 问题背景

现已知有一家生产企业一年有 48 周需要生产，每周产品的产能为 2.82 万立方米，企业需要根据产能选择不同的供应商和转运商，并提前 24 周确定原材料的订购和运输方案。

对于原材料的订购而言，已知有 402 家供应商可以为企业提供 A、B、C 三种类型原材料用于产品生产，每立方米的产品可以由 $0.6m^3$ 的 A 类材料，或者 $0.66m^3$ 的 B 类材料，或者 $0.72m^3$ 的 C 类材料独立生产而来，而 A 类和 B 类每立方米的采购成本则分别比 C 类高 20% 和 10%。

对于原材料的运输而言，已知有 8 家运输商承包供应商到生产企业的转运任务，每家运输商每周运输能力为 6000 立方米，而且运输过程中三种材料均会有一定损耗。

此外，企业为了保障生产，要求每周原材料的库存量要尽量满足至少未来两周的生产需求，所以该企业会把供应商所有的供应全部买下，而且在转运过程中要尽量保证一家供应商的原材料由一家运输商转运。附件 1 中保存了历史五年中 402 家供应商在 240 周内收到的订货量与实际的供货量数据，附件 2 保存了 8 家转运商在 240 周内的运输损耗率数据。

1.2 问题重述

（1）根据附件 1，用数学建模的方法选择对于保障企业生产最重要的 50 家供应商。

（2）根据上题，求解至少要多少家供应商才能满足企业的生产需求。针对这些供应商，为企业制定未来 24 周最经济的原材料订购方案，并在此基础上选择转运商制定最少损耗率的转运方案，分析方案实施结果。

（3）为减少仓储和转运成本，该企业在满足生产条件的前提下打算尽量少地采购 C 类原材料而尽量多地采购 A 类材料，同时尽可能减少产品转运过程中的损耗率，在此条件下为企业制定 24 周的原材料订购和转运方案，分析实施效果。

（4）根据附件 1 中供应商和运输商的情况，判断该企业可以提高多少的产能，并制定 24 周的订购和转运方案。

问题分析

题目要求根据附件 1 和附件 2 的数据，以及每个子问题的不同限制条件，给出在该条件下的最合理的产品订购和转运方案。可以通过多种评价类模型建立评价体系，然后根据不同规划问题求得每个问题的最优解，具体分析如下：

2.1 问题一的分析

问题 1 需要确定对于保障企业生产最重要的 50 家供货商，可以通过“评价 + 决策”模型来求解，利用附件 1 中的数据确定评价指标，然后根据这些指标构建评价模型来计算每个供应商的得分，最后排序获得前 50 名供应商。

所谓保障企业生产最重要的 50 家供应商，就是其供应量对企业每周产能影响最大的 50 家供应商，因此本文通过研究每家供应商对于企业产能的影响大小来筛选出影响最大的 50 家供应商作为结果。分析思路如下：在研究某家供应商的影响重要性时，可以将其先从供应商列表中删除，也即假设其供货量在 240 周内均为零，此时企业的产能相比未去掉该供应商时必然有所下降，因此建立相应模型评价企业产能的下降水平便可间接体现该供应商的重要程度。

根据上述思路，本文建立了“缺失评价模型”：在不考虑运输损耗的条件下，选择一个供货商，假设其供货量恒为零，然后计算生产企业相比原来产能的降低程度，程度越大，表明该供货商其对于企业生产越重要。通过这种方为 402 家供应商打分，选择分数最高的 50 家变为所求解。

此外，为了衡量“缺失评价模型”的评价效果，本文又建立了一个基于 TOPSIS 分析的模型与之比较。分析附件 1 中的数据，本文总结出三个比较合适的评价指标：供应商的供货量、供货稳定性、供货原材料类型。理由如下：供货量直接反映供应商对生产的重要性，供货量越大，重要程度越高；选取订货量与实际供货的差别作为衡量供货稳定性的指标，若某家供应商的订货量与供货量的差别越小，说明该供应商对于企业订单响应的稳定性越高，企业会更加偏好选择该供应商，因此其对于生产企业的重要性也越高；对于供货商的供货材料而言，由于相同数量的原料最终能够转化为产品的量是不同的，且每家供货商只能供应一种原材料，因此供应不同原材料的供应商的重要性也不同，具体重要性为供应 A 类的 > 供应 B 类的 > 供应 C 类的。

通过对两个不同视角下的模型的评价结果进行比对，可以确定最重要的 50 家供应商。

2.2 问题二的分析

问题二一共有三个小问题，下面针对每问进行具体分析。

第一问需要求解满足生产条件的最少供应商数量。由于在问题一中已经求解出最重要的 50 家供应商，且经过简单计算发现这 50 家供应商的供货总量在某些时间段已经可以满足该企业的基本生产需求，因此本文合理假设由于剩余三百余家供应商的供货能力较弱，不可能出现在本题所求的可行解空间内，也即问题一选出的 50 家供应商一定涵盖了本问所要求解的所有供货商。因此题目可以转化为在问题一中确定的 50 家供应商中选择尽可能少的供应商，满足该企业的生产需求。

由于本题第二问要求制定未来 24 周的订购计划，因此需要确定未来 24 周供应商的供货计划，而供应商的供应能力是随着时间的变化而变化的，因此需要针对 24 周的长周期来进行整体规划并确定其中每周的不同供货情况，而不能忽略供货情况的时间分布差异只是简单制定一周的供货计划然后重复 24 次。因此作为第二问铺垫的第一问也需要在求解时考虑能持续满足 24 周企业产能需求的最少供应商数量，而不能只是简单计算历史 240 周内某周出现过的最少供应商数量。此外，由于选出的最重要的 50 家供应商的供货总量确实低于全部 402 家的总量，因此会在历史 240 周内部分时间不满足生产需求，使得满足条件的周和不满足条件的周穿插交错，不利于我们研究连续 24 周的供应商供货情况，为此我们引入”滑动窗口“的概念，通过设置一个长度为 24 周的滑动窗口在 240 周的数据中”滑动“筛选 50 家供应商连续满足 24 周生产条件的时间段，并以这些时间段的数据作为基础来进一步求解满足生产条件的最少供应商数量。此时便可运用” 0-1 规划模型“完成第一问的求解。

第二问中需要针对第一问选出来的供应商制定 24 周的最经济的采购方案，由于我们在第一问中已经考虑了供应商在 24 周内供应能力的变化，并通过历史数据计算了不同供应商在一个计划周期（24 周）内每周供货能力的峰值，因此可以用上述随时间变化的峰值作为所选供应商在 24 周内

每周供货能力的上限，通过建立合适的”整数规划”模型，并考虑其他约束条件，便可轻松求解第二问。

第三小问可以通过相似的“整数规划“模型求解损耗最小的转运方案，具体分析见后文模型建立与求解。

2.3 问题三的分析

问题三中需要问题二考虑最经济的原料订购方案的基础上考虑转运和存储成本，以进一步降低生产成本。

问题三相较于问题二不同的约束条件为：尽量多的采购 A 类原材料，尽量少的采购 C 类原材料，该约束条件的合理性在于：用同样的价格购买 A 和 C 从事生产，最终获得的产品数量是相同的，而 C 所占的体积更大，因此会提高储藏成本。由于这一题同样需要制定订购方案和储存方案，因此可以参考借鉴问题二中后两个小问题所建立的模型构造对应的”整数规划“模型，通过改变约束条件进行求解。为了在模型中反映该约束条件，在计算总的原材料成本时，相较于采购单价，我们可以为转运和储藏成本设置更高的权重，这样模型就会偏向于选择用 A 类的原材料而不是 C 类材料。

2.4 问题四的分析

问题四求企业能够提高的产能，显然是需要计算在全部供应商都供货时的最高产能。可以利用历史数据为每家供应商设定供应量的上限作为约束条件，并建立相应的规划模型，求解在供应商最大供应能力内和转运商最大转运能力内的最大供货量，并计算相应的最大产能，最后计算出企业产能提升的上限。

模型假设

1. 为了保证正常生产的需要，原材料的库存量要尽量不少于两周生产的需求
2. 全部收购供货商实际提供的原料
3. 每家运输商的运输能力为定值，6000 立方米/周
4. 每家运输商在一周内的运输损耗率为一定的
5. 产品需要的原材料消耗不会改变，材料的价格比不会改变
6. 供求链稳定，不会有任意一方破产倒闭
7. 每家供货商的供货能力与时间有关

符号说明

符号	说明	单位
P_t	企业每周的总产能 (2.82)	万立方米
$type$	供应材料类型	
w	原料对应的产品转化率	立方米
T_a	转运商的运输能力 (6000)	立方米/周
n	供应商个数 (402)	
m	5 年中的生产周 (240)	
t	运输商个数 (8)	
B	订货数据矩阵	
b_{ij}	第 i 个供应商的第 j 周订货数	
S	供货数据矩阵	
s_{ij}	第 i 个供应商的第 j 周供货数	
L	损耗率数据矩阵	
l_{ij}	第 i 个运输商的第 j 周损耗率	%
$stock_i$	第 i 周开始时的原料库存量	立方米%

模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 模型的建立

首先建立缺失评价模型，需要计算把某一家供应商从原本的数据中删除后对产能的影响，记 pro_{ij} 为把第 i 家供应商删去后的第 j 周产能，可以得出：

$$pro_{ij} = \begin{cases} \sum_{k=1}^{n} \frac{s_{kj}}{w_k} - \frac{s_{ij}}{w_i} + pro_{ij-1} - P_t & j \geqslant 1 \quad \text{and} \quad pro_{ij-1} \geqslant P_t \\ \sum_{k=1}^{n} \frac{s_{kj}}{w_k} - \frac{s_{ij}}{w_i}, & j = 1 \quad \text{or} \quad pro_{ij-1} < P_t \end{cases} \tag{1}$$

其中 n 为供应商的个数 402，P_t 为当前企业每周的总产能 2.82 万立方米，s_{ij} 为第 i 家供应商在 j 周的供货量，s_{ij} 为第 i 家供应商在 j 周的供货量，w_i 为第 i 家供应商的原材料类型对应的产品转化率，则有

$$w_i = \begin{cases} 0.60 & type_i = A \\ 0.66 & type_i = B \\ 0.72 & type_i = C \end{cases} \tag{2}$$

其中 $type_i$ 为第 i 家供应商的材料类型，所以在不考虑损耗率的情况下，$\sum_{k=1}^{n} \frac{s_{kj}}{w_k}$ 即为把第 j 周的供货转化为产品的量，然后再减去第 i 家供应商的贡献。当前一周的产能大于目标的时候，多余产能作为库存在下一周加入计算；若前一周没有满足产能要求或当前为第一周，当周库存就为 0。

然后计算供应商对产能的影响程度 E_i，

$$E_i = \sqrt{\frac{\sum_{j=1}^{m} ProD^2(pro_{ij})}{m}} \qquad ProD(pro_{ij}) = \begin{cases} P_t - pro_{ij}, & pro_{ij} < P_t \\ 0, & pro_{ij} \geqslant P_t \end{cases} \tag{3}$$

其中 $m = 240$，表示 240 周的总量，E_i 为通过平方平均得到第 i 家供应商对于产能的影响得分，函数 $ProD(x)$ 为未达标的产能与目标的差值。差值进行平方运算可以放大这一差值的影响，使得在某一周的总供货量具有巨大占比的供应商也能获得较高得分。

为了和上述缺失评价模型进行比对，下面建立 TOPSIS 评价模型，其三个评价指标分别**供货量**，**稳定性**和**材料类型**。

对于**供货量指标**有：

$$SumS_i = (\sum_{j=1}^{m} \frac{s_{ij}}{w_i}) \qquad AverageS_i = SumS_i/m \tag{4}$$

其中 $m = 240$，$SumS_i$ 为第 i 家供货商在 240 周内的总供货量，$AverageS_i$ 为该供货商在 240 周内的平均供货量。

对于**稳定性指标**，考虑企业订货量与供应商针对该订单实际供货的差值，差值越大说明影响越大，所以有：

$$Diff_i = \sqrt{\frac{\sum_{j=1}^{m} (b_{ij} - s_{ij})^2}{m}} \qquad DiffW_i = Diff_i/Average_i \tag{5}$$

其中 $Diff_i$ 为采用平方平均的方法计算第 i 家企业的订户量与实际供货的缺货指标，$DiffW_i$ 为经过正规化处理的缺货指标。

通过计算 $Diff_i$ 后我们与实际情况进行比对，发现部分供应商在收到高额订单后的当周供货量为很少（详见图 1，所有供货商在 240 周内的缺货量的折线图），在此模型中这些供货商的稳定性得分将变得非常低，导致直接被排除，这不符合常理，我们需要对模型进行调整。由于供货量的增多，缺货数量也会增大，为了正规化指标，这里再除以第 i 家企业的平均供货量 $Average_i$，得到 $DiffW_i$，

代入 $Diff_i$ 后，不难发现 $DiffW_i$ 关注的指标时缺货数量与供货数量的比值，即缺货率。

$$DiffW_i = \sqrt{\frac{\sum_{j=1}^{m} [(b_{ij} - s_{ij})/Average_i]^2}{m}} \tag{6}$$

对于**原料类型指标**，按照生产单位产品所需的 A, B, C 类原材料的体积 $0.6m^3, 0.66m^3, 0.72m^3$ 进行赋值。

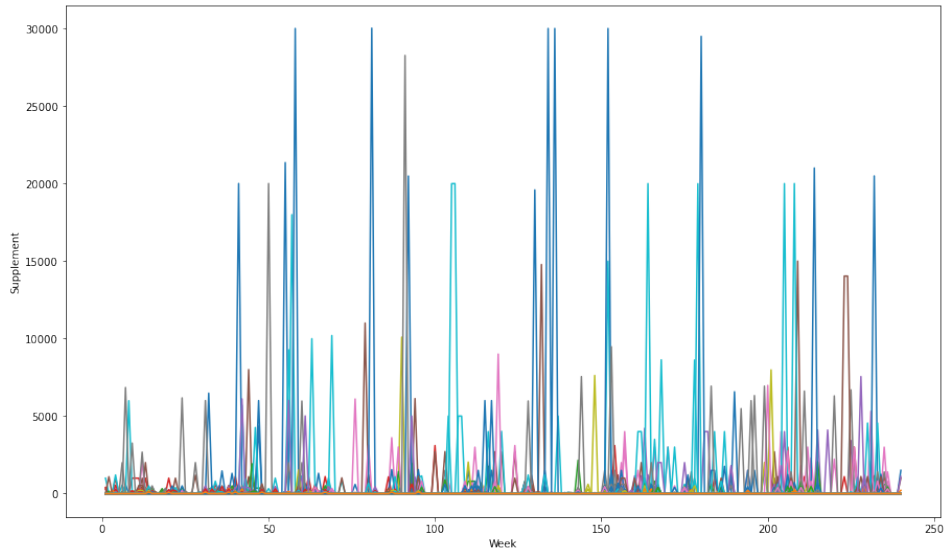

图 1: 订货量与供货量的差值

5.1.2 模型的求解

使用 Python 中程序分别对上述模型两个模型进行求解。

对于缺失评价模型，我们根据其建立的过程，用程序对其进行直观模拟（详见附录 Listing**??**中的代码），在得到所有供应商的得分后，选出前 50 个供应商。

对于 TOPSIS 模型，根据附件 1 的数据得到了 402×3 的评价矩阵，即 402 家企业在 3 个评价指标上的得分构建的评价矩阵，然后对其中的数据进行正规化和正向化，这样一来，评价矩阵所有的数据都在 0 到 1 之间，且三个指标都是极大型指标，即指标值越大得分越高。

然后用 TOPSIS 算法，在评价矩阵三列每一列求出最大值和最小值，把获得三个最大值组成一个三维向量，称为最大点，同理获得最小点。根据评价矩阵每一行的三维向量和最大点和最小点的距离来计算得分，离最大点越近，得分越高，即为对应供应商的得分（算法详见附录 Listing2中的代码）。

用程序对结果进行比较，发现两个模型选出的 50 家供应商只有一家不同，缺失评价模型选择的是第 150 家而 TOPSIS 模型选择的是第 273 家供货商，两个模型的结果高度吻合，可以认为两个模型的实际效果很好。

最后把两个模型得到的 50 家供应商在 240 周内的供货量作折线图（见图 2），从图像也可以看出结果基本一致。

由于缺失评价模型（右图中的 Missing-Assessment-Model）建立过程中的采取主观指标较少，不妨按照缺失评价模型给出的 50 家供应商来作为最终的结果，并以此作为后续建模的数据，现给出排名表格（排名越小，得分越高）：

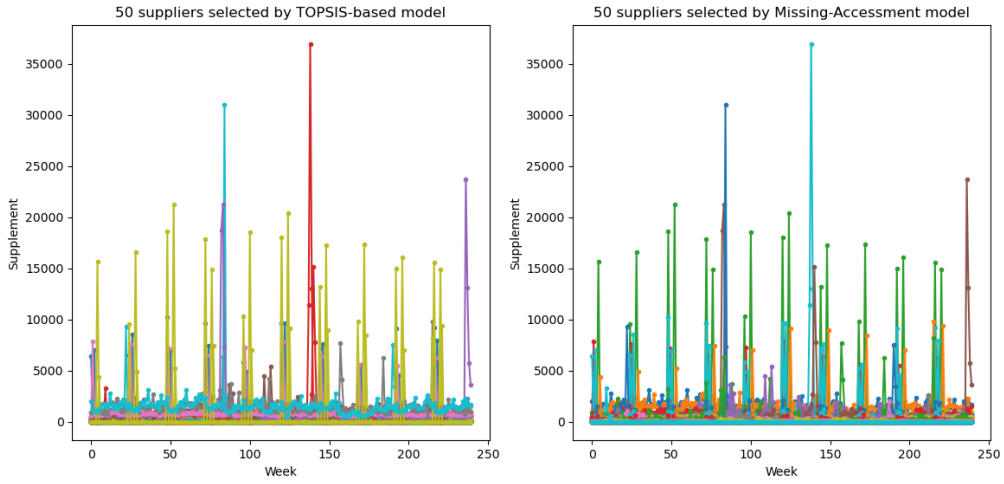

图 2: “TOPSIS 模型”和“缺失评价模型”的比较

1 - 5	6 -10	11-15	15-20	21-25	26-30	31-35	36-40	41-45	46-50
228	150	130	305	200	125	337	373	188	291
360	274	329	351	306	283	363	73	243	217
139	328	307	142	394	30	366	79	209	207
107	339	355	193	246	364	54	293	113	2
281	138	267	347	36	39	345	85	77	149

表 1: “缺失评价模型”得出的前 50 名供应商

5.2 问题二模型的建立与求解

5.2.1 第一问模型的建立

先选择问题一求出的 50 家供应商，获得 50×240 的数据矩阵，然后固定通过 24 周的滑动窗口，遍历所有连续的 24 周，判断哪 24 周满足约束条件，即满足两周生产需求的原材料库存量。然后把满足条件的 50×24 矩阵 *data* 中的原料供量转变成产品供量来进一步简化计算，以此作为下一步的输入。

在获得了 *data* 后，利用 0-1 规划求解，目标函数为：

$$\sum_{j=0}^{49} status_j, \qquad status_j \in \{0, 1\} \tag{7}$$

其中，$status_j$ 指的是是否选取 *data* 中的第 j 家供应商，也就是决策变量，0 表示不选，1 表示选取，下标 j 从 0 开始，到 49 结束。显然当函数 T 最小时，选取的供应商最少。

然后是约束条件，首先是每周的库存 $stock_i$ 需要满足一定的限制

$$stock_i = \begin{cases} 0 & i = 0 \\ stock_{i-1} + supply_{i-1} - P_t & i > 0 \end{cases} \tag{8}$$

其中下标 i 从 0 开始，到 23 结束，$stock_i$ 指的是第 i 周开始时的库存，在第 0 周由于之前没有任何进货，所以 $stock_0 = 0$，$supply_{i-1}$ 指的是第 $i-1$ 周的总供货量，Pt 为每周要达标的产能 2.82 万立方米。

然后是求 $supply_i$，不难得到

$$supply_i = \sum_{j=0}^{49} status_j \cdot data_{ji} \tag{9}$$

又因为有 $stock_i = stock_{i-1} + supply_{i-1} - P_t$，可以得到

$$\begin{aligned} stock_i - stock_{i-1} &= supply_{i-1} - P_t \\ stock_{i-1} - stock_{i-2} &= supply_{i-2} - P_t \\ \vdots \qquad & \qquad \vdots \\ stock_1 - stock_0 &= supply_0 - P_t \\ stock_0 &= 0 \end{aligned} \tag{10}$$

将上式相加得到 $stock_i = \sum_{k=0}^{i-1} supply_k - i \cdot P_t$，把对应的 $supply_i$ 代入得

$$\begin{aligned} stock_i &= \sum_{k=0}^{i} \sum_{j=0}^{49} status_j \cdot data_{jk} - i \cdot P_t \\ &= \sum_{k=0}^{i} (status_0 \cdot data_{0k} + status_1 \cdot data_{1k} \cdots + status_{49} \cdot data_{49k}) - i \cdot P_t \\ &= \sum_{j=0}^{49} (status_j \cdot \sum_{i=0}^{i-1} data_{ji}) - i \cdot P_t, \qquad i = 0, 2, \cdots, 23 \end{aligned} \tag{11}$$

让 $stock_i$ 大于 P_t，即当前周供货量与库存量的和大于等于两倍产能，即认为满足了两周生产需求的原材料库存量，得到最终的约束条件，然后使用 python 的 pulp 库进行 0-1 规划求解。

5.2.2 第一问模型的求解

首先用滑动窗口方法，选取满足条件的从 138 周开始的 24 周，然后利用 Python 的求解 0-1 规划的库函数，根据以下的规划模型进行求解

$$\begin{aligned} &\min T = \sum_{j=0}^{49} status_j \\ &\text{s.t.} \qquad \sum_{j=0}^{49} (status_j \cdot \sum_{i=0}^{i-1} data_{ji}) - i \cdot P_t \geqslant P_t \end{aligned} \tag{12}$$

可以方便地求出 T 最小为 25，也就是说至少需要 25 家才能满足需求，然后绘制 25 家供应商在这 24 周中的供货量的图像 3，绘图代码见附录 Listing4。

图 3: 选中的 25 家供货商在 24 周内的满足生产条件的供货曲线

不难发现，这些供应商能够满足生产条件得益于前几周有一家供应商提供了巨额的供量。

5.2.3 第二问模型的建立

第二问要给出未来的订购方案，因此用预测模型更合适，考虑到供应商供货应该存在季节性的波动，而现在需要给出的是的一年 48 周生产计划中的前 24 周的原料订购计划，所以可以利用 5 年中每年的前 24 周数据作为参考，这样预测的效果可能会更好。

第一问已经获得了 25 家能够满足企业生产的供应商，针对这些供应商，可以先获取它们的 5 年中每年的前 24 周的数据，然后每年相同的周取平均，得到该供应商在该周的平均供货量，最后构成 25×24 的平均供货量矩阵 avg，其中 avg_{ij} 的指的是第 i 家供货商在每年的第 j 周的平均供货量。

经过简单尝试可知，单使用平均供货量为参考无法满足企业的生产需要，所以我们需要在保证其周期性的同时提高供货量。

若选用供货商 5 年中最高的供货量作为上限的话，根据图 2，发现有很多供应商存在少数几次的大量周供货，按照最高供货量作为上限，也就是认为供应商随时可以提供大量的货物的话，不仅不符合实际，而且可能使得最后的规划能够轻易满足约束条件，模型的效果会因此下降。

我们在此用一个介于 0 到 1 之间的系数矩阵乘以最高供货量作为单周的供货上限，系数矩阵可利用矩阵 avg 构造得到 coe，把 avg 的每一个元素除以元素所在行（即同一家供货商在 24 周的不同平均供货量）的最大值来映射到 0 和 1 之间，即：

$$coe_{ij} = \frac{avg_{ij}}{\sum_{i=0}^{23} avg_{ij}} \times 0.88 + 0.12 \tag{13}$$

上式中所乘的参数可使得供货量极大值极小值分布更加平缓，参数可调整。之后由此求得第 i 个供应商第 j 周的供货上限 $bound_{ij}$，其中 $\max(i)$ 为第 i 个供应商 5 年的最高供货量

$$bound_{ij} = coe_{ij} \cdot \max(i) \tag{14}$$

然后采用用整数规划的方法进行建模。

首先是价格函数，也就是目标函数，$cost_j$ 为第 j 周的原料购买费用，需要让所有周的原料购买费用之和尽可能小

$$cost_j = \sum_{i=0}^{24} y_{ij} \cdot price_i \tag{15}$$

其中 y_{ij} 指的是第 i 家第 j 周的供货量，也就是决策变量，而 $price_i$ 指的是第 i 家供应商的原料单价，显然有

$$price_i = \begin{cases} 1.2 & type_i = A \\ 1.1 & type_i = B \\ 1 & type_i = C \end{cases} \tag{16}$$

可得整数规划模型：

$$\begin{aligned} \min Cost_{price} &= \sum_{j=0}^{23} cost_j = \sum_{j=0}^{23} \sum_{i=0}^{24} y_{ij} \cdot price_i \\ \text{s.t.} \quad & \sum_{i=0}^{24} \frac{y_{ij}}{w_{ij}} + stock_j - P_t \geqslant P_t \\ & 0 \leqslant y_{ij} \leqslant bound_{ij} \end{aligned} \tag{17}$$

第一个约束条件指的是每周的剩余原料要保证两周的产能，第二个约束条件给定了供应量的范围，w_i 表示的是原料和产能的关系，$stock_j$ 表示的是第 j 周的库存。

$$stock_j = \begin{cases} 0 & j = 0 \\ \sum_{i=0}^{24} \frac{y_{ij}}{w_{ij}} + stock_{j-1} - P_t & j \neq 0 \end{cases} \qquad w_i = \begin{cases} 0.6 & type_i = A \\ 0.66 & type_i = B \\ 0.72 & type_i = C \end{cases} \tag{18}$$

在获得了最经济的供货方案后，因为供货方案与订货量方案有偏差，需要根据供货商们的历史数据（订货量与实际供货的差别）来制定最后的订货方案。

图 4为一家供应商订货与对应的供货的散点图（代码见附件），同时还有直线 $y = x$ 以作比较，发现在很多情况下在给出大订单后供应商只能提供少部分的供货，对应于图像的右下角的点。实际上系数矩阵限制了供货上限，我们的模型不会在供应商没有能力供货的区间订极大量的货物，所以我们认为这样的情况不具有参考价值。将供货量小于订货量的 80% 的点剔除后，也就是把位于直线 $y = 0.8x$ 以下的点排除，然后再用剩下的点求供应商的平均供货率 α（供货量/订货量）。

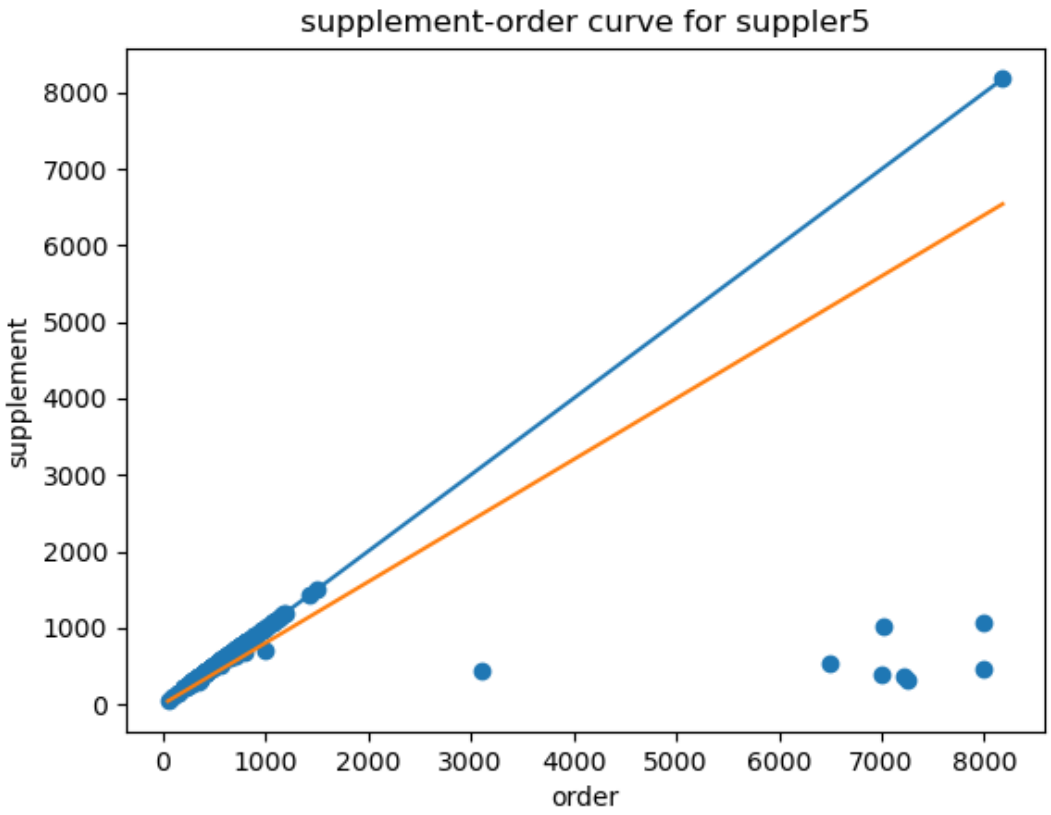

图 4: 示例供应商的散点图以及曲线 $y = 0.8x$

5.2.4 第二问模型的求解与分析

根据整数规划的模型不难求得最经济的供货方案，在此前也尝试过用参数矩阵 coe_{ij} 乘以其它不同参数，而不是 $\max(i)$ 来确定上限 $bound_{ij}$，但是几乎都不能满足第一个约束条件，所以最后还是用上述的方法来确定供货上限。

在求得 25×24（25 家、24 周）的供货方案矩阵 S 后，把第 i 家供应商的供货除以其平均供货率，得到最终的订货矩阵 B。

现分析一下第二题第二问的整数规划模型的实际效果。首先由于 A 类和 B 类的采购价格分别为 C 类价格的 1.2 倍和 1.1 倍，设 C 类原料的采购价格为 $Price_C$，则 $Price_A = 1.2Price_C$，$Price_B = 1.1Price_C$，同时由于生产单位体积 (m^3) 产品消耗的不同原材料的体积分别为 $Vol_A = 0.6m^3$，$Vol_B = 0.66m^3$，$Vol_C = 0.72m^3$，则采用不同原材料生产单位体积产品所需要的采购成本分别为

$$\begin{aligned} Cost_C &= Price_C \cdot Vol_C = 0.72Price_C \\ Cost_A &= Price_A \cdot Vol_A = 0.72Price_C \\ Cost_B &= Price_B \cdot Vol_B = 0.726Price_C \end{aligned} \tag{19}$$

因此在不考虑运损和储存成本的情况下，最经济的采购方式为尽可能采购 A 类和 C 类原材料用来生产。模型计算出的结果显示，在每一周的采购方案中，均为优先采购 A 类和 C 类产品以满足生产需求，当无法满足时再补充采购 B 类产品，与实际相符。

把上述模型求得的 24 周的采购成本绘图呈现出来，得到图 5。

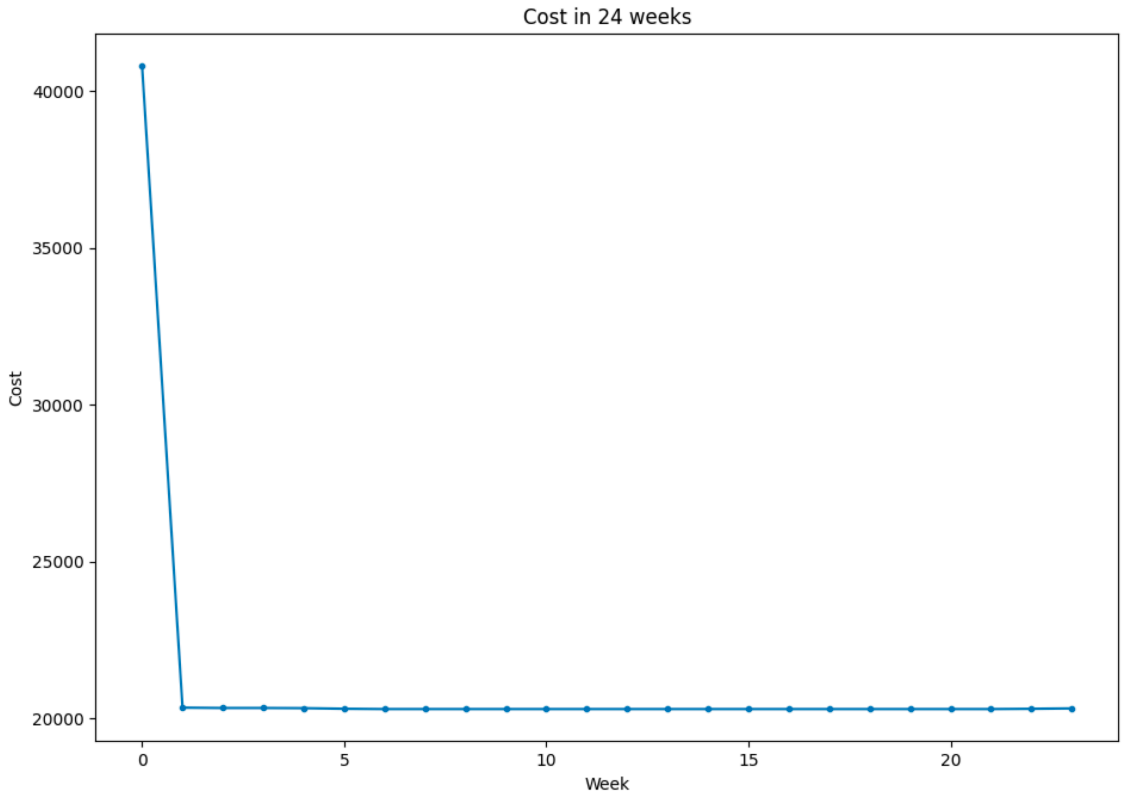

图 5: 最经济的供货计划的 24 周的采购成本折线图

图像显示从第二周开始采购成本基本为定值，第一周订购的价格则为之后的两倍。由于需要保证至少两周产能 $2P_t$ 的库存量，所以第一周在库存为 0 的时候需要有 $2P_t$ 的库存量，之后由于已经有了一倍的库存，所以只需要购买一周产能的原料即可，也就是 P_t，而图像与上述理论分析十分吻合，即本模型在选出来的 25 家供货商的供货能力下达到了相当可观的效果。

此外值得注意的是，本模型在通过供应量计算订单量的过程中没有考虑损耗量，主要是因为后，但经过分析实际并不需要过多考虑损耗量。之前建立了 25 家供应商的订货量与供货量的关系的模型，运行得出的数据为表格 2：

1 - 5	6 - 10	11-15	16-20	21-25
1.0002352	1.0032156	1.0237724	1.0239607	1.0434217
1.0039813	1.0033100	1.0184186	1.0125897	1.0082938
1.0011539	1.0072018	1.0198967	0.9971785	1.0110397
1.0039939	1.0058418	1.0233171	1.0358356	1.0279503
1.0033693	0.9984736	1.0374447	0.9942775	1.1008827

表 2: 25 家供应商的供货量与订单量的比值

可以发现从历史数据的角度，每家供货商在他们接受的订单范围内都倾向于提供多于订单量的供货量。而在建立损耗量模型的时候发现，8 家转运商在一个 24 周的阶段内的平均损耗量也时非常小的量，见表 3。

1 - 4	5 - 8
0.0190476917	0.0244725211
0.0092137042	0.0048986907
0.0092137042	0.0207883333
0.0157048235	0.0100231706

表 3: 8 家转运商在一个 24 周阶段内的平均损耗量

经过计算和比较可以发现，运输过程中的损耗量与供货量超过订单量的值大致相互抵消，因此根据上一步建立的最经济订货量模型，也就是第二题第二小问中构建的模型（其中没有考虑损耗量）确定的订单量在考虑损耗率的情况下依旧可以满足企业的产能需求，同时该模型的目标又是最经济的，因此模型效果极佳。

5.2.5 第三问模型的建立

在制定好一个供货方案后，需要合理安排转运商来运货，目标是使得损耗最小，模型大概有如下的几条约束，每家转运商的运输能力为 6000 立方米/周，而一家供应商每周供应的原材料尽量由一家转运商运输，同样构建整数模型求解。

绘制了 8 家转运商的损耗率的折线图 6（代码详见附录 Listing4），

其中第 1、2 家的损耗率比较平稳，而第 0、4、5、6、7 家的损耗率呈周期性变化，而第 3 家的损耗率波动较大。

根据上述的情况，采用不同的方法计算不同商家的损耗，对于 1、2 家转运商，先求出它们的损耗率平均值（忽略损耗率为 0 的值，也就是没有进行转运的周），把损耗率矩阵的对应行全部赋成该值，对于损耗率不稳定的 0、4、5、6、7 家，取平均值不够合理，需要换一种方式，可以把

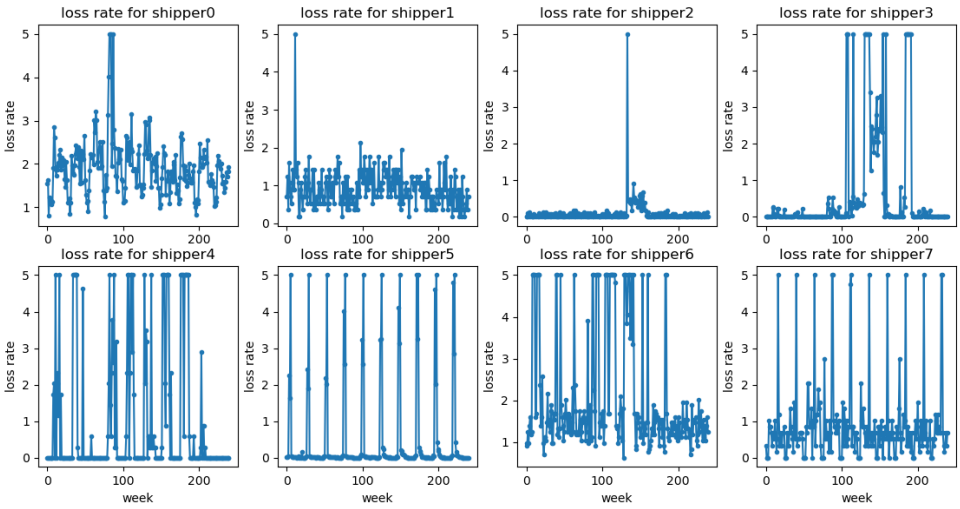

图 6: 8 家转运商的损耗率的折线图

它们 240 周的数据分成 10 份，相当于 10 个 24 维的向量，取这些向量的平均后得到一个 24 维向量，然后将其放到损耗率矩阵的对应行上，用这个方式可以让损耗率的周期性也能得到相应的体现。对于第 3 家，仍然采用求损耗率平均值的办法，因为其除了一部分的极端数据，大部分数据都是相对平稳的，用平均值也可以看作是它的那几次极端值带来的惩罚，拉高了整体的损失率，但同时也保留了其竞争能力。

最后得到 8×24（8 家，24 周）的损耗率矩阵 L，l_{ij} 表示第 i 家转运商在第 j 周的损耗率。设决策变量为矩阵 X，它的元素 x_{ijk} 表示在第 k 周，第 j 家转运商转运第 i 家供应商的的货物数量，此外还有 25×24 的供货数据矩阵 S 整数规划模型为

$$\begin{aligned} \min TotalLoss &= \sum_{k=0}^{23}\sum_{j=0}^{7}\sum_{i=0}^{24}\frac{x_{ijk}\cdot l_{ik}}{w_i} \\ \text{s.t.} \quad & \sum_{j=0}^{7} x_{ijk} = s_{ik} \\ & \sum_{i=0}^{24} x_{ij} \leqslant 6000 \end{aligned} \tag{20}$$

上述规划模型并没有把“一家供应商每周供应的原材料尽量由一家转运商运输”这个约束条件包含在内，首先因为在订购方案中存在某一家供货商单周供货量大于 6000 平方米的情况，多于一家转运商的情况不可避免。其次为了让模型尽量达到最优解，我们不妨放宽这一约束，在得到求解结果后进行评估。

5.2.6 第三问模型的求解与分析

根据上述的整数规划模型，得到决策变量矩阵 X，根据这个结果，发现尽管没有直接包含上述的约束条件，但是每周仅有一两家供应商由两到三家转运商分开运输，在大部分情况下，都是

由一家转运商完全负责一家供应商的，该模型在给出了最经济、损耗最少的方案的同时，也尽量满足了约束条件，所以我们认为这个模型的最终结果令人满意。

5.3 问题三模型的建立与求解

5.3.1 模型的建立

根据问题三的分析，因为用同样的价格购买 A 和 C，最终获得的产品数量是相同的，而 C 所占的体积更大，所以只要考虑转运以及仓储的成本，那么 C 的值一定会有所降低，但会有一个下限，因为只提供 A 和 B 类原料可能没法满足生产需要。由于题目要求尽量多地采购 A 类，尽量少地采购 C 类，所以只需要逐步提高转运以及仓储的成本（相当于加权），当转运、仓储费远大于原料单价时模型就会转而向减少供货体积量优化。最优情况时 A 的供货量应达到 A 的供货限制，此时继续增大转运仓储费结果应不变。

设 C_s 为单位体积的仓储费，C_t 为单位体积的运输费，而 24 周的生产计划中除了第一周进货了大约两倍产能的原料（要保持不少于满足两周生产需求的原材料库存量），此后的每一周都是进大约一倍产能的量，所以可以近似地把仓储费用看成每周实际到货的原料体积乘以 C_s，而运输费为供货量乘以 C_s，可以得到总仓储费 $Cost_{store}$ 和运输费 $Cost_{transfer}$ 为

$$\begin{aligned} Cost_{store} &= \sum_{j=0}^{23}\sum_{i=0}^{24} y_{ij}\cdot C_s \\ Cost_{transfer} &= \sum_{j=0}^{23}\sum_{i=0}^{24} y_{ij}\cdot C_t \end{aligned} \tag{21}$$

总的花费就变成了 $Cost_{price}, Cost_{store}, Cost_{transfer}$ 三者之和，参考式 (17)，不难写出新的整数规划模型，求得新的订购方案：

$$\begin{aligned} \min Cost_{total} &= \sum_{j=0}^{23}\sum_{i=0}^{24} y_{ij}\cdot(price_i + C_s + C_t) \\ \text{s.t.} \quad & \sum_{i=0}^{24}\frac{y_{ij}}{w_{ij}} + stock_j - P_t \geqslant 2P_t \\ & 0 \leqslant y_{ij} \leqslant bound_{ij} \end{aligned} \tag{22}$$

对于第二问，可以利用第二题第三问的模型 (20)，根据现在的订购方案求得损耗最少的转运方案。

5.3.2 模型的求解

对模型 (22) 进行整数规划，通过不断增加 $C_s + C_t$ 的值来降低 C 的供量，直到其不再改变，最后把 24 周内 C 的最少的供货量编成表格 4。

从表 4中可以看到，原料 C 的供应普遍达到了较小的值，由此可以得出该模型在很大程度上满足了企业的“尽量多地采购 A 类和尽量少地采购 C 类原材料”的要求

1 - 4	5 - 8	9 -12	13-16	17-20	21-24
1	7	0	0	9	8
1452	1474	0	0	0	0
7	633	7	6	221	989
497	10	6	8	6	7

表 4: 在 24 周内 C 最少的供货量（单位：立方米）

在获得了新的供货数据之后，按照之前求出的平均供货率 α 计算最后的订货单。

按照模型 (20) 的方法，同样用 Python 进行整数规划的求解，求出损耗最少的转运方案。

5.4 问题四模型的建立与求解

5.4.1 模型的建立

这一题因为需要考虑企业产能的提高，所以不再局限于之前求得的 25 家比较重要的供应商，而是应该把所有的供应商都考虑进来。

因为供应商供货应该存在季节性的波动，而现在需要给出的是的一年 48 周生产计划中的前 24 周的原料订购计划，所以可以利用 5 年中每年的前 24 周数据作为参考。

仿照问题 2 的第二小问进行建模，把 402 家供应商的每年的前 24 周的供货取平均值，获得一个 24 维的向量，然后把这些向量按列拼在一起获得 402×24 的平均值矩阵 M，矩阵的元素 m_{ij} 指的是第 i 家供应商五年的第 j 周的供货量的平均值。

同样地，为了把矩阵的值映射到 0 到 1 之间，让每一个元素除以其所在行的最大值，得到系数矩阵 coe_{ij}

$$coe_{ij} = \begin{cases} 0 & \max(m_i) = 0 \\ m_{ij}/\max(m_i) & \max(m_i) \neq 0 \end{cases} \tag{23}$$

其中，m_i 指的是矩阵 M 的第 i 行，$\max(m_i)$ 即为矩阵 M 的第 i 行所有数据的最大值。定义上限矩阵 $Bound$，其中 $bound_{ij}$ 指的是第 i 家供应商在第 j 周能提供的供货的上限

$$bound_{ij} = \lfloor \max(s_i) \times coe_{ij} \rfloor \tag{24}$$

这里向下取整是为了满足供货量一定是整数的隐性要求。因为这里考虑的主要目标是尽可能提高企业每周的产能，在 8 个转运商共 48000 立方米/周的的转运能力条件下，需要尽可能地提高单位面积的原料的产出，显然偏好程度为：A>B>C。

然后分别计算原料 A、B、C 的每周的最大供应量，分别记作 $boundA_j$，$boundB_j$ 和 $boundC_j$，下标 j 表示的是第 j 周的供应，有：

$$boundA_j = \sum_{i=1}^{402} bound_{ij} \cdot statusA_i \quad boundB_j = \sum_{i=1}^{402} bound_{ij} \cdot statusB_i \quad boundC_j = \sum_{i=1}^{402} bound_{ij} \cdot statusC_i \tag{25}$$

其中 $statusA_i$ 指的是第 i 家供应商供应的原料类型是否是 A，若是 A 则可以把其供应量的上限算在原料 A 的上限里，同理 B 和 C 也是如此，所以有

$$statusA_i = \begin{cases} 0 & type_i \neq A \\ 1 & type_i = A \end{cases} \quad statusB_i = \begin{cases} 0 & type_i \neq B \\ 1 & type_i = B \end{cases} \quad statusC_i = \begin{cases} 0 & type_i \neq C \\ 1 & type_i = C \end{cases} \tag{26}$$

因为每周总共的转运量为 48000 立方米，所以先全部分配给提供原料 A 的的供应商，若有剩余再给 B，若还有剩才给 C，最后可以求得每周的产能上限 $product_j$

$$product_j = boundA_j/0.6 + boundB_j/0.66 + boundC_j/0.72 \tag{27}$$

因为企业要尽可能保持不少于两周的的生产需求，所以第一周的产能需要除以 2，然后与后续所有的产能比较，取其中最小值，得出最后的目标产能的上限 $Capacity_{max}$

$$Capacity_{max} = \min(product_1/2, \quad product_i) \qquad i = 1, 2, 3, \cdots, 402 \tag{28}$$

最后得到的 $Capacity_{max}$ 只是一个理想状态的极大值，建模过程并没有考虑到转运过程中可能出现的损失，而规划求解时会计算损耗率。所以在后续规划求解时是不可能达到这个产能，需要把现在得到的产能乘以一个小于 1 的固定损耗系数作为结果。在规划求解的过程中设置固损系数为 0.95。

5.4.2 模型的求解与分析

通过第二问第三小问的模型，将得到的 $Capacity$ 代入，即可通过整数规划求解。最后提升后的产能为 35747.06 立方米，相比 2.82 万立方米提高了 26.76%。将求得的订购和转运方案填入到对应表格中。

[1]

模型的评价、改进与推广

6.1 模型的优点

我们构建的模型有如下的优点：1. “缺失评价模型”采取主观评价指标较少，相比之下更为客观，置信度更高。

2. 用 TOPSIS 算法继续进行了建模，并将其与“缺失评价模型”进行比较，使得最终结果更有说服力，更具稳健性。3. 根据实际数据特征进行了不同处理，比如 8 家转运商的损失率的计算、基于历史数据判断还是进行预测等，使得建模更贴合实际情况。4. 几个规划模型尽可能地保证了模型的优化程度，适当放宽约束条件，比如第二题第三问没有严格保证每家供应商只由一家转运

商负责，但是获得的结果依旧相当符合预期，基本达到了当前约束条件下的最优情况。5. 不拘泥于模型本身的限制，更注重于解决问题，如“缺失评价模型”解决实际问题的过程十分简洁；“滑动窗口算法”最常用于算法编程题求解。与此同时我们也注重模型的简化，在原有模型基础上添加参数以解决特化或推广的其他问题。

6.2 模型的缺点

我们的模型主要的缺点是 1. 有些约束条件有放宽，结果没有严格按照企业的要求。2. 新颖的模型与算法的有效性和合理性没有得到严格的证明，导致结果可能有一定偏差。3. 模型之间的前后关联性很强，模型的量化分析与检验难度较大，误差可能逐步累积。

6.3 模型的改进

模型的改进 1. 在 TOPSIS 模型中计算得分的时候，直接把三个指标的权重相加，即默认它们的权重是相同的，实际可以考虑不同的权重，使得最终的结果更合理。2. 之所以适当地放开约束的一个关键原因是约束使得部分模型比较复杂，难以求解，所以进行适当的简化，放松了部分约束，如果能够求解这些模型的话，应该是可以求得更精确、更符合实际情况的结果的。

3. 在问题 4 求解提高的产能时，我们采用了固定损耗参数来让模型更具抗风险性。实际上可以根据历史数据再对损耗率另外建模，使得损耗参数随时间变化，更加合理。

6.4 模型的推广

我们提出的“缺失评价模型”实现十分方便，直接根据对于结果的影响来判断权重，更可观，效果相比从影响因素开始分析的建立主观指标的模型更占优势，只要有一个可以量化的最终结果（本题中即为与产量的差值），就可根据结果的变化来衡量影响程度。

参考文献

[1] 孙继红张男星. 基于 topsis 评价模型的“双一流”高校分类评价实证研究. 黑龙江高教研究, 38(8):36–43, 2020.

附录

附录 1
支撑材料的文件列表: 附件 A 订购方案数据结果.xlsx 附件 B 转运方案数据结果.xlsx 1_Missing-Accessment.py 1_TOPSIS-based.py 2_3-shipper.py 2_min-list-graph.py 2_y=x.py 3_1.py 4_1.py

以下代码是用 Python 编写的，为了解决第一题，其通过缺失评价模型算出供应商的得分，并选取前 50 名进行画图

Listing 1: **Missing-Accessment-Model**

```
import pandas as pd
import numpy as np

import matplotlib as mp
mp.use('tkagg')
import matplotlib.pyplot as plt


data = pd.read_excel(r"D:\Courses\additional\mathematical_modeling\题目\C\附件1 近5年402家供应商的相关数据.xlsx",
                     sheet_name='供应商的供货量（m§）')

data=data.iloc[0:,1:]
data=np.array(data)

ratio = 1                          #原料转换比
data_ref=np.zeros((402,240))       #将ABC的供货量转化为产品的生产量

production = np.zeros(240)         #每周总原料量
production_deal = np.zeros(240) #每周去掉特定生产商的原料量

stock = 0                          #库存
difference = 0                     #差值 中间变量
result = np.zeros(402)

```

```
for i in  range(402):
    if data[i][0]=='A':
        ratio = 0.6
    elif data[i][0]=='B':
        ratio = 0.66
    else:
        ratio = 0.72

    for j in  range(240):
        data_ref[i][j] = data[i][j+1]/ratio

for i in  range(240):
    for j in  range(402):
        production[i]+= data_ref[j][i]

for j in  range(402):
    stock = 0
    difference = 0
    for i in  range(240):
        production_deal[i]=production[i]-data_ref[j][i]
    for i in  range(240):
        if stock+production_deal[i]-28200 > 0:
            stock=stock+production_deal[i]-28200
            difference = 0
        else:
            difference = 28200 - stock - production_deal[i]
            stock = 0
        result[j]+=difference**2
    result[j]=(result[j]/240)**0.5

num = np.arange(1,403)

num_result =  zip(result,num)
list_result =  list(num_result)

list_result= sorted(list_result,key=(lambda x:x[0]),reverse=True)

x_list =  list(np.arange(240))

name_list_1 = []
name_list_2 = []

plt.title('Supplement-Week curve for 50 suppliers selected by Missing-Accessment model')
#缺失评估
plt.xlabel('Week')
plt.ylabel('Supplement')
```

```
71
72
73  for i in  range(50):
74      #print(list_result[i])
75      plt.plot(x_list, data[list_result[i][1]-1][1:], marker = 'o', markersize = 3)
76      if i in  range(25):
77          name_list_1.append('S' +  str(list_result[i][1]))
78      else :
79          name_list_2.append('S' +  str(list_result[i][1]))
80
81
82  #first_legend = plt.legend(name_list_1, loc=3)
83  #ax = plt.gca().add_artist(first_legend)
84  #plt.legend(name_list_2, loc = 4)
85
86  plt.show()
```

以下代码是用 Python 编写的，为了解决第一题，其通过 TOPSIS 模型算出供应商的得分，并选取前 50 名进行画图

Listing 2: **TOPSIS-Model**

```
1   import numpy as np
2   import pandas as pd
3
4   import matplotlib as mp
5   mp.use('tkagg')
6   import matplotlib.pyplot as plt
7
8
9   M = np.matrix(np.array([[1.,3],[4,2]]))
10  idxType = np.array([2,2])
11
12
13  def MatrixNormalization(M: np.matrix):
14      N = np.multiply(M, M)
15      M /= np.power(np. sum(N, 0), 1/2)
16
17
18  def ScoreCalculation(M: np.matrix) -> np.matrix:
19      n = M.shape[0]
20      Max_m = np.repeat(np. max(M,0), n, 0)
21      Min_m = np.repeat(np. min(M,0), n, 0)
22      D_P = np.power(np. sum(np.power(M - Max_m, 2), 1), 1/2)
23      D_N = np.power(np. sum(np.power(M - Min_m, 2), 1), 1/2)
24      return D_N / (D_N+D_P)
25
```

```
26
27  def Min2Max(Col: np.matrix) -> np.matrix:
28      return  max(Col) - Col
29
30
31  def Mid2Max(Col: np.matrix, best:  float) -> np.matrix:
32      M =  max( abs(Col-best))
33      return 1 -  abs(Col-best)/M
34
35
36  def Inter2Max(Col: np.matrix, a:  float, b:  float) -> np.matrix:
37      M =  max( max(Col)-a, b- min(Col))
38      n = Col.shape[0]
39      tmp = np.matrix(np.zeros(Col.shape))
40      for i in  range(n):
41          num = Col[i,0]
42          if(num > a):
43              tmp[i,0] = 1-(num-a) / M
44          elif(num < b):
45              tmp[i,0] = 1-(b-num) / M
46          else:
47              tmp[i,0] = 1
48      return tmp
49
50
51  def MatrixPositivization(M: np.matrix, idxType: np.array):
52      print("Processing matrix positivization")
53      n = M.shape[1]
54      if idxType.shape[0] != n:
55          print("inputs are not aligned")
56          exit(1)
57      for i,idx in  enumerate(idxType):
58          print("processing column:",i,end=":" )
59          if(idx == 0):     #极大型
60              print("max")
61          elif(idx == 1):   #极小型指标转化成极大型指标
62              print("min")
63              M[:,i] = Min2Max(M[:,i])
64          elif(idx == 2):   #中间型指标
65              print("mid")
66              best =  float( input("please input the ideal MIDDLE measure:"))
67              M[:,i] = Mid2Max(M[:,i], best)
68          elif(idx == 3):   #区间型指标
69              print("Inter")
70              Up =  float( input("please assign the UPPER bound of the range"))
71              Down =  float( input("please assign the LOWER bound of the range"))
```

```
            M[:,i] = Inter2Max(M[:,i], Up, Down)
            #print(N[:,i])
        else:
            print("Invalid idx type")
            exit(1)


supplierFile = r'D:\Courses\additional\mathematical_modeling\题目\C\附件1 近5年402家供应商的相关数据.xlsx'
bookingTable = pd.read_excel(supplierFile, sheet_name='企业的订货量（m§）')
supplyingTable = pd.read_excel(supplierFile, sheet_name='供应商的供货量（m§）')

#idxType = np.array([0,2,1,3])
bookingTable.head()

tableS = supplyingTable.values[:,2:]
judge = [0] * tableS.shape[0]
TotalWeeks = 240

for i,t in  enumerate(supplyingTable['材料分类']):
    if t == 'A': judge[i] = 0.6
    elif t == 'B': judge[i] = 0.66
    else: judge[i] = 0.72
judge = np.matrix(judge)
tableSW = tableS / judge.T
dailyProduction = np. sum(tableSW, axis=0)
SumS = np. sum(tableSW, axis=1)
AverageS = SumS / TotalWeeks

tableB = bookingTable.values[:,2:]
tableD = tableB - tableS
Diff = np.power((np.square(np. sum(tableS-tableB, axis=1)) / TotalWeeks), 1/2)
DiffW = (np.matrix(Diff).T) / AverageS
em = np.hstack((AverageS, DiffW, judge.T))
idxType = np.array([0,1,1])
MatrixNormalization(em)
MatrixPositivization(em, idxType)

result = ScoreCalculation(em)


rslt_list = result.argsort(axis=0)[-50:]
x_list =  list(np.arange(240))


select_list = []
```

```
for i in  range(50):
    cur =  str(rslt_list[i]+1)
    cur = cur[2:]
    cur = cur[:-2]
    cur = 'S' + cur
    select_list.append(cur)

name_list_1 = []
name_list_2 = []


plt.figure(figsize=(20,10))

plt.title('50 suppliers selected by TOPSIS-based model')
plt.xlabel('Week')
plt.ylabel('Supplement')


for i in  range(50):
    plt.plot(x_list, tableS[rslt_list[i][0]][0][0], marker = 'o', markersize = 3)
    if i in  range(25):
        name_list_1.append(select_list[i])
    else :
        name_list_2.append(select_list[i])

plt.show()
```

以下代码是用 Python 编写的，其绘制了第二题第一问的图像，也就是 30 家供应商在 24 周内的供货曲线。

Listing 3: **30-suppliers-in-24-weeks**

```
import pandas as pd
import numpy as np

import matplotlib as mp
mp.use('tkagg')
import matplotlib.pyplot as plt


data = pd.read_excel(r"D:\Courses\additional\mathematical_modeling\题目\C\附件1 近5年402家供应商的相关数据.xlsx",
                     sheet_name='供应商的供货量（m$）')

data=data.iloc[0:,1:]
data=np.array(data)

```

```
ratio = 1                       #原料转换比
data_ref=np.zeros((402,240))    #将ABC的供货量转化为产品的生产量

production = np.zeros(240)      #每周总原料量
production_deal = np.zeros(240) #每周去掉特定生产商的原料量

stock = 0                       #库存
difference = 0                  #差值 中间变量
result = np.zeros(402)

for i in  range(402):
    if data[i][0]=='A':
        ratio = 0.6
    elif data[i][0]=='B':
        ratio = 0.66
    else:
        ratio = 0.72

    for j in  range(240):
        data_ref[i][j] = data[i][j+1]/ratio

for i in  range(240):
    for j in  range(402):
        production[i]+= data_ref[j][i]

for j in  range(402):
    stock = 0
    difference = 0
    for i in  range(240):
        production_deal[i]=production[i]-data_ref[j][i]
    for i in  range(240):
        if stock+production_deal[i]-28200 > 0:
            stock=stock+production_deal[i]-28200
            difference = 0
        else:
            difference = 28200 - stock - production_deal[i]
            stock = 0
        result[j]+=difference**2
    result[j]=(result[j]/240)**0.5

num = np.arange(1,403)

num_result =  zip(result,num)
list_result =  list(num_result)

list_result= sorted(list_result,key=(lambda x:x[0]),reverse=True)
```

```
61
62  x_list =  list(np.arange(138, 138+24))
63
64  name_list_1 = []
65  name_list_2 = []
66
67  plt.title('least suppliers selected by Missing-Accessment model in 24 weeks')
68  #缺失评估
69  plt.xlabel('Week')
70  plt.ylabel('Supplement')
71
72  min_list = [360, 139, 107, 138, 329, 307, 355, 267, 305, 142, 347, 200, 36, 283, 30, 364, 39, 337,
        54, 345, 373, 73, 79, 85, 243, 209, 113, 77, 217, 149]
73  min_name_list =  list(np.arange(30))
74
75  print(data[1][0])
76
77  for i in  range(30):
78      #print(list_result[i])
79      plt.plot(x_list, data[min_list[i]][138:138+24], marker = 'o', markersize = 3)
80
81
82  #first_legend = plt.legend(name_list_1, loc=3)
83  #ax = plt.gca().add_artist(first_legend)
84  #plt.legend(name_list_2, loc = 4)
85  #plt.legend(min_name_list)
86
87  plt.show()
```

以下代码是用 Python 编写的，为了解决第二题第三问，其绘制了 8 家转运商在 240 周内的损失率图像

Listing 4: **8-shippers-loss-rate**

```
1  import pandas as pd
2  import numpy as np
3
4  import matplotlib as mp
5  mp.use('tkagg')
6  import matplotlib.pyplot as plt
7
8
9  data = pd.read_excel(r"D:\Courses\additional\mathematical_modeling\题目\C\附件1 近5年402家供应商的
       相关数据.xlsx",
10                 sheet_name='供应商的供货量（m$）')
11
12 data=data.iloc[0:,1:]
```

```
13 data=np.array(data)
14
15 ratio = 1                          #原料转换比
16 data_ref=np.zeros((402,240))       #将ABC的供货量转化为产品的生产量
17
18 production = np.zeros(240)         #每周总原料量
19 production_deal = np.zeros(240) #每周去掉特定生产商的原料量
20
21 stock = 0                          #库存
22 difference = 0                    #差值 中间变量
23 result = np.zeros(402)
24
25 for i in  range(402):
26     if data[i][0]=='A':
27         ratio = 0.6
28     elif data[i][0]=='B':
29         ratio = 0.66
30     else:
31         ratio = 0.72
32
33     for j in  range(240):
34         data_ref[i][j] = data[i][j+1]/ratio
35
36 for i in  range(240):
37     for j in  range(402):
38         production[i]+= data_ref[j][i]
39
40 for j in  range(402):
41     stock = 0
42     difference = 0
43     for i in  range(240):
44         production_deal[i]=production[i]-data_ref[j][i]
45     for i in  range(240):
46         if stock+production_deal[i]-28200 > 0:
47             stock=stock+production_deal[i]-28200
48             difference = 0
49         else:
50             difference = 28200 - stock - production_deal[i]
51             stock = 0
52         result[j]+=difference**2
53     result[j]=(result[j]/240)**0.5
54
55 num = np.arange(1,403)
56
57 num_result =  zip(result,num)
58 list_result =  list(num_result)
```

```
59
60 list_result= sorted(list_result,key=(lambda x:x[0]),reverse=True)
61
62 x_list =  list(np.arange(138, 138+24))
63
64 name_list_1 = []
65 name_list_2 = []
66
67 plt.title('least suppliers selected by Missing-Accessment model in 24 weeks')
68 #缺失评估
69 plt.xlabel('Week')
70 plt.ylabel('Supplement')
71
72 min_list = [360, 139, 107, 138, 329, 307, 355, 267, 305, 142, 347, 200, 36, 283, 30, 364, 39, 337,
       54, 345, 373, 73, 79, 85, 243, 209, 113, 77, 217, 149]
73 min_name_list =  list(np.arange(30))
74
75 print(data[1][0])
76
77 for i in  range(30):
78     #print(list_result[i])
79     plt.plot(x_list, data[min_list[i]][138:138+24], marker = 'o', markersize = 3)
80
81
82 #first_legend = plt.legend(name_list_1, loc=3)
83 #ax = plt.gca().add_artist(first_legend)
84 #plt.legend(name_list_2, loc = 4)
85 #plt.legend(min_name_list)
86
87 plt.show()
```

以下代码是用 Python 编写的，为了解决第二题第二问，绘制的运营商的订货与供货散点图以及 $y = x$ 和 $y = 0.8x$ 两条直线

Listing 5: **points-between-book-supply**

```
1  import pandas as pd
2  import numpy as np
3  import pulp
4
5  import matplotlib as mp
6  mp.use('tkagg')
7  import matplotlib.pyplot as plt
8
9
10
11 data = pd.read_excel(r"D:\Courses\additional\mathematical_modeling\题目\C\附件1 近5年402家供应商的
```

```
    相关数据.xlsx",
12                  sheet_name='供应商的供货量（m§）')
13  data_1=pd.read_excel(r"D:\Courses\additional\mathematical_modeling\题目\C\k附件1 近5年402家供应商
    的相关数据.xlsx",
14                  sheet_name='企业的订货量（m§）')
15
16  data=data.iloc[0:,1:]
17  data_1=data_1.iloc[0:,1:]
18
19  data=np.array(data)
20  data_1=np.array(data_1)
21
22  ratio = 1                          #原料转换比
23  data_ref=np.zeros((402,240))                  #将ABC的供货量转化为产品的生产量
24
25  production = np.zeros(240)       #每周总原料量
26  production_deal = np.zeros(240) #每周去掉特定生产商的原料量
27
28  stock = 0                          #库存
29  difference = 0                     #差值 中间变量
30  result = np.zeros(402)
31
32  for i in  range(402):
33      if data[i][0]=='A':
34          ratio = 0.6
35      elif data[i][0]=='B':
36          ratio = 0.66
37      else:
38          ratio = 0.72
39
40      for j in  range(240):
41          data_ref[i][j] = data[i][j+1]/ratio
42
43  for i in  range(240):
44      for j in  range(402):
45          production[i]+= data_ref[j][i]
46
47  for j in  range(402):
48      stock = 0
49      difference = 0
50      for i in  range(240):
51          production_deal[i]=production[i]-data_ref[j][i]
52      for i in  range(240):
53          if stock+production_deal[i]-28200 > 0:
54              stock=stock+production_deal[i]-28200
55              difference = 0
```

```
        else:
            difference = 28200 - stock - production_deal[i]
            stock = 0
        result[j]+=difference**2
    result[j]=(result[j]/240)**0.5

num = np.arange(402)

num_result =  zip(result,num)
list_result =  list(num_result)


list_result= sorted(list_result,key=(lambda x:x[0]),reverse=True)


first_50 = []

for i in  range(50):
    first_50.append(list_result[i][1])              #此处供应商序号从0开始，非从1开始

production_for_window=np.zeros(240)

count=0
data_for_window=np.zeros((50,240))

for i in  range(402):
    if i in first_50:
        data_for_window[count]=data_ref[i]
        count+=1
        for j in  range(240):
            production_for_window[j]+=data_ref[i][j]


satisfy=0

window_width=24     #滑动窗口宽度
start_week_set=[]      #满足条件的窗口 起始月份集合


for i in  range(240-window_width):
    index=i
    stock=0
    satisfy=1
    while index < i+window_width :
        if stock + production[index] > 3*28200:
            stock=stock + production[index]-28200
```

```
            index+=1
        else:
            satisfy=0
            break
    if satisfy==1:
        start_week_set.append(i)

print(start_week_set)
#print(production_for_window[82])
'''
plt.title('supply')  # 折线图标题
plt.rcParams['font.sans-serif'] = ['SimHei']  # 显示汉字
plt.xlabel('week')  # x轴标题
plt.ylabel('supply')  # y轴标题

'''
sol_list=[]
'''
for start_week in start_week_set:

    sum_matrix = np.zeros((50,24))

    for i in range(50):
        sum_matrix[i][0]=data_for_window[i][start_week]

    for i in range(50):
        for j in range(start_week+1,start_week+24):
            sum_matrix[i][j-start_week]=sum_matrix[i][j-1-start_week]+data_for_window[i][j]

    InvestLP = pulp.LpProblem("problem_for_window", sense=pulp.LpMinimize)

    types=[str(i) for i in range(1,51)]
    status = pulp.LpVariable.dicts("supplier",types,cat='Binary')

    InvestLP += pulp.lpSum([(status[i]) for i in types])


    for i in range(24):
        InvestLP += ((pulp.lpSum([(status[types[j]]*sum_matrix[j][i]) for j in range(50)])-i
            *28200)>=0)


    InvestLP.solve()

    print(InvestLP.name)
    print("Status :", pulp.LpStatus[InvestLP.status])  # 输出求解状态
```

```
    for v in InvestLP.variables():
        print(v.name, "=", v.varValue)  # 输出每个变量的最优值

    print("Min f(x) =", pulp.value(InvestLP.objective))  # 输出最优解的目标函数值

    #sol_list.append(pulp.value(InvestLP.objective))
    #if pulp.value(InvestLP.objective) == 21:

#print(sol_list)
#print(len(start_week_set))
'''
start_week=start_week_set[0]

sum_matrix = np.zeros((50,24))

for i in  range(50):
    sum_matrix[i][0]=data_for_window[i][start_week]

for i in  range(50):
    for j in  range(start_week+1,start_week+24):
        sum_matrix[i][j-start_week]=sum_matrix[i][j-1-start_week]+data_for_window[i][j]

InvestLP = pulp.LpProblem("problem_for_window", sense=pulp.LpMinimize)

types=['01','02','03','04','05','06','07','08','09']
for i in  range(10,51):
    types.append( str(i))


status = pulp.LpVariable.dicts("supplier",types,cat='Binary')

InvestLP += pulp.lpSum([(status[i]) for i in types])


for i in  range(24):
    InvestLP += ((pulp.lpSum([(status[types[j]]*sum_matrix[j][i]) for j in
        range(50)])-i*28200)>=2*28200)


InvestLP.solve()
print(InvestLP.name)
print("Status :", pulp.LpStatus[InvestLP.status])  # 输出求解状态
for v in InvestLP.variables():
    print(v.name, "=", v.varValue)  # 输出每个变量的最优值
print("Min f(x) =", pulp.value(InvestLP.objective))  # 输出最优解的目标函数值

```

```
count=0

least_suppliers=[]

for v in InvestLP.variables():
    if v.varValue==1:
        least_suppliers.append(count)
    count+=1

num_suppliers= len(least_suppliers)

material_type=[]
ratio_type={}
price_type={}

ratio_type['A']=0.6
ratio_type['B']=0.66
ratio_type['C']=0.72
price_type['A']=1.2
price_type['B']=1.1
price_type['C']=1

for i in least_suppliers:
    material_type.append(data[first_50[i]][0])


real_suppliers=[]

for i in least_suppliers:
    real_suppliers.append(first_50[i])

data_for_supply= np.zeros((num_suppliers,240)) #选定的21家供货商240周的供货情况 未转换为产品量
data_for_order = np.zeros((num_suppliers,240)) #选定的21家供货商240周的订货情况 未转换为产品量

count=0
for i in real_suppliers:
        data_for_order[count]=data_1[i][1:]
        data_for_supply[count]=data[i][1:]
        count+=1

hist_max=np.zeros(num_suppliers) # 选定的供货商历史供货峰值(转化为产品)

data_for_xxx= np.zeros((num_suppliers,240))

count=0
for i in real_suppliers:
```

```
238     data_for_xxx[count]=data_ref[i]
239     count+=1
240 
241 for i in  range(num_suppliers):
242         for k in  range(240):
243             hist_max[i]= max(hist_max[i],data_for_xxx[i][k])
244 
245 #print(hist_max)
246 
247 s=np.zeros(24)
248 for i in  range(num_suppliers):
249     for j in  range(24):
250         s[j]+=hist_max[j]
251 
252 
253 
254 one_list=np.zeros(24)
255 
256 stock=np.zeros(24)
257 for i in  range(1,24):
258     stock[i]=stock[i-1]+s[i-1]-28200
259     if stock[i] >=28200:
260         one_list[i]=1
261 
262 print(real_suppliers) #the index for each supplier
263 #print(stock)
264 #print(one_list)
265 #print(len(start_week_set))
266 #print(start_week_set)
267 
268 
269 flag = 1
270 #flag 0 order-supplement
271 #flag 1 supplement-order
272 
273 if flag == 0 :
274     for i in  range( len(real_suppliers)):
275         plt.title('order-supplement curve for suppler'+ str(i))
276         plt.xlabel('supplement')
277         plt.ylabel('order')
278 
279         x_list = data[real_suppliers[i]][1:] #supplement
280         y_list = data_1[real_suppliers[i]][1:] #order
281 
282         none_zero_cnt = 0
283         for i in  range( len(x_list)):
```

```
            if x_list[i] > 0:
                none_zero_cnt = none_zero_cnt + 1

        start =  min(x_list)
        end =  max(x_list)
        print(start, end, none_zero_cnt)

        ref_x = np.arange(start, end, 1)
        ref_y = ref_x

        plt.plot(ref_x, ref_y, marker = None)
        plt.scatter(x_list, y_list, marker='o')


        plt.show()
elif flag == 1 :
    for i in  range( len(real_suppliers)):
        plt.title('supplement-order curve for suppler'+ str(i))
        plt.ylabel('supplement')
        plt.xlabel('order')

        y_list = data[real_suppliers[i]][1:]
        x_list = data_1[real_suppliers[i]][1:]

        start =  min(x_list)
        end =  max(x_list)
        print(start, end,  len(x_list))

        ref_x = np.arange(start, end, 1)
        ref_y = ref_x

        ref_y1 = 0.8 * ref_x

        plt.plot(ref_x, ref_y, marker = None)
        plt.plot(ref_x, ref_y1, marker = None)
        plt.scatter(x_list, y_list, marker='o')


        plt.show()
else :
    pass



plt.title('order-supplement curve for supplier0')
```

```
plt.xlabel('supplement')
plt.ylabel('order')

index = 0

x_list = data[real_suppliers[index]][1:]
y_list = data_1[real_suppliers[index]][1:]

start =  min(x_list)
end =  max(x_list)
print(start, end,  len(x_list))

ref_x = np.arange(start, end, 1)
ref_y = ref_x

plt.plot(ref_x, ref_y, marker = None)
plt.scatter(x_list, y_list, marker='o')


plt.show()

#print(data)
```

以下代码是用 Python 编写的，为了解决第二题和第三题

Listing 6: **models-for-p2-p3**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pulp
import openpyxl


data    =   pd.read_excel("附件1 近5年402家供应商的相关数据.xlsx",
                sheet_name='供应商的供货量（m§）')
data_1  =   pd.read_excel("附件1 近5年402家供应商的相关数据.xlsx",
                sheet_name='企业的订货量（m§）')
data_2  =   pd.read_excel("附件2 近5年8家转运商的相关数据.xlsx",
                sheet_name = '运输损耗率（%）')

data    =   data.iloc[0:,1:]
data_1  =   data_1.iloc[0:,1:]
data_2  =   data_2.iloc[0:,1:]

data=np.array(data)
data_1=np.array(data_1)
```

```
data_2=np.array(data_2)

ratio = 1                       #原料转换比
data_ref=np.zeros((402,240))                #将ABC的供货量转化为产品的生产量

production = np.zeros(240)      #每周总原料量
production_deal = np.zeros(240) #每周去掉特定生产商的原料量

stock = 0                       #库存
difference = 0                  #差值 中间变量
result = np.zeros(402)

for i in  range(402):
    if data[i][0]=='A':
        ratio = 0.6
    elif data[i][0]=='B':
        ratio = 0.66
    else:
        ratio = 0.72

    for j in  range(240):
        data_ref[i][j] = data[i][j+1]/ratio

for i in  range(240):
    for j in  range(402):
        production[i]+= data_ref[j][i]

for j in  range(402):
    stock = 0
    difference = 0
    for i in  range(240):
        production_deal[i]=production[i]-data_ref[j][i]
    for i in  range(240):
        if stock+production_deal[i]-28200 > 0:
            stock=stock+production_deal[i]-28200
            difference = 0
        else:
            difference = 28200 - stock - production_deal[i]
            stock = 0
        result[j]+=difference**2
    result[j]=(result[j]/240)**0.5

num = np.arange(402)

num_result =  zip(result,num)
list_result =  list(num_result)
```

```
67
68
69  list_result= sorted(list_result,key=(lambda x:x[0]),reverse=True)
70
71
72  first_50 = []
73
74  for i in  range(50):
75      first_50.append(list_result[i][1])              #此处供应商序号从0开始，非从1开始
76
77  production_for_window=np.zeros(240)
78
79  count=0
80  data_for_window=np.zeros((50,240))
81
82  for i in  range(402):
83      if i in first_50:
84          data_for_window[count]=data_ref[i]
85          count+=1
86          for j in  range(240):
87              production_for_window[j]+=data_ref[i][j]
88
89
90  satisfy=0
91
92  window_width=24     #滑动窗口宽度
93  start_week_set=[]      #满足条件的窗口 起始月份集合
94
95
96  for i in  range(240-window_width):
97      index=i
98      stock=0
99      satisfy=1
100     while index < i+window_width :
101         if stock + production[index] > 2*28200:
102             stock=stock + production[index]-28200
103             index+=1
104         else:
105             satisfy=0
106             break
107     if satisfy==1:
108         start_week_set.append(i)
109
110 #print(start_week_set)
111 #print(production_for_window[82])
112 '''
```

```
plt.title('supply')  # 折线图标题
plt.rcParams['font.sans-serif'] = ['SimHei']  # 显示汉字
plt.xlabel('week')  # x轴标题
plt.ylabel('supply')  # y轴标题

'''
# sol_list=[]
'''
for start_week in start_week_set:

    sum_matrix = np.zeros((50,24))

    for i in range(50):
        sum_matrix[i][0]=data_for_window[i][start_week]

    for i in range(50):
        for j in range(start_week+1,start_week+24):
            sum_matrix[i][j-start_week]=sum_matrix[i][j-1-start_week]+data_for_window[i][j]

    InvestLP = pulp.LpProblem("problem_for_window", sense=pulp.LpMinimize)

    types=[str(i) for i in range(1,51)]
    status = pulp.LpVariable.dicts("supplier",types,cat='Binary')

    InvestLP += pulp.lpSum([(status[i]) for i in types])


    for i in range(24):
        InvestLP += ((pulp.lpSum([(status[types[j]]*sum_matrix[j][i]) for j in range(50)])-i
            *28200)>=0)


    InvestLP.solve()

    print(InvestLP.name)
    print("Status :", pulp.LpStatus[InvestLP.status])  # 输出求解状态
    for v in InvestLP.variables():
        print(v.name, "=", v.varValue)  # 输出每个变量的最优值

    print("Min f(x) =", pulp.value(InvestLP.objective))  # 输出最优解的目标函数值

    #sol_list.append(pulp.value(InvestLP.objective))
    #if pulp.value(InvestLP.objective) == 21:

#print(sol_list)
#print(len(start_week_set))
```

```
'''
start_week=start_week_set[2]

sum_matrix = np.zeros((50,24))

for i in  range(50):
    sum_matrix[i][0]=data_for_window[i][start_week]

for i in  range(50):
    for j in  range(start_week+1,start_week+24):
        sum_matrix[i][j-start_week]=sum_matrix[i][j-1-start_week]+data_for_window[i][j]

InvestLP = pulp.LpProblem("problem_for_window", sense=pulp.LpMinimize)

types=['01','02','03','04','05','06','07','08','09']
for i in  range(10,51):
    types.append( str(i))


status = pulp.LpVariable.dicts("supplier",types,cat='Binary')

InvestLP += pulp.lpSum([(status[i]) for i in types])


for i in  range(24):
    InvestLP += ((pulp.lpSum([(status[types[j]]*sum_matrix[j][i]) for j in
        range(50)])-i*28200)>=28200)


InvestLP.solve()
print(InvestLP.name)
print("Status :", pulp.LpStatus[InvestLP.status])  # 输出求解状态
for v in InvestLP.variables():
    print(v.name, "=", v.varValue)  # 输出每个变量的最优值
print("Min f(x) =", pulp.value(InvestLP.objective))  # 输出最优解的目标函数值

sup_list=[]

count = 0
for v in InvestLP.variables():
    if v.varValue==1:
        sup_list.append(count)
    count+=1


#
```

```
    ##########################################################################################################################

real_list=[]

for i in sup_list:
    real_list.append(first_50[i])




count=0

least_suppliers=[]

for v in InvestLP.variables():
    if v.varValue==1:
        least_suppliers.append(count)
    count+=1

num_suppliers= len(least_suppliers)

material_type=[]
ratio_type={}
price_type={}

ratio_type['A']=0.6
ratio_type['B']=0.66
ratio_type['C']=0.72
price_type['A']=1.2
price_type['B']=1.1
price_type['C']=1


real_suppliers=[]

for i in least_suppliers:
    real_suppliers.append(first_50[i])


for i in real_suppliers:
    material_type.append(data[i][0])


data_for_supply= np.zeros((num_suppliers,240)) #选定的21家供货商240周的供货情况 未转换为产品量
data_for_order = np.zeros((num_suppliers,240)) #选定的21家供货商240周的订货情况 未转换为产品量

```

```
count=0
for i in real_suppliers:
        data_for_order[count]=data_1[i][1:]
        data_for_supply[count]=data[i][1:]
        count+=1

hist_max=np.zeros(num_suppliers) # 选定的供货商历史供货峰值(转化为产品)

data_for_xxx= np.zeros((num_suppliers,240))

count=0
for i in real_suppliers:
    data_for_xxx[count]=data_ref[i]
    count+=1

for i in  range(num_suppliers):
        for k in  range(240):
            hist_max[i]= max(hist_max[i],data_for_xxx[i][k])

#print(hist_max)

s=np.zeros(24)
for i in  range(num_suppliers):
    for j in  range(24):
        s[j]+=hist_max[j]



one_list=np.zeros(24)

stock=np.zeros(24)
for i in  range(1,24):
    stock[i]=stock[i-1]+s[i-1]-28200
    if stock[i] >=28200:
        one_list[i]=1

#print(real_suppliers)
#print(stock)
#print(one_list)
#print(len(start_week_set))
#print(start_week_set)

avr_matrix=np.zeros((num_suppliers,24))

for i in  range(num_suppliers):
    for j in  range(24):
```

```
293         for k in  range(5):
294             avr_matrix[i][j]+=data_for_xxx[i][k*48+j]
295         avr_matrix[i][j]/=5
296 
297 base_value=np.zeros(num_suppliers) #平均值峰值 系数计算基准
298 
299 for i in  range(num_suppliers):
300     for j in  range(24):
301         base_value[i]= max(base_value[i],avr_matrix[i][j])
302 
303 coefficient_matrix=np.zeros((num_suppliers,24))
304 
305 for i in  range(num_suppliers):
306     for j in  range(24):
307         coefficient_matrix[i][j]=(avr_matrix[i][j]/base_value[i])*0.88+0.12
308 
309 Bound=np.zeros((num_suppliers,24))
310 
311 real_max=np.zeros(num_suppliers) # 选定的供货商历史供货峰值(转化为产品)
312 
313 data_for_bound= np.zeros((num_suppliers,240))
314 
315 count=0
316 for i in real_suppliers:
317     data_for_bound[count]=data[i][1:]
318     count+=1
319 
320 for i in  range(num_suppliers):
321     for k in  range(240):
322         real_max[i]= max(real_max[i],data_for_bound[i][k])
323 
324 for i in  range(num_suppliers):
325     for j in  range(24):
326         Bound[i][j]=real_max[i]*coefficient_matrix[i][j]
327         #Bound[i][j]=real_max[i]
328 
329 
330 stock_for_cost=np.zeros(24)
331 
332 status_list=[]
333 
334 result_2_2=np.zeros((num_suppliers,24))
335 
336 min_2_2=[]
337 
338 for index in  range(24):
```

```
339     costLP=pulp.LpProblem("cost_problem_"+ str(index),sense=pulp.LpMinimize)
340
341     yield_list=[]
342
343     for i in  range(num_suppliers):
344         if i < 10:
345             yield_list.append(pulp.LpVariable('0'+ str(i),lowBound=0,upBound=40000,cat='Integer'))
346         else:
347             yield_list.append(pulp.LpVariable( str(i),lowBound=0,upBound=40000,cat='Integer'))
348
349     costLP += pulp.lpSum([(yield_list[i]*price_type[material_type[i]])for i in
            range(num_suppliers)])
350
351     costLP += ((pulp.lpSum([(yield_list[i]*(ratio_type[material_type[i]])**(-1) )for i in
            range(num_suppliers)])+stock_for_cost[index]-28200) >= 28200)
352
353     for i in  range(num_suppliers):
354         costLP += (yield_list[i]<=Bound[i][index])
355
356     costLP.solve()
357     print(costLP.name)
358     print("Status :", pulp.LpStatus[costLP.status])  # 输出求解状态
359     count=0
360     for v in costLP.variables():
361         result_2_2[count][index]=v.varValue
362         print(v.name, "=", v.varValue)  # 输出每个变量的最优值
363         count+=1
364
365     print("Min f(x) =", pulp.value(costLP.objective))  # 输出最优解的目标函数值
366
367     min_2_2.append(pulp.value(costLP.objective))
368
369     status_list.append(pulp.LpStatus[costLP.status])
370
371     sum_prod=0
372     count=0
373     for v in costLP.variables():
374         sum_prod+=v.varValue/ratio_type[material_type[count]]
375         count+=1
376     if index <23:
377         stock_for_cost[index+1]=stock_for_cost[index]+sum_prod-28200
378
379 #print(status_list)
380
381
382 #
      ##########################################################################################################
```

```
383
384 prob=np.zeros(num_suppliers)
385 count=0
386
387 for i in real_suppliers:
388     temp_sup=0
389     temp_ord=0
390     for j in  range(240):
391         if data_1[i][j+1]>0 and data[i][j+1]/data_1[i][j+1] >= 0.8 :
392             temp_ord+=data_1[i][j+1]
393             temp_sup+=data[i][j+1]
394     prob[count]=temp_sup/temp_ord
395     count+=1
396
397 #print(prob)
398
399 #for i in range(num_suppliers):
400     #result_2_2[i][0]=real_suppliers[i]
401     #for j in range(24):
402     #    result_2_2[i][j+1]=int(result_2_2[i][j+1]/prob[i])
403
404 dealt_shipper=np.zeros((8,24))
405
406 avg_list=[1,2,3]
407
408 for i in  range(8):
409     if i in avg_list:
410         count=0
411         Sum=0
412         for k in  range(240):
413             if data_2[i][k]>0:
414                 count+=1
415                 Sum+=data_2[i][k]
416         Sum=Sum/count
417         for j in  range(24):
418             dealt_shipper[i][j]=Sum
419     else:
420         for j in  range(24):
421             count=0
422             for k in  range(10):
423                 if data_2[i][24*k+j]!=0:
424                     dealt_shipper[i][j]+=data_2[i][24*k+j]
425                     count+=1
426             if count==0:
427                 c=0
```

```
                for k in  range(240):
                    if data_2[i][k]!=0:
                        c+=1
                        dealt_shipper[i][j]+=data_2[i][k]
                dealt_shipper[i][j]/=c
            else:
                dealt_shipper[i][j]/=count

#print(dealt_shipper)

result_2_3=[]

for index in  range(24):

    lossLP=pulp.LpProblem("loss_problem_"+ str(index),sense=pulp.LpMinimize)

    dec_list=[]

    par=np.zeros(num_suppliers*8)

    for i in  range(num_suppliers*8):
        par[i]=(dealt_shipper[i%8][index]/100)/ratio_type[material_type[ int(i/8)]]

    #print(dealt_shipper)
    #print(par)

    for i in  range(8*num_suppliers):
        if i < 10:
            dec_list.append(pulp.LpVariable('00'+ str(i),lowBound=0,upBound=6000,cat='Integer'))
        elif i <100:
            dec_list.append(pulp.LpVariable('0'+ str(i),lowBound=0,upBound=6000,cat='Integer'))
        else:
            dec_list.append(pulp.LpVariable( str(i),lowBound=0,upBound=6000,cat='Integer'))

    for i in  range(num_suppliers):
        lossLP += (pulp.lpSum([(dec_list[i*8+j])for j in  range(8)])==result_2_2[i][index])

    for j in  range(8):
        lossLP += (pulp.lpSum([(dec_list[i*8+j])for i in  range(num_suppliers)])<=6000)

    lossLP += pulp.lpSum([(dec_list[i]*par[i])  for i in  range(num_suppliers*8)])


    lossLP.solve()

    print(lossLP.name)
```

```
    print("Status :", pulp.LpStatus[lossLP.status])  # 输出求解状态
    for v in lossLP.variables():
        print(v.name, "=", v.varValue)  # 输出每个变量的最优值

    print("Min f(x) =", pulp.value(lossLP.objective))  # 输出最优解的目标函数值

    rlt=np.zeros((num_suppliers,8))

    #print(par)
    count=0
    for v in lossLP.variables():
        rlt[ int(count/8)][count%8]=v.varValue
        count+=1
        #print(v.name, "=", v.varValue)  # 输出每个变量的最优值

    result_2_3.append(rlt)
    #print(rlt)


#
    ##########################################################################################################################


stock_for_cost=np.zeros(24)

status_list=[]

result_3_1=np.zeros((num_suppliers,24))

for index in  range(24):
    costLP=pulp.LpProblem("cost_problem_2_"+ str(index),sense=pulp.LpMinimize)

    yield_list=[]

    for i in  range(num_suppliers):
        if i < 10:
            yield_list.append(pulp.LpVariable('0'+ str(i),lowBound=0,upBound=40000,cat='Integer'))
        else:
            yield_list.append(pulp.LpVariable( str(i),lowBound=0,upBound=40000,cat='Integer'))


    C=20  #转运+仓储的固定成本 参数
    costLP += pulp.lpSum([(yield_list[i]*(price_type[material_type[i]]+C))for i in
        range(num_suppliers)])

```

```
    costLP += ((pulp.lpSum([(yield_list[i]*(ratio_type[material_type[i]])**(-1) )for i in
        range(num_suppliers)])+stock_for_cost[index]-28200) >= 28200)

    for i in  range(num_suppliers):
        costLP += (yield_list[i]<=Bound[i][index])

    costLP.solve()
    print(costLP.name)
    print("Status :", pulp.LpStatus[costLP.status])  # 输出求解状态
    count=0
    for v in costLP.variables():
        result_3_1[count][index]=v.varValue
        print(v.name, "=", v.varValue)  # 输出每个变量的最优值
        count+=1

    print("Min f(x) =", pulp.value(costLP.objective))  # 输出最优解的目标函数值

    status_list.append(pulp.LpStatus[costLP.status])

    sum_prod=0
    count=0
    for v in costLP.variables():
        sum_prod+=v.varValue/ratio_type[material_type[count]]
        count+=1
    if index <23:
        stock_for_cost[index+1]=stock_for_cost[index]+sum_prod-28200

print(status_list)

#
    ##########################################################################################################################


#print(dealt_shipper)

result_3_2 = []

for index in  range(24):

    lossLP=pulp.LpProblem("loss_problem_"+ str(index),sense=pulp.LpMinimize)

    dec_list=[]

    par=np.zeros(num_suppliers*8)

```

```
    for i in  range(num_suppliers*8):
        par[i]=(dealt_shipper[i%8][index]/100)/ratio_type[material_type[ int(i/8)]]

    #print(dealt_shipper)
    #print(par)

    for i in  range(8*num_suppliers):
        if i < 10:
            dec_list.append(pulp.LpVariable('00'+ str(i),lowBound=0,upBound=6000,cat='Integer'))
        elif i <100:
            dec_list.append(pulp.LpVariable('0'+ str(i),lowBound=0,upBound=6000,cat='Integer'))
        else:
            dec_list.append(pulp.LpVariable( str(i),lowBound=0,upBound=6000,cat='Integer'))

    for i in  range(num_suppliers):
        lossLP += (pulp.lpSum([(dec_list[i*8+j])for j in  range(8)])==result_3_1[i][index])

    for j in  range(8):
        lossLP += (pulp.lpSum([(dec_list[i*8+j])for i in  range(num_suppliers)])<=6000)

    lossLP += pulp.lpSum([(dec_list[i]*par[i])  for i in  range(num_suppliers*8)])

    lossLP.solve()
    print(lossLP.name)
    print("Status :", pulp.LpStatus[lossLP.status])  # 输出求解状态
    for v in lossLP.variables():
        print(v.name, "=", v.varValue)  # 输出每个变量的最优值
    print("Min f(x) =", pulp.value(lossLP.objective))  # 输出最优解的目标函数值

    rlt=np.zeros((num_suppliers,8))

    count=0
    for v in lossLP.variables():
        rlt[ int(count/8)][count%8]=v.varValue
        count+=1
        #print(v.name, "=", v.varValue)  # 输出每个变量的最优值
    result_3_2.append(rlt)

#print(result_3_2[0])


print(prob)
print(real_list)

display_2_2=np.zeros((num_suppliers,24))
display_3_1=np.zeros((num_suppliers,24))
```

```
606
607 for i in  range(num_suppliers):
608     for j in  range(24):
609         display_2_2[i][j]= int(result_2_2[i][j]/prob[i])
610
611 for i in  range(num_suppliers):
612     for j in  range(24):
613         display_3_1[i][j]= int(result_3_1[i][j]/prob[i])
614
615 c_list=np.zeros(24)
616
617 for i in  range(num_suppliers):
618     if data[real_suppliers[i]][0]=='C':
619         for j in  range(24):
620             c_list[j]+=result_3_1[i][j]
621
622 for i in  range(24):
623     print(c_list[i])
624
625
626
627 workbook=openpyxl.load_workbook("附件A 订购方案数据结果.xlsx")
628 worksheet=workbook.worksheets[0]
629
630 for i in  range(num_suppliers):
631     for j in  range(24):
632         x=real_suppliers[i]
633         worksheet.cell(7+x,2+j,display_2_2[i][j])
634
635 worksheet=workbook.worksheets[1]
636
637 for i in  range(num_suppliers):
638     for j in  range(24):
639         x=real_suppliers[i]
640         worksheet.cell(7+x,2+j,display_3_1[i][j])
641
642
643 workbook.save(filename="附件A 订购方案数据结果.xlsx")
644
645 workbook=openpyxl.load_workbook("附件B 转运方案数据结果.xlsx")
646 worksheet=workbook.worksheets[0]
647
648
649 for i in  range(num_suppliers):
650     for j in  range(24):
651         for k in  range(8):
```

```
            x=real_suppliers[i]
            worksheet.cell(7+x,2+8*j+k,result_2_3[j][i][k])

worksheet=workbook.worksheets[1]


for i in  range(num_suppliers):
    for j in  range(24):
        for k in  range(8):
            x=real_suppliers[i]
            worksheet.cell(7+x,2+8*j+k,result_3_2[j][i][k])


workbook.save(filename="附件B 转运方案数据结果.xlsx")
```